```
; ****************************************************************************
; UNIX386.ASM (RETRO UNIX 386 Kernel) - v0.2.0.16
; ----------------------------------------------------------------------------
; NASM version 2.11 (unix386.s)
;
; RETRO UNIX 386 (Retro Unix == Turkish Rational Unix)
; Operating System Project (v0.2) by ERDOGAN TAN (Beginning: 24/12/2013)
;
; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
; (v0.1 - Beginning: 11/07/2012)
;
; [ Last Modification: 09/12/2015 ]
;
; Derived from UNIX Operating System (v1.0 for PDP-11)
; (Original) Source Code by Ken Thompson (1971-1972)
; <Bell Laboratories (17/3/1972)>
; <Preliminary Release of UNIX Implementation Document>
;
; Derived from 'UNIX v7/x86' source code by Robert Nordier (1999)
; UNIX V7/x86 source code: see www.nordier.com/v7x86 for details.
;
; ****************************************************************************

; 24/12/2013

; Entering protected mode:
; Derived from 'simple_asm.txt' source code file and
; 'The world of Protected mode' tutorial/article by Gregor Brunmar (2003)
; (gregor.brunmar@home.se)
; http://www.osdever.net/tutorials/view/the-world-of-protected-mode
;

; "The Real, Protected, Long mode assembly tutorial for PCs"
; by Michael Chourdakis (2009)
; http://www.codeproject.com/Articles/45788/
; http://www.michaelchourdakis.com
;

; Global Descriptor Table:
; Derived from 'head.s" source code of Linux v1.0 kernel
; by Linus Torvalds (1991-1992)
;

KLOAD  equ 10000h ; Kernel loading address
 ; NOTE: Retro UNIX 8086 v1 /boot code loads kernel at 1000h:0000h
KCODE  equ 08h     ; Code segment descriptor (ring 0)
KDATA  equ 10h     ; Data segment descriptor (ring 0)
; 19/03/2015
UCODE  equ 1Bh ; 18h + 3h  (ring 3)
UDATA  equ 23h ; 20h + 3h  (ring 3)
; 24/03/2015
TSS    equ 28h     ; Task state segment descriptor (ring 0)
; 19/03/2015
CORE   equ 400000h  ; Start of USER's virtual/linear address space
            ; (at the end of the 1st 4MB)
ECORE  equ 0FFC00000h ; End of USER's virtual address space (4GB - 4MB)
            ; ULIMIT = (ECORE/4096) - 1 = 0FFBFFh (in GDT)

; 27/12/2013
KEND    equ KLOAD + 65536 ; (28/12/2013) (end of kernel space)

; IBM PC/AT BIOS ----- 10/06/85 (postequ.inc)
;--------- CMOS TABLE LOCATION ADDRESS'S -------------------------------------
CMOS_SECONDS EQU   00H         ; SECONDS (BCD)
CMOS_MINUTES EQU   02H         ; MINUTES (BCD)
CMOS_HOURS   EQU   04H         ; HOURS (BCD)
CMOS_DAY_WEEKEQU   06H         ; DAY OF THE WEEK  (BCD)
CMOS_DAY_MONTH     EQU   07H         ; DAY OF THE MONTH (BCD)
CMOS_MONTH   EQU   08H         ; MONTH (BCD)
CMOS_YEAR    EQU   09H         ; YEAR (TWO DIGITS) (BCD)
CMOS_CENTURY EQU   32H         ; DATE CENTURY BYTE (BCD)
CMOS_REG_A   EQU   0AH         ; STATUS REGISTER A
CMOS_REG_B   EQU   00BH        ; STATUS REGISTER B  ALARM
CMOS_REG_C   EQU   00CH        ; STATUS REGISTER C  FLAGS
CMOS_REG_D   EQU   0DH         ; STATUS REGISTER D  BATTERY
CMOS_SHUT_DOWN     EQU   0FH         ; SHUTDOWN STATUS COMMAND BYTE
;----------------------------------------
;CMOS EQUATES FOR THIS SYSTEM ;
;-----------------------------------------------------------------------------
CMOS_PORT    EQU   070H        ; I/O ADDRESS OF CMOS ADDRESS PORT
CMOS_DATA    EQU   071H        ; I/O ADDRESS OF CMOS DATA PORT
NMI          EQU   10000000B   ; DISABLE NMI INTERRUPTS MASK -
                         ; HIGH BIT OF CMOS LOCATION ADDRESS

; Memory Allocation Table Address
; 05/11/2014
; 31/10/2014
MEM_ALLOC_TBLequ   100000h           ; Memory Allocation Table at the end of
                         ; the 1st 1 MB memory space.
                         ; (This address must be aligned
                         ;  on 128 KB boundary, if it will be
                         ;  changed later.)
```

```
                                                         ; ((lower 17 bits of 32 bit M.A.T.
                                                         ;   address must be ZERO)).
                                                         ; ((((Reason: 32 bit allocation
                                                         ;     instructions, dword steps)))
                                                         ; (((byte >> 12 --> page >> 5)))
                                 ;04/11/2014
                                 PDE_A_PRESENTequ   1          ; Present flag for PDE
                                 PDE_A_WRITE  equ   2          ; Writable (write permission) flag
                                 PDE_A_USER   equ   4          ; User (non-system/kernel) page flag
                                 ;
                                 PTE_A_PRESENTequ   1          ; Present flag for PTE (bit 0)
                                 PTE_A_WRITE  equ   2          ; Writable (write permission) flag (bit 1)
                                 PTE_A_USER   equ   4          ; User (non-system/kernel) page flag (bit 2)
                                 PTE_A_ACCESS    equ      32          ; Accessed flag (bit 5) ; 09/03/2015

                                 ; 17/02/2015 (unix386.s)
                                 ; 10/12/2014 - 30/12/2014 (0B000h -> 9000h) (dsectrm2.s)
                                 DPT_SEGM equ 09000h  ; FDPT segment (EDD v1.1, EDD v3)
                                 ;
                                 HD0_DPT equ 0          ; Disk parameter table address for hd0
                                 HD1_DPT equ 32         ; Disk parameter table address for hd1
                                 HD2_DPT equ 64         ; Disk parameter table address for hd2
                                 HD3_DPT equ 96         ; Disk parameter table address for hd3


                                 ; FDPT (Phoenix, Enhanced Disk Drive Specification v1.1, v3.0)
                                 ;      (HDPT: Programmer's Guide to the AMIBIOS, 1993)
                                 ;
                                 FDPT_CYLS    equ 0 ; 1 word, number of cylinders
                                 FDPT_HDS     equ 2 ; 1 byte, number of heads
                                 FDPT_TT      equ 3 ; 1 byte, A0h = translated FDPT with logical values
                                              ; otherwise it is standard FDPT with physical values
                                 FDPT_PCMP    equ 5 ; 1 word, starting write precompensation cylinder
                                              ; (obsolete for IDE/ATA drives)
                                 FDPT_CB      equ 8 ; 1 byte, drive control byte
                                             ; Bits 7-6 : Enable or disable retries (00h = enable)
                                             ; Bit 5     : 1 = Defect map is located at last cyl. + 1
                                             ; Bit 4 : Reserved. Always 0
                                             ; Bit 3 : Set to 1 if more than 8 heads
                                             ; Bit 2-0 : Reserved. Alsways 0
                                 FDPT_LZ      equ 12 ; 1 word, landing zone (obsolete for IDE/ATA drives)
                                 FDPT_SPT     equ 14 ; 1 byte, sectors per track

                                 ; Floppy Drive Parameters Table (Programmer's Guide to the AMIBIOS, 1993)
                                 ; (11 bytes long) will be used by diskette handler/bios
                                 ; which is derived from IBM PC-AT BIOS (DISKETTE.ASM, 21/04/1986).

                                 [BITS 16]       ; We need 16-bit intructions for Real mode

                                 [ORG 0]
                                  ; 12/11/2014
                                  ; Save boot drive number (that is default root drive)
00000000 8816[D46D]               mov   [boot_drv], dl ; physical drv number

                                  ; Determine installed memory
                                  ; 31/10/2014
                                  ;
00000004 B801E8                   mov   ax, 0E801h ; Get memory size
00000007 CD15                     int   15h      ; for large configurations
00000009 7308                     jnc   short chk_ms
0000000B B488                     mov   ah, 88h    ; Get extended memory size
0000000D CD15                     int   15h
                                  ;
                                  ;mov  al, 17h     ; Extended memory (1K blocks) low byte
                                  ;out  70h, al ; select CMOS register
                                  ;in   al, 71h ; read data (1 byte)
                                  ;mov  cl, al
                                  ;mov  al, 18h ; Extended memory (1K blocks) high byte
                                  ;out  70h, al ; select CMOS register
                                  ;in   al, 71h ; read data (1 byte)
                                  ;mov  ch, al
                                  ;
0000000F 89C1                     mov   cx, ax
00000011 31D2                     xor   dx, dx
                                 chk_ms:
00000013 890E[D06D]               mov   [mem_1m_1k], cx
00000017 8916[D26D]               mov   [mem_16m_64k], dx
                                  ; 05/11/2014
                                  ;and  dx, dx
                                  ;jz   short L2
0000001B 81F90004                        cmp     cx, 1024
0000001F 7315                     jnb   short L0
                                         ; insufficient memory_error
                                         ; Minimum 2 MB memory is needed...
                                  ; 05/11/2014
                                  ; (real mode error printing)
00000021 FB                       sti
00000022 BE[496C]                 mov   si, msg_out_of_memory
00000025 BB0700                   mov   bx, 7
00000028 B40E                     mov   ah, 0Eh     ; write tty
                                 oom_1:
0000002A AC                       lodsb
```

```
0000002B 08C0                         or    al, al
0000002D 7404                         jz    short oom_2
0000002F CD10                         int   10h
00000031 EBF7                         jmp   short oom_1
                                     oom_2:
00000033 F4                                  hlt
00000034 EBFD                         jmp   short oom_2

                                     L0:
                                     %include 'diskinit.inc' ; 07/03/2015
                                 <1> ; Retro UNIX 386 v1 Kernel - DISKINIT.INC
                                 <1> ; Last Modification: 10/07/2015
                                 <1> 
                                 <1> ; DISK I/O SYSTEM INITIALIZATION - Erdogan Tan (Retro UNIX 386 v1 project)
                                 <1> 
                                 <1> ; ///////// DISK I/O SYSTEM STRUCTURE INITIALIZATION ///////////////
                                 <1> 
                                 <1>  ; 10/12/2014 - 02/02/2015 - dsectrm2.s
                                 <1> ;L0:
                                 <1>  ; 12/11/2014 (Retro UNIX 386 v1 - beginning)
                                 <1>  ; Detecting disk drives... (by help of ROM-BIOS)
00000036 BA7F00                  <1>  mov   dx, 7Fh
                                 <1> L1:
00000039 FEC2                    <1>  inc   dl
0000003B B441                    <1>  mov   ah, 41h ; Check extensions present
                                 <1>                ; Phoenix EDD v1.1 - EDD v3
0000003D BBAA55                  <1>  mov   bx, 55AAh
00000040 CD13                    <1>  int   13h
00000042 721A                    <1>  jc    short L2
00000044 81FB55AA                <1>  cmp   bx, 0AA55h
00000048 7514                    <1>  jne   short L2
0000004A FE06[D76D]              <1>  inc   byte [hdc]  ; count of hard disks (EDD present)
0000004E 8816[D66D]              <1>         mov     [last_drv], dl  ; last hard disk number
00000052 BB[5A6D]                <1>  mov   bx, hd0_type - 80h
00000055 01D3                    <1>  add   bx, dx
00000057 880F                    <1>  mov   [bx], cl ; Interface support bit map in CX
                                 <1>               ; Bit 0 - 1, Fixed disk access subset ready
                                 <1>               ; Bit 1 - 1, Drv locking and ejecting ready
                                 <1>               ; Bit 2 - 1, Enhanced Disk Drive Support
                                 <1>                          ;            (EDD) ready (DPTE ready)
                                 <1>               ; Bit 3 - 1, 64bit extensions are present
                                 <1>                          ;            (EDD-3)
                                 <1>               ; Bit 4 to 15 - 0, Reserved
00000059 80FA83                  <1>  cmp   dl, 83h      ; drive number < 83h
0000005C 72DB                    <1>  jb    short L1
                                 <1> L2:
                                 <1>  ; 23/11/2014
                                 <1>  ; 19/11/2014
0000005E 30D2                    <1>  xor   dl, dl  ; 0
00000060 66BE[D86D0000]          <1>  mov   esi, fd0_type
                                 <1> L3:
                                 <1>  ; 14/01/2015
00000066 8816[D56D]              <1>  mov   [drv], dl
                                 <1>  ;
0000006A B408                    <1>  mov   ah, 08h ; Return drive parameters
0000006C CD13                    <1>  int   13h
0000006E 7215                    <1>  jc    short L4
                                 <1>        ; BL = drive type (for floppy drives)
                                 <1>        ; DL = number of floppy drives
                                 <1>        ;
                                 <1>        ; ES:DI = Address of DPT from BIOS
                                 <1>        ;
00000070 67881E                  <1>  mov   [esi], bl ;  Drive type
                                 <1>              ; 4 = 1.44 MB, 80 track, 3 1/2"
                                 <1>  ; 14/01/2015
00000073 E8BE02                  <1>  call  set_disk_parms
                                 <1>  ; 10/12/2014
00000076 6681FE[D86D0000]        <1>  cmp   esi, fd0_type
0000007D 7706                    <1>  ja    short L4
0000007F 6646                    <1>  inc   esi ; fd1_type
00000081 B201                    <1>  mov   dl, 1
00000083 EBE1                    <1>  jmp   short L3
                                 <1> L4:
                                 <1>  ; Older BIOS (INT 13h, AH = 48h is not available)
00000085 B27F                    <1>  mov   dl, 7Fh
                                 <1>  ; 24/12/2014 (Temporary)
00000087 803E[D76D]00            <1>  cmp   byte [hdc], 0 ; EDD present or not ?
0000008C 0F879000                <1>        ja      L10       ; yes, all fixed disk operations
                                 <1>               ; will be performed according to
                                 <1>               ; present EDD specification
                                 <1> L6:
00000090 FEC2                    <1>  inc   dl
00000092 8816[D56D]              <1>        mov     [drv], dl
00000096 8816[D66D]              <1>        mov     [last_drv], dl ; 14/01/2015
0000009A B408                    <1>  mov   ah, 08h ; Return drive parameters
0000009C CD13                    <1>  int   13h   ; (conventional function)
0000009E 0F829901                <1>        jc      L13      ; fixed disk drive not ready
000000A2 8816[D76D]              <1>        mov     [hdc], dl ; number of drives
                                 <1>  ;; 14/01/2013
                                 <1>  ;;push      cx
000000A6 E88B02                  <1>  call  set_disk_parms
                                 <1>  ;;pop cx
```

```
277                              <1>  ;
278                              <1>  ;;and cl, 3Fh      ; sectors per track (bits 0-6)
279 000000A9 8A16[D56D]          <1>         mov     dl, [drv]
280 000000AD BB0401              <1>  mov   bx, 65*4 ; hd0 parameters table (INT 41h)
281 000000B0 80FA80              <1>  cmp   dl, 80h
282 000000B3 7603                <1>  jna   short L7
283 000000B5 83C314              <1>  add   bx, 5*4      ; hd1 parameters table (INT 46h)
284                              <1> L7:
285 000000B8 31C0                <1>  xor   ax, ax
286 000000BA 8ED8                <1>  mov   ds, ax
287 000000BC 8B37                <1>         mov     si, [bx]
288 000000BE 8B4702              <1>         mov     ax, [bx+2]
289 000000C1 8ED8                <1>  mov   ds, ax
290 000000C3 3A4C0E              <1>         cmp     cl, [si+FDPT_SPT] ; sectors per track
291 000000C6 0F856D01            <1>         jne     L12 ; invalid FDPT
292 000000CA BF0000              <1>  mov   di, HD0_DPT
293 000000CD 80FA80              <1>  cmp   dl, 80h
294 000000D0 7603                <1>  jna   short L8
295 000000D2 BF2000              <1>  mov   di, HD1_DPT
296                              <1> L8:
297                              <1>  ; 30/12/2014
298 000000D5 B80090              <1>  mov   ax, DPT_SEGM
299 000000D8 8EC0                <1>  mov   es, ax
300                              <1>  ; 24/12/2014
301 000000DA B90800              <1>  mov   cx, 8
302 000000DD F3A5                <1>  rep   movsw  ; copy 16 bytes to the kernel's DPT location
303 000000DF 8CC8                <1>  mov   ax, cs
304 000000E1 8ED8                <1>  mov   ds, ax
305                              <1>  ; 02/02/2015
306 000000E3 8A0E[D56D]          <1>         mov     cl, [drv]
307 000000E7 88CB                <1>  mov   bl, cl
308 000000E9 B8F001              <1>  mov   ax, 1F0h
309 000000EC 80E301              <1>  and   bl, 1
310 000000EF 7406                <1>  jz    short L9
311 000000F1 C0E304              <1>  shl   bl, 4
312 000000F4 2D8000              <1>  sub   ax, 1F0h-170h
313                              <1> L9:
314 000000F7 AB                  <1>  stosw ; I/O PORT Base Address (1F0h, 170h)
315 000000F8 050602              <1>  add   ax, 206h
316 000000FB AB                  <1>  stosw ; CONTROL PORT Address (3F6h, 376h)
317 000000FC 88D8                <1>  mov   al, bl
318 000000FE 04A0                <1>  add   al, 0A0h
319 00000100 AA                  <1>  stosb ; Device/Head Register upper nibble
320                              <1>  ;
321 00000101 FE06[D56D]          <1>  inc   byte [drv]
322 00000105 BB[5A6D]            <1>  mov   bx, hd0_type - 80h
323 00000108 01CB                <1>  add   bx, cx
324 0000010A 800F80              <1>         or      byte [bx], 80h  ; present sign (when lower nibble is 0)
325 0000010D A0[D76D]            <1>  mov   al, [hdc]
326 00000110 FEC8                <1>  dec   al
327 00000112 0F842501            <1>         jz      L13
328 00000116 80FA80              <1>  cmp   dl, 80h
329 00000119 0F8673FF            <1>         jna     L6
330 0000011D E91B01              <1>         jmp     L13
331                              <1> L10:
332 00000120 FEC2                <1>  inc   dl
333                              <1>  ; 25/12/2014
334 00000122 8816[D56D]          <1>  mov   [drv], dl
335 00000126 B408                <1>  mov   ah, 08h ; Return drive parameters
336 00000128 CD13                <1>  int   13h   ; (conventional function)
337 0000012A 0F820D01            <1>         jc      L13
338                              <1>  ; 14/01/2015
339 0000012E 8A16[D56D]          <1>  mov   dl, [drv]
340 00000132 52                  <1>  push  dx
341 00000133 51                  <1>  push  cx
342 00000134 E8FD01              <1>  call  set_disk_parms
343 00000137 59                  <1>  pop   cx
344 00000138 5A                  <1>  pop   dx
345                              <1>  ;
346 00000139 66BE[54810000]      <1>  mov   esi, _end ; 30 byte temporary buffer address
347                              <1>               ; at the '_end' of kernel.
348 0000013F 67C7061E00          <1>  mov   word [esi], 30
349 00000144 B448                <1>  mov   ah, 48h      ; Get drive parameters (EDD function)
350 00000146 CD13                <1>  int   13h
351 00000148 0F82EF00            <1>         jc      L13
352                              <1>  ; 14/01/2015
353 0000014C 6629DB              <1>  sub   ebx, ebx
354 0000014F 88D3                <1>  mov   bl, dl
355 00000151 80EB80              <1>  sub   bl, 80h
356 00000154 6681C3[DA6D0000]    <1>  add   ebx, hd0_type
357 0000015B 678A03              <1>  mov   al, [ebx]
358 0000015E 0C80                <1>  or    al, 80h
359 00000160 678803              <1>  mov   [ebx], al
360 00000163 6681EB[D86D0000]    <1>  sub   ebx, hd0_type - 2 ; 15/01/2015
361 0000016A 6681C3[246E0000]    <1>  add   ebx, drv.status
362 00000171 678803              <1>  mov   [ebx], al
363 00000174 66678B4610          <1>  mov   eax, [esi+16]
364                              <1>  ; 28/02/2015
365 00000179 6621C0              <1>  and   eax, eax
366 0000017C 7416                <1>  jz    short L10_A0h
367                              <1>              ; 'CHS only' disks on EDD system
368                              <1>              ;  are reported with ZERO disk size
```

```
0000017E 6681EB[246E0000]    <1>  sub   ebx, drv.status
00000185 66C1E302            <1>  shl   ebx, 2
00000189 6681C3[086E0000]    <1>  add   ebx, drv.size ; disk size (in sectors)
00000190 66678903            <1>  mov   [ebx], eax
                             <1> L10_A0h: ; Jump here to fix a ZERO (LBA) disk size problem
                             <1>    ; for CHS disks (28/02/2015)
                             <1>  ; 30/12/2014
00000194 BF0000              <1>  mov   di, HD0_DPT
00000197 88D0                <1>  mov   al, dl
00000199 83E003              <1>  and   ax, 3
0000019C C0E005              <1>  shl   al, 5 ; *32
0000019F 01C7                <1>  add   di, ax
000001A1 B80090              <1>  mov   ax, DPT_SEGM
000001A4 8EC0                <1>  mov   es, ax
                             <1>  ;
000001A6 88E8                <1>  mov   al, ch      ; max. cylinder number (bits 0-7)
000001A8 88CC                <1>  mov   ah, cl
000001AA C0EC06              <1>  shr   ah, 6 ; max. cylinder number (bits 8-9)
000001AD 40                  <1>  inc   ax    ; logical cylinders (limit 1024)
000001AE AB                  <1>  stosw
000001AF 88F0                <1>  mov   al, dh      ; max. head number
000001B1 FEC0                <1>  inc   al
000001B3 AA                  <1>  stosb       ; logical heads (limits 256)
000001B4 B0A0                <1>  mov   al, 0A0h ; Indicates translated table
000001B6 AA                  <1>  stosb
000001B7 8A440C              <1>  mov   al, [si+12]
000001BA AA                  <1>  stosb        ; physical sectors per track
000001BB 31C0                <1>  xor   ax, ax
                             <1>  ;dec  ax     ; 02/01/2015
000001BD AB                  <1>  stosw        ; precompensation (obsolete)
                             <1>  ;xor  al, al       ; 02/01/2015
000001BE AA                  <1>  stosb        ; reserved
000001BF B008                <1>  mov   al, 8  ; drive control byte
                             <1>                 ; (do not disable retries,
                             <1>               ; more than 8 heads)
000001C1 AA                  <1>  stosb
000001C2 8B4404              <1>  mov   ax, [si+4]
000001C5 AB                  <1>  stosw        ; physical number of cylinders
                             <1>  ;push ax     ; 02/01/2015
000001C6 8A4408              <1>  mov   al, [si+8]
000001C9 AA                  <1>  stosb        ; physical num. of heads (limit 16)
000001CA 29C0                <1>  sub   ax, ax
                             <1>  ;pop  ax     ; 02/01/2015
000001CC AB                  <1>  stosw        ; landing zone (obsolete)
000001CD 88C8                <1>  mov   al, cl       ; logical sectors per track (limit 63)
000001CF 243F                <1>  and   al, 3Fh
000001D1 AA                  <1>  stosb
                             <1>  ;sub  al, al       ; checksum
                             <1>  ;stosb
                             <1>  ;
000001D2 83C61A              <1>  add   si, 26   ; (BIOS) DPTE address pointer
000001D5 AD                  <1>  lodsw
000001D6 50                  <1>  push  ax     ; (BIOS) DPTE offset
000001D7 AD                  <1>  lodsw
000001D8 50                  <1>  push  ax     ; (BIOS) DPTE segment
                             <1>  ;
                             <1>  ; checksum calculation
000001D9 89FE                <1>  mov   si, di
000001DB 06                  <1>  push  es
000001DC 1F                  <1>  pop   ds
                             <1>  ;mov  cx, 16
000001DD B90F00              <1>  mov   cx, 15
000001E0 29CE                <1>  sub   si, cx
000001E2 30E4                <1>  xor   ah, ah
                             <1>  ;del  cl
                             <1> L11:
000001E4 AC                  <1>  lodsb
000001E5 00C4                <1>  add   ah, al
000001E7 E2FB                <1>  loop  L11
                             <1>  ;
000001E9 88E0                <1>  mov   al, ah
000001EB F6D8                <1>  neg   al    ; -x+x = 0
000001ED AA                  <1>  stosb       ; put checksum in byte 15 of the tbl
                             <1>  ;
000001EE 1F                  <1>  pop   ds    ; (BIOS) DPTE segment
000001EF 5E                  <1>  pop   si    ; (BIOS) DPTE offset
                             <1>  ;
                             <1>  ; 23/02/2015
000001F0 57                  <1>  push  di
                             <1>  ; ES:DI points to DPTE (FDPTE) location
                             <1>  ;mov  cx, 8
000001F1 B108                <1>  mov   cl, 8
000001F3 F3A5                <1>  rep   movsw
                             <1>  ;
                             <1>  ; 23/02/2015
                             <1>  ; (P)ATA drive and LBA validation
                             <1>  ; (invalidating SATA drives and setting
                             <1>  ; CHS type I/O for old type fixed disks)
000001F5 5B                  <1>  pop   bx
000001F6 8CC8                <1>  mov   ax, cs
000001F8 8ED8                <1>  mov   ds, ax
000001FA 268B07              <1>  mov   ax, [es:bx]
```

```
000001FD 3DF001               <1>  cmp   ax, 1F0h
00000200 7418                 <1>  je    short L11a
00000202 3D7001               <1>  cmp   ax, 170h
00000205 7413                 <1>  je    short L11a
                              <1>  ; invalidation
                              <1>  ; (because base port address is not 1F0h or 170h)
00000207 30FF                 <1>  xor   bh, bh
00000209 88D3                 <1>  mov   bl, dl
0000020B 80EB80               <1>  sub   bl, 80h
0000020E C687[DA6D]00         <1>  mov   byte [bx+hd0_type], 0 ; not a valid disk drive !
00000213 808F[266E]F0         <1>         or      byte [bx+drv.status+2], 0F0h ; (failure sign)
00000218 EB14                 <1>  jmp   short L11b
                              <1> L11a:
                              <1>  ; LBA validation
0000021A 268A4704             <1>  mov   al, [es:bx+4] ; Head register upper nibble
0000021E A840                 <1>  test  al, 40h ; LBA bit (bit 6)
00000220 750C                 <1>  jnz   short L11b ; LBA type I/O is OK! (E0h or F0h)
                              <1>  ; force CHS type I/O for this drive (A0h or B0h)
00000222 28FF                 <1>  sub   bh, bh
00000224 88D3                 <1>  mov   bl, dl
00000226 80EB80               <1>  sub   bl, 80h ; 26/02/2015
00000229 80A7[266E]FE         <1>         and     byte [bx+drv.status+2], 0FEh ; clear bit 0
                              <1>                    ; bit 0 = LBA ready bit
                              <1>  ; 'diskio' procedure will check this bit !
                              <1> L11b:
0000022E 3A16[D66D]           <1>  cmp   dl, [last_drv] ; 25/12/2014
00000232 7307                 <1>         jnb     short L13
00000234 E9E9FE               <1>         jmp     L10
                              <1> L12:
                              <1>  ; Restore data registers
00000237 8CC8                 <1>  mov   ax, cs
00000239 8ED8                 <1>  mov   ds, ax
                              <1> L13:
                              <1>  ; 13/12/2014
0000023B 0E                   <1>  push  cs
0000023C 07                   <1>  pop   es
                              <1> L14:
0000023D B411                 <1>  mov   ah, 11h
0000023F CD16                 <1>  int   16h
00000241 7406                 <1>  jz    short L15 ; no keys in keyboard buffer
00000243 B010                 <1>  mov   al, 10h
00000245 CD16                 <1>  int   16h
00000247 EBF4                 <1>  jmp   short L14
                              <1> L15:
                              <1> ; //////
                              <1>  ; 24/11/2014
                              <1>  ; 19/11/2014
                              <1>  ; 14/11/2014
                              <1>  ; Temporary code for disk searching code check
                              <1>  ;
                              <1>  ; This code will show existing (usable) drives and also
                              <1>  ; will show EDD interface support status for hard disks
                              <1>  ; (If status bit 7 is 1, Identify Device info is ready,
                              <1>  ; no need to get it again in protected mode...)
                              <1>  ;
                              <1>  ; 13/11/2014
00000249 BB0700               <1>  mov   bx, 7
0000024C B40E                 <1>  mov   ah, 0Eh
0000024E A0[D86D]             <1>  mov   al, [fd0_type]
00000251 20C0                 <1>  and   al, al
00000253 743D                 <1>  jz    short L15a
00000255 88C2                 <1>  mov   dl, al
00000257 B046                 <1>  mov   al, 'F'
00000259 CD10                 <1>  int   10h
0000025B B044                 <1>  mov   al, 'D'
0000025D CD10                 <1>  int   10h
0000025F B030                 <1>  mov   al, '0'
00000261 CD10                 <1>  int   10h
00000263 B020                 <1>  mov   al, ' '
00000265 CD10                 <1>  int   10h
00000267 E8B300               <1>  call  L15c
0000026A B020                 <1>  mov   al, ' '
0000026C CD10                 <1>  int   10h
                              <1>  ;
0000026E A0[D96D]             <1>  mov   al, [fd1_type]
00000271 20C0                 <1>  and   al, al
00000273 741D                 <1>  jz    short L15a
00000275 88C2                 <1>  mov   dl, al
00000277 B046                 <1>  mov   al, 'F'
00000279 CD10                 <1>  int   10h
0000027B B044                 <1>  mov   al, 'D'
0000027D CD10                 <1>  int   10h
0000027F B031                 <1>  mov   al, '1'
00000281 CD10                 <1>  int   10h
00000283 B020                 <1>  mov   al, ' '
00000285 CD10                 <1>  int   10h
00000287 E89300               <1>  call  L15c
0000028A B020                 <1>  mov   al, ' '
0000028C CD10                 <1>  int   10h
0000028E B020                 <1>  mov   al, ' '
00000290 CD10                 <1>  int   10h
                              <1> L15a:
```

```
00000292 A0[DA6D]            <1>  mov   al, [hd0_type]
00000295 20C0                <1>  and   al, al
00000297 7479                <1>  jz    short L15b
00000299 88C2                <1>  mov   dl, al
0000029B B048                <1>  mov   al, 'H'
0000029D CD10                <1>  int   10h
0000029F B044                <1>  mov   al, 'D'
000002A1 CD10                <1>  int   10h
000002A3 B030                <1>  mov   al, '0'
000002A5 CD10                <1>  int   10h
000002A7 B020                <1>  mov   al, ' '
000002A9 CD10                <1>  int   10h
000002AB E86F00              <1>  call  L15c
000002AE B020                <1>  mov   al, ' '
000002B0 CD10                <1>  int   10h
                             <1>  ;
000002B2 A0[DB6D]            <1>  mov   al, [hd1_type]
000002B5 20C0                <1>  and   al, al
000002B7 7459                <1>  jz    short L15b
000002B9 88C2                <1>  mov   dl, al
000002BB B048                <1>  mov   al, 'H'
000002BD CD10                <1>  int   10h
000002BF B044                <1>  mov   al, 'D'
000002C1 CD10                <1>  int   10h
000002C3 B031                <1>  mov   al, '1'
000002C5 CD10                <1>  int   10h
000002C7 B020                <1>  mov   al, ' '
000002C9 CD10                <1>  int   10h
000002CB E84F00              <1>  call  L15c
000002CE B020                <1>  mov   al, ' '
000002D0 CD10                <1>  int   10h
                             <1>  ;
000002D2 A0[DC6D]            <1>  mov   al, [hd2_type]
000002D5 20C0                <1>  and   al, al
000002D7 7439                <1>  jz    short L15b
000002D9 88C2                <1>  mov   dl, al
000002DB B048                <1>  mov   al, 'H'
000002DD CD10                <1>  int   10h
000002DF B044                <1>  mov   al, 'D'
000002E1 CD10                <1>  int   10h
000002E3 B032                <1>  mov   al, '2'
000002E5 CD10                <1>  int   10h
000002E7 B020                <1>  mov   al, ' '
000002E9 CD10                <1>  int   10h
000002EB E82F00              <1>  call  L15c
000002EE B020                <1>  mov   al, ' '
000002F0 CD10                <1>  int   10h
                             <1>  ;
000002F2 A0[DD6D]            <1>  mov   al, [hd3_type]
000002F5 20C0                <1>  and   al, al
000002F7 7419                <1>  jz    short L15b
000002F9 88C2                <1>  mov   dl, al
000002FB B048                <1>  mov   al, 'H'
000002FD CD10                <1>  int   10h
000002FF B044                <1>  mov   al, 'D'
00000301 CD10                <1>  int   10h
00000303 B033                <1>  mov   al, '3'
00000305 CD10                <1>  int   10h
00000307 B020                <1>  mov   al, ' '
00000309 CD10                <1>  int   10h
0000030B E80F00              <1>  call  L15c
0000030E B020                <1>  mov   al, ' '
00000310 CD10                <1>  int   10h
                             <1>  ;
                             <1> L15b:
00000312 B00D                <1>  mov   al, 0Dh
00000314 CD10                <1>  int   10h
00000316 B00A                <1>  mov   al, 0Ah
00000318 CD10                <1>  int   10h
                             <1>  ;;xor ah, ah
                             <1>  ;;int       16h
                             <1>  ;
0000031A E99A00              <1>         jmp     L16  ; jmp short L16
                             <1>         ;
                             <1> L15c:
0000031D 88D6                <1>  mov   dh, dl
0000031F C0EE04              <1>  shr   dh, 4
00000322 80C630              <1>  add   dh, 30h
00000325 80E20F              <1>  and   dl, 15
00000328 80C230              <1>  add   dl, 30h
0000032B 88F0                <1>  mov   al, dh
0000032D CD10                <1>  int   10h
0000032F 88D0                <1>  mov   al, dl
00000331 CD10                <1>  int   10h
00000333 C3                  <1>  retn
                             <1>  ;
                             <1>  ; end of temporary code for disk searching code check
                             <1>
                             <1> ; //////
                             <1>
                             <1> set_disk_parms:
                             <1>  ; 10/07/2015
                             <1>  ; 14/01/2015
                             <1>  ;push ebx
00000334 6629DB              <1>  sub   ebx, ebx
00000337 8A1E[D56D]          <1>  mov   bl, [drv]
0000033B 80FB80              <1>  cmp   bl, 80h
0000033E 7203                <1>  jb    short sdp0
00000340 80EB7E              <1>  sub   bl, 7Eh
                             <1> sdp0:
00000343 6681C3[246E0000]    <1>  add   ebx, drv.status
0000034A 67C60380            <1>        mov   byte [ebx], 80h ; 'Present' flag
                             <1>  ;
0000034E 88E8                <1>  mov   al, ch ; last cylinder (bits 0-7)
00000350 88CC                <1>  mov   ah, cl ;
00000352 C0EC06              <1>  shr   ah, 6  ; last cylinder (bits 8-9)
00000355 6681EB[246E0000]    <1>  sub   ebx, drv.status
0000035C D0E3                <1>  shl   bl, 1
0000035E 6681C3[DE6D0000]    <1>  add   ebx, drv.cylinders
00000365 40                  <1>  inc   ax  ; convert max. cyl number to cyl count
00000366 678903              <1>  mov   [ebx], ax
00000369 50                  <1>  push  ax ; ** cylinders
0000036A 6681EB[DE6D0000]    <1>  sub   ebx, drv.cylinders
00000371 6681C3[EC6D0000]    <1>  add   ebx, drv.heads
00000378 30E4                <1>  xor   ah, ah
0000037A 88F0                <1>  mov   al, dh ; heads
0000037C 40                  <1>  inc   ax
0000037D 678903              <1>  mov   [ebx], ax
00000380 6681EB[EC6D0000]    <1>         sub     ebx, drv.heads
00000387 6681C3[FA6D0000]    <1>         add     ebx, drv.spt
0000038E 30ED                <1>  xor   ch, ch
00000390 80E13F              <1>  and   cl, 3Fh     ; sectors (bits 0-6)
00000393 67890B              <1>  mov   [ebx], cx
00000396 6681EB[FA6D0000]    <1>         sub     ebx, drv.spt
0000039D 66D1E3              <1>  shl   ebx, 1
000003A0 6681C3[086E0000]    <1>  add   ebx, drv.size ; disk size (in sectors)
                             <1>  ; LBA size = cylinders * heads * secpertrack
000003A7 F7E1                <1>  mul   cx
000003A9 89C2                <1>  mov   dx, ax      ; heads*spt
000003AB 58                  <1>  pop   ax ; ** cylinders
000003AC 48                  <1>  dec   ax ; 1 cylinder reserved (!?)
000003AD F7E2                <1>  mul   dx ; cylinders * (heads*spt)
000003AF 678903              <1>  mov   [ebx], ax
000003B2 67895302            <1>  mov   [ebx+2], dx
                             <1>  ;
                             <1>  ;pop  ebx
000003B6 C3                  <1>  retn
                             <1>
                             <1> ;align 2
                             <1>
                             <1> ;cylinders :  dw 0, 0, 0, 0, 0, 0
                             <1> ;heads   :  dw 0, 0, 0, 0, 0, 0
                             <1> ;spt     :  dw 0, 0, 0, 0, 0, 0
                             <1> ;disk_size :  dd 0, 0, 0, 0, 0, 0
                             <1>
                             <1> ;last_drv:
                             <1> ;db  0
                             <1> ;drv_status:
                             <1> ;db  0,0,0,0,0,0
                             <1> ;db 0
                             <1>
                             <1>
                             <1> ; End Of DISK I/O SYSTEM STRUCTURE INITIALIZATION /// 06/02/2015
                             <1>
                             <1> L16:

                                  ; 10/11/2014
000003B7 FA                             cli   ; Disable interrupts (clear interrupt flag)
                                        ; Reset Interrupt MASK Registers (Master&Slave)
                                  ;mov  al, 0FFh    ; mask off all interrupts
                                  ;out  21h, al           ; on master PIC (8259)
                                  ;jmp  $+2  ; (delay)
                                  ;out  0A1h, al    ; on slave PIC (8259)
                                  ;
                                  ; Disable NMI
000003B8 B080                     mov         al, 80h
000003BA E670                     out         70h, al           ; set bit 7 to 1 for disabling NMI
                                  ;23/02/2015
000003BC 90                       nop               ;
                                  ;in   al, 71h           ; read in 71h just after writing out to 70h
                                                  ; for preventing unknown state (!?)
                                  ;
                                  ; 20/08/2014
                                  ; Moving the kernel 64 KB back (to physical address 0)
                                  ; DS = CS = 1000h
                                  ; 05/11/2014
000003BD 31C0                     xor   ax, ax
000003BF 8EC0                     mov   es, ax ; ES = 0
                                  ;
000003C1 B90040                   mov   cx, (KEND - KLOAD)/4
000003C4 31F6                     xor   si, si
000003C6 31FF                     xor   di, di
000003C8 F366A5                   rep   movsd
                                  ;
```

```
000003CB 06                      push  es ; 0
000003CC 68[D003]                push  L17
000003CF CB                      retf
                                 ;
                                L17:
                                 ; Turn off the floppy drive motor
000003D0 BAF203                         mov     dx, 3F2h
000003D3 EE                             out     dx, al ; 0 ; 31/12/2013

                                 ; Enable access to memory above one megabyte
                                L18:
000003D4 E464                    in    al, 64h
000003D6 A802                    test  al, 2
000003D8 75FA                           jnz     short L18
000003DA B0D1                    mov   al, 0D1h    ; Write output port
000003DC E664                    out   64h, al
                                L19:
000003DE E464                    in    al, 64h
000003E0 A802                    test  al, 2
000003E2 75FA                           jnz     short L19
000003E4 B0DF                    mov   al, 0DFh    ; Enable A20 line
000003E6 E660                    out   60h, al
                                ;L20:
                                 ;
                                 ; Load global descriptor table register

                                        ;mov     ax, cs
                                        ;mov     ds, ax

000003E8 2E0F0116[8068]                 lgdt    [cs:gdtd]

000003EE 0F20C0                         mov     eax, cr0
                                 ; or  eax, 1
000003F1 40                      inc     ax
000003F2 0F22C0                  mov     cr0, eax

                                 ; Jump to 32 bit code

000003F5 66                      db 66h                 ; Prefix for 32-bit
000003F6 EA                      db 0EAh           ; Opcode for far jump
000003F7 [FD030000]              dd StartPM        ; Offset to start, 32-bit
                                                   ; (1000h:StartPM = StartPM + 10000h)
000003FB 0800                    dw KCODE          ; This is the selector for CODE32_DESCRIPTOR,
                                                   ; assuming that StartPM resides in code32

                                [BITS 32]

                                StartPM:
                                 ; Kernel Base Address = 0 ; 30/12/2013
000003FD 66B81000                mov ax, KDATA           ; Save data segment identifier
00000401 8ED8                           mov ds, ax              ; Move a valid data segment into DS register
00000403 8EC0                          mov es, ax              ; Move data segment into ES register
00000405 8EE0                          mov fs, ax              ; Move data segment into FS register
00000407 8EE8                          mov gs, ax              ; Move data segment into GS register
00000409 8ED0                           mov ss, ax              ; Move data segment into SS register
0000040B BC00000900                     mov esp, 90000h         ; Move the stack pointer to 090000h

                                clear_bss: ; Clear uninitialized data area
                                 ; 11/03/2015
00000410 31C0                    xor  eax, eax ; 0
00000412 B9C5040000              mov  ecx, (bss_end - bss_start)/4
                                 ;shr  ecx, 2 ; bss section is already aligned for double words
00000417 BF[406E0000]            mov  edi, bss_start
0000041C F3AB                    rep  stosd

                                memory_init:
                                 ; Initialize memory allocation table and page tables
                                 ; 16/11/2014
                                 ; 15/11/2014
                                 ; 07/11/2014
                                 ; 06/11/2014
                                 ; 05/11/2014
                                 ; 04/11/2014
                                 ; 31/10/2014 (Retro UNIX 386 v1 - Beginning)
                                 ;
                                ;xor   eax, eax
                                ;xor   ecx, ecx
0000041E B108                    mov   cl, 8
00000420 BF00001000              mov   edi, MEM_ALLOC_TBL
00000425 F3AB                    rep   stosd          ; clear Memory Allocation Table
                                                      ; for the first 1 MB memory
                                 ;
00000427 668B0D[D06D0000]        mov   cx, [mem_1m_1k]      ; Number of contiguous KB between
                                                      ; 1 and 16 MB, max. 3C00h = 15 MB.
0000042E 66C1E902                shr   cx, 2          ; convert 1 KB count to 4 KB count
00000432 890D[B0700000]          mov   [free_pages], ecx
00000438 668B15[D26D0000]        mov   dx, [mem_16m_64k]  ; Number of contiguous 64 KB blocks
                                                      ; between 16 MB and 4 GB.
0000043F 6609D2                  or    dx, dx
00000442 7413                    jz    short mi_0
                                 ;
00000444 6689D0                  mov   ax, dx
```

```
00000447 C1E004                   shl   eax, 4               ; 64 KB -> 4 KB (page count)
0000044A 0105[B0700000]           add   [free_pages], eax
00000450 0500100000               add   eax, 4096      ; 16 MB = 4096 pages
00000455 EB07                     jmp   short mi_1
                                 mi_0:
00000457 6689C8                   mov   ax, cx
0000045A 66050001                 add   ax, 256              ; add 256 pages for the first 1 MB
                                 mi_1:
0000045E A3[AC700000]             mov   [memory_size], eax ; Total available memory in pages
                                                       ; 1 alloc. tbl. bit = 1 memory page
                                                       ; 32 allocation bits = 32 mem. pages
                                  ;
00000463 05FF7F0000               add   eax, 32767     ; 32768 memory pages per 1 M.A.T. page
00000468 C1E80F                   shr   eax, 15              ; ((32768 * x) + y) pages (y < 32768)
                                                       ;  --> x + 1 M.A.T. pages, if y > 0
                                                       ;  --> x M.A.T. pages, if y = 0
0000046B 66A3[C0700000]           mov   [mat_size], ax        ; Memory Alloc. Table Size in pages
00000471 C1E00C                   shl   eax, 12              ; 1 M.A.T. page = 4096 bytes
                                  ;                    ; Max. 32 M.A.T. pages (4 GB memory)
00000474 89C3                     mov   ebx, eax       ; M.A.T. size in bytes
                                  ; Set/Calculate Kernel's Page Directory Address
00000476 81C300001000             add   ebx, MEM_ALLOC_TBL
0000047C 891D[A8700000]           mov   [k_page_dir], ebx  ; Kernel's Page Directory address
                                                       ; just after the last M.A.T. page
                                  ;
00000482 83E804                   sub   eax, 4               ; convert M.A.T. size to offset value
00000485 A3[B8700000]             mov   [last_page], eax   ; last page ofset in the M.A.T.
                                  ;                    ; (allocation status search must be
                                                       ; stopped after here)
0000048A 31C0                     xor   eax, eax
0000048C 48                       dec   eax            ; FFFFFFFFh (set all bits to 1)
0000048D 6651                     push  cx
0000048F C1E905                   shr   ecx, 5                ; convert 1 - 16 MB page count to
                                                       ; count of 32 allocation bits
00000492 F3AB                     rep   stosd
00000494 6659                     pop   cx
00000496 40                       inc   eax            ; 0
00000497 80E11F                   and   cl, 31               ; remain bits
0000049A 7412                     jz    short mi_4
0000049C 8907                     mov   [edi], eax     ; reset
                                 mi_2:
0000049E 0FAB07                   bts   [edi], eax     ; 06/11/2014
000004A1 FEC9                     dec   cl
000004A3 7404                     jz    short mi_3
000004A5 FEC0                     inc   al
000004A7 EBF5                     jmp   short mi_2
                                 mi_3:
000004A9 28C0                     sub   al, al               ; 0
000004AB 83C704                   add   edi, 4               ; 15/11/2014
                                 mi_4:
000004AE 6609D2                   or    dx, dx              ; check 16M to 4G memory space
000004B1 7421                     jz    short mi_6    ; max. 16 MB memory, no more...
                                  ;
000004B3 B900021000               mov   ecx, MEM_ALLOC_TBL + 512 ; End of first 16 MB memory
                                  ;
000004B8 29F9                     sub   ecx, edi      ; displacement (to end of 16 MB)
000004BA 7406                     jz    short mi_5    ; jump if EDI points to
                                                      ;         end of first 16 MB
000004BC D1E9                     shr   ecx, 1              ; convert to dword count
000004BE D1E9                     shr   ecx, 1              ; (shift 2 bits right)
000004C0 F3AB                     rep   stosd         ; reset all bits for reserved pages
                                                      ; (memory hole under 16 MB)
                                 mi_5:
000004C2 6689D1                   mov   cx, dx              ; count of 64 KB memory blocks
000004C5 D1E9                     shr   ecx, 1              ; 1 alloc. dword per 128 KB memory
000004C7 9C                     pushf                 ; 16/11/2014
000004C8 48                     dec     eax           ; FFFFFFFFh (set all bits to 1)
000004C9 F3AB                   rep     stosd
000004CB 40                     inc     eax           ; 0
000004CC 9D                     popf                  ; 16/11/2014
000004CD 7305                   jnc     short mi_6
000004CF 6648                   dec     ax            ; eax = 0000FFFFh
000004D1 AB                     stosd
000004D2 6640                   inc     ax            ; 0
                                 mi_6:
000004D4 39DF                     cmp   edi, ebx      ; check if EDI points to
000004D6 730A                     jnb   short mi_7    ; end of memory allocation table
                                  ;                   ; (>= MEM_ALLOC_TBL + 4906)
000004D8 89D9                     mov   ecx, ebx      ; end of memory allocation table
000004DA 29F9                     sub   ecx, edi      ; convert displacement/offset
000004DC D1E9                     shr   ecx, 1              ; to dword count
000004DE D1E9                     shr   ecx, 1              ; (shift 2 bits right)
000004E0 F3AB                     rep   stosd         ; reset all remain M.A.T. bits
                                 mi_7:
                                  ; Reset M.A.T. bits in M.A.T. (allocate M.A.T. pages)
000004E2 BA00001000               mov   edx, MEM_ALLOC_TBL
                                  ;sub  ebx, edx      ; Mem. Alloc. Tbl. size in bytes
                                  ;shr  ebx, 12             ; Mem. Alloc. Tbl. size in pages
000004E7 668B0D[C0700000]         mov   cx, [mat_size]      ; Mem. Alloc. Tbl. size in pages
000004EE 89D7                     mov   edi, edx
000004F0 C1EF0F                   shr   edi, 15             ; convert M.A.T. address to
                                                      ; byte offset in M.A.T.
```

```
                                                     ; (1 M.A.T. byte points to
                                                     ;         32768 bytes)
                                                     ; Note: MEM_ALLOC_TBL address
                                                     ; must be aligned on 128 KB
                                                     ; boundary!
000004F3 01D7                    add   edi, edx      ; points to M.A.T.'s itself
                                 ; eax = 0
000004F5 290D[B0700000]          sub   [free_pages], ecx ; 07/11/2014
                                mi_8:
000004FB 0FB307                  btr   [edi], eax    ; clear bit 0 to bit x (1 to 31)
                                 ;dec  bl
000004FE FEC9                    dec   cl
00000500 7404                    jz    short mi_9
00000502 FEC0                    inc   al
00000504 EBF5                    jmp   short mi_8
                                mi_9:
                                 ;
                                 ; Reset Kernel's Page Dir. and Page Table bits in M.A.T.
                                 ;           (allocate pages for system page tables)

                                 ; edx = MEM_ALLOC_TBL
00000506 8B0D[AC700000]          mov   ecx, [memory_size] ; memory size in pages (PTEs)
0000050C 81C1FF030000            add   ecx, 1023    ; round up (1024 PTEs per table)
00000512 C1E90A                  shr   ecx, 10            ; convert memory page count to
                                                    ; page table count (PDE count)
                                 ;
00000515 51                      push  ecx          ; (**) PDE count (<= 1024)
                                 ;
00000516 41                      inc   ecx          ; +1 for kernel page directory
                                 ;
00000517 290D[B0700000]          sub   [free_pages], ecx ; 07/11/2014
                                 ;
0000051D 8B35[A8700000]          mov   esi, [k_page_dir] ; Kernel's Page Directory address
00000523 C1EE0C                  shr   esi, 12            ; convert to page number
                                mi_10:
00000526 89F0                    mov   eax, esi     ; allocation bit offset
00000528 89C3                    mov   ebx, eax
0000052A C1EB03                  shr   ebx, 3             ; convert to alloc. byte offset
0000052D 80E3FC                  and   bl,  0FCh    ; clear bit 0 and bit 1
                                                    ;   to align on dword boundary
00000530 83E01F                  and   eax, 31            ; set allocation bit position
                                                    ;  (bit 0 to bit 31)
                                 ;
00000533 01D3                    add   ebx, edx     ; offset in M.A.T. + M.A.T. address
                                 ;
00000535 0FB303                  btr   [ebx], eax   ; reset relevant bit (0 to 31)
                                 ;
00000538 46                      inc   esi          ; next page table
00000539 E2EB                    loop  mi_10        ; allocate next kernel page table
                                                    ; (ecx = page table count + 1)
                                 ;
0000053B 59                      pop   ecx          ; (**) PDE count (= pg. tbl. count)
                                 ;
                                 ; Initialize Kernel Page Directory and Kernel Page Tables
                                 ;
                                 ; Initialize Kernel's Page Directory
0000053C 8B3D[A8700000]          mov   edi, [k_page_dir]
00000542 89F8                    mov   eax, edi
00000544 0C03                    or    al, PDE_A_PRESENT + PDE_A_WRITE
                                                   ; supervisor + read&write + present
00000546 89CA                    mov   edx, ecx    ; (**) PDE count (= pg. tbl. count)
                                mi_11:
00000548 0500100000              add   eax, 4096   ; Add page size (PGSZ)
                                                     ; EAX points to next page table
0000054D AB                      stosd
0000054E E2F8                    loop  mi_11
00000550 29C0                    sub   eax, eax    ; Empty PDE
00000552 66B90004                mov   cx, 1024    ; Entry count (PGSZ/4)
00000556 29D1                    sub   ecx, edx
00000558 7402                    jz    short mi_12
0000055A F3AB                    rep   stosd             ; clear remain (empty) PDEs
                                 ;
                                 ; Initialization of Kernel's Page Directory is OK, here.
                                mi_12:
                                 ; Initialize Kernel's Page Tables
                                 ;
                                 ; (EDI points to address of page table 0)
                                 ; eax = 0
0000055C 8B0D[AC700000]          mov   ecx, [memory_size] ; memory size in pages
00000562 89CA                    mov   edx, ecx    ; (***)
00000564 B003                    mov   al, PTE_A_PRESENT + PTE_A_WRITE
                                                  ; supervisor + read&write + present
                                mi_13:
00000566 AB                      stosd
00000567 0500100000              add   eax, 4096
0000056C E2F8                    loop  mi_13
0000056E 6681E2FF03              and   dx, 1023    ; (***)
00000573 740B                    jz    short mi_14
00000575 66B90004                mov   cx, 1024
00000579 6629D1                  sub   cx, dx           ; from dx (<= 1023) to 1024
0000057C 31C0                    xor   eax, eax
0000057E F3AB                    rep   stosd       ; clear remain (empty) PTEs
```

```
                                                   ; of the last page table
                                 mi_14:
                                  ;  Initialization of Kernel's Page Tables is OK, here.
                                  ;
00000580 89F8                     mov   eax, edi    ; end of the last page table page
                                                      ; (beginging of user space pages)
00000582 C1E80F                   shr   eax, 15           ; convert to M.A.T. byte offset
00000585 24FC                     and   al, 0FCh    ; clear bit 0 and bit 1 for
                                                    ; aligning on dword boundary

00000587 A3[BC700000]             mov   [first_page], eax
0000058C A3[B4700000]             mov   [next_page], eax ; The first free page pointer
                                                     ; for user programs
                                                     ; (Offset in Mem. Alloc. Tbl.)
                                  ;
                                  ; Linear/FLAT (1 to 1) memory paging for the kernel is OK, here.
                                  ;

                                  ; Enable paging
                                  ;
00000591 A1[A8700000]                    mov     eax, [k_page_dir]
00000596 0F22D8                   mov   cr3, eax
00000599 0F20C0                   mov   eax, cr0
0000059C 0D00000080               or    eax, 80000000h    ; set paging bit (bit 31)
000005A1 0F22C0                   mov   cr0, eax
                                         ;jmp    KCODE:StartPMP

000005A4 EA                       db 0EAh           ; Opcode for far jump
000005A5 [AB050000]                      dd StartPMP           ; 32 bit offset
000005A9 0800                     dw KCODE          ; kernel code segment descriptor


                                 StartPMP:
                                  ; 06/11//2014
                                  ; Clear video page 0
                                  ;
                                  ; Temporary Code
                                  ;
000005AB B9E8030000               mov   ecx, 80*25/2
000005B0 BF00800B00               mov   edi, 0B8000h
000005B5 31C0                     xor   eax, eax    ; black background, black fore color
000005B7 F3AB                     rep   stosd

                                  ; 19/08/2014
                                  ; Kernel Base Address = 0
                                  ; It is mapped to (physically) 0 in the page table.
                                  ; So, here is exactly 'StartPMP' address.
                                  ;
                                  ;;mov ah, 4Eh       ; Red background, yellow forecolor
                                  ;;mov esi, msgPM
                                  ;; 14/08/2015 (kernel version message will appear
                                  ;;           when protected mode and paging is enabled)
000005B9 B40B                     mov   ah, 0Bh ; Black background, light cyan forecolor
000005BB BE[616B0000]             mov   esi, msgKVER
000005C0 BF00800B00               mov   edi, 0B8000h ; 27/08/2014
                                  ; 20/08/2014
000005C5 E896010000               call  printk

                                  ; 'UNIX v7/x86' source code by Robert Nordier (1999)
                                  ; // Set IRQ offsets
                                  ;
                                  ;  Linux (v0.12) source code by Linus Torvalds (1991)
                                  ;
                                                         ;; ICW1
000005CA B011                     mov   al, 11h                ; Initialization sequence
000005CC E620                     out   20h, al                ;     8259A-1
                                  ; jmp       $+2
000005CE E6A0                     out   0A0h, al         ;     8259A-2
                                                         ;; ICW2
000005D0 B020                     mov   al, 20h                ; Start of hardware ints (20h)
000005D2 E621                     out   21h, al                ;     for 8259A-1
                                  ; jmp       $+2
000005D4 B028                     mov   al, 28h                ; Start of hardware ints (28h)
000005D6 E6A1                     out   0A1h, al         ;     for 8259A-2
                                                         ;
000005D8 B004                     mov   al, 04h                ;; ICW3
000005DA E621                     out   21h, al                ;     IRQ2 of 8259A-1 (master)
                                  ; jmp       $+2
000005DC B002                     mov   al, 02h                ;     is 8259A-2 (slave)
000005DE E6A1                     out   0A1h, al         ;
                                                         ;; ICW4
000005E0 B001                     mov   al, 01h                ;
000005E2 E621                     out   21h, al                ;     8086 mode, normal EOI
                                  ; jmp       $+2
000005E4 E6A1                     out   0A1h, al         ;     for both chips.

                                  ;mov  al, 0FFh    ; mask off all interrupts for now
                                  ;out  21h, al
                                  ;; jmp      $+2
                                  ;out  0A1h, al

                                  ; 02/04/2015
                                  ; 26/03/2015 System call (INT 30h) modification
                                  ;  DPL = 3 (Interrupt service routine can be called from user mode)

                                  ;
                                  ;; Linux (v0.12) source code by Linus Torvalds (1991)
                                  ;  setup_idt:
                                  ;
                                         ;; 16/02/2015
                                  ;;mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
                                  ; 21/08/2014 (timer_int)
000005E6 BE[8C680000]             mov   esi, ilist
000005EB 8D3D[406E0000]           lea   edi, [idt]
                                  ; 26/03/2015
000005F1 B930000000               mov   ecx, 48           ; 48 hardware interrupts (INT 0 to INT 2Fh)
                                  ; 02/04/2015
000005F6 BB00000800               mov   ebx,  80000h
                                 rp_sidt1:
000005FB AD                       lodsd
000005FC 89C2                     mov   edx, eax
000005FE 66BA008E                 mov   dx, 8E00h
00000602 6689C3                   mov   bx, ax
00000605 89D8                     mov   eax, ebx    ; /* selector = 0x0008 = cs */
                                                          ; /* interrupt gate - dpl=0, present */
00000607 AB                       stosd ; selector & offset bits 0-15
00000608 89D0                     mov   eax, edx
0000060A AB                       stosd ; attributes & offset bits 16-23
0000060B E2EE                     loop  rp_sidt1
0000060D B110                     mov   cl, 16        ; 16 software interrupts (INT 30h to INT 3Fh)
                                 rp_sidt2:
0000060F AD                       lodsd
00000610 21C0                     and   eax, eax
00000612 7413                     jz    short rp_sidt3
00000614 89C2                     mov   edx, eax
00000616 66BA00EE                 mov   dx, 0EE00h  ; P=1b/DPL=11b/01110b
0000061A 6689C3                   mov   bx, ax
0000061D 89D8                     mov   eax, ebx    ; selector & offset bits 0-15
0000061F AB                       stosd
00000620 89D0                     mov   eax, edx
00000622 AB                       stosd
00000623 E2EA                     loop  rp_sidt2
00000625 EB16                     jmp   short sidt_OK
                                 rp_sidt3:
00000627 B8[8A0A0000]             mov   eax, ignore_int
0000062C 89C2                     mov   edx, eax
0000062E 66BA00EE                 mov   dx, 0EE00h  ; P=1b/DPL=11b/01110b
00000632 6689C3                   mov   bx, ax
00000635 89D8                     mov   eax, ebx    ; selector & offset bits 0-15
                                 rp_sidt4:
00000637 AB                       stosd
00000638 92                       xchg  eax, edx
00000639 AB                       stosd
0000063A 92                       xchg  edx, eax
0000063B E2FA                     loop  rp_sidt4
                                 sidt_OK:
0000063D 0F011D[86680000]         lidt  [idtd]
                                  ;
                                  ; TSS descriptor setup ; 24/03/2015
00000644 B8[40700000]             mov   eax, task_state_segment
00000649 66A3[7A680000]           mov   [gdt_tss0], ax
0000064F C1C010                   rol   eax, 16
00000652 A2[7C680000]             mov   [gdt_tss1], al
00000657 8825[7F680000]           mov   [gdt_tss2], ah
0000065D 66C705[A6700000]68-      mov   word [tss.IOPB], tss_end - task_state_segment
00000665 00
                                        ;
                                        ; IO Map Base address (When this address points
                                        ; to end of the TSS, CPU does not use IO port
                                        ; permission bit map for RING 3 IO permissions,
                                        ; access to any IO ports in ring 3 will be forbidden.)
                                        ;
                                  ;mov  [tss.esp0], esp ; TSS offset 4
                                  ;mov  word [tss.ss0], KDATA ; TSS offset 8 (SS)
00000666 66B82800                       mov   ax, TSS  ; It is needed when an interrupt
                                               ; occurs (or a system call -software INT- is requested)
                                               ; while cpu running in ring 3 (in user mode).
                                               ; (Kernel stack pointer and segment will be loaded
                                               ; from offset 4 and 8 of the TSS, by the CPU.)
0000066A 0F00D8                   ltr   ax  ; Load task register
                                  ;
                                 esp0_set0:
                                  ; 30/07/2015
0000066D 8B0D[AC700000]           mov   ecx, [memory_size] ; memory size in pages
00000673 C1E10C                   shl   ecx, 12 ; convert page count to byte count
00000676 81F900004000             cmp   ecx, CORE ; beginning of user's memory space (400000h)
                                                ; (kernel mode virtual address)
0000067C 7605                     jna   short esp0_set1
                                  ;
                                  ; If available memory > CORE (end of the 1st 4 MB)
                                  ; set stack pointer to CORE
                                  ;(Because, PDE 0 is reserved for kernel space in user's page directory)
                                  ;(PDE 0 points to page table of the 1st 4 MB virtual address space)
0000067E B900004000               mov   ecx, CORE
```

```
                                 esp0_set1:
00000683 89CC                     mov   esp, ecx ; top of kernel stack (**tss.esp0**)
                                 esp0_set_ok:
                                  ; 30/07/2015 (**tss.esp0**)
00000685 8925[44700000]           mov   [tss.esp0], esp
0000068B 66C705[48700000]10-             mov     word [tss.ss0], KDATA
00000693 00
                                  ; 14/08/2015
                                  ; 10/11/2014 (Retro UNIX 386 v1 - Erdogan Tan)
                                  ;
                                  ;cli  ; Disable interrupts (for CPU)
                                  ;    (CPU will not handle hardware interrupts, except NMI!)
                                  ;
00000694 30C0                     xor   al, al            ; Enable all hardware interrupts!
00000696 E621                     out   21h, al           ; (IBM PC-AT compatibility)
00000698 EB00                     jmp   $+2         ; (All conventional PC-AT hardware
0000069A E6A1                     out   0A1h, al    ;  interrupts will be in use.)
                                                    ; (Even if related hardware component
                                                    ;  does not exist!)
                                  ; Enable NMI
0000069C B07F                     mov   al, 7Fh           ; Clear bit 7 to enable NMI (again)
0000069E E670                     out   70h, al
                                  ; 23/02/2015
000006A0 90                       nop
000006A1 E471                     in    al, 71h           ; read in 71h just after writing out to 70h
                                                    ; for preventing unknown state (!?)
                                  ;
                                  ; Only a NMI can occur here... (Before a 'STI' instruction)
                                  ;
                                  ; 02/09/2014
000006A3 6631DB                   xor   bx, bx
000006A6 66BA0002                 mov   dx, 0200h   ; Row 2, column 0  ; 07/03/2015
000006AA E8920F0000               call  set_cpos
                                  ;
                                  ; 06/11/2014
                                  ; Temporary Code
                                  ;
000006AF E8C1110000               call  memory_info
                                  ; 14/08/2015
                                  ;call getch ; 28/02/2015
                                 drv_init:
000006B4 FB                       sti   ; Enable Interrupts
                                  ; 06/02/2015
000006B5 8B15[DA6D0000]           mov   edx, [hd0_type] ; hd0, hd1, hd2, hd3
000006BB 668B1D[D86D0000]         mov   bx, [fd0_type] ; fd0, fd1
                                  ; 22/02/2015
000006C2 6621DB                   and   bx, bx
000006C5 751B                     jnz   short di1
                                  ;
000006C7 09D2                     or    edx, edx
000006C9 7529                     jnz   short di2
                                  ;
                                 setup_error:
000006CB BE[856C0000]             mov   esi, setup_error_msg
                                 psem:
000006D0 AC                       lodsb
000006D1 08C0                     or    al, al
                                  ;jz   short haltx ; 22/02/2015
000006D3 7426                     jz    short di3
000006D5 56                       push  esi
000006D6 31DB                     xor   ebx, ebx ; 0
                                              ; Video page 0 (bl=0)
000006D8 B407                     mov   ah, 07h ; Black background,
                                              ; light gray forecolor
000006DA E83E0E0000               call  write_tty
000006DF 5E                       pop   esi
000006E0 EBEE                     jmp   short psem

                                 di1:
                                  ; supress 'jmp short T6'
                                  ;  (activate fdc motor control code)
000006E2 66C705[E1070000]90-      mov   word [T5], 9090h ; nop
000006EA 90
                                  ;
                                  ;mov  ax, int_0Eh ; IRQ 6 handler
                                  ;mov  di, 0Eh*4   ; IRQ 6 vector
                                  ;stosw
                                  ;mov  ax, cs
                                  ;stosw
                                  ;; 16/02/2015
                                         ;;mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
                                  ;
000006EB E8D6200000               CALL  DSKETTE_SETUP     ; Initialize Floppy Disks
                                  ;
000006F0 09D2                     or    edx, edx
000006F2 7407                            jz      short di3
                                 di2:
000006F4 E813210000               call        DISK_SETUP  ; Initialize Fixed Disks
000006F9 72D0                            jc      short setup_error
                                 di3:
000006FB E849110000               call  setup_rtc_int     ; 22/05/2015 (dsectrpm.s)
                                  ;
```

```
00000700 E8A2600000              call  display_disks ; 07/03/2015  (Temporary)
                                ;haltx:
                                 ; 14/08/2015
                                 ;call getch ; 22/02/2015
00000705 FB                      sti   ; Enable interrupts (for CPU)
                                 ; 14/08/2015
00000706 B9FFFFFF0F              mov   ecx, 0FFFFFFFh
                                md_info_msg_wait:
0000070B 51                      push  ecx
0000070C B001                    mov   al, 1
0000070E 8A25[D6700000]          mov   ah, [ptty] ; active (current) video page
00000714 E8BF5D0000              call  getc_n
00000719 59                      pop   ecx
0000071A 7502                    jnz   short md_info_msg_ok
0000071C E2ED                    loop  md_info_msg_wait
                                md_info_msg_ok:
                                 ; 30/06/2015
0000071E E800310000              call  sys_init
                                 ;
                                 ;jmp  cpu_reset ; 22/02/2015
                                hang:
                                 ; 23/02/2015
                                 ;sti              ; Enable interrupts
00000723 F4                      hlt
                                 ;
                                 ;nop
                                 ;; 03/12/2014
                                 ;; 28/08/2014
                                 ;mov  ah, 11h
                                 ;call getc
                                 ;jz      _c8
                                 ;
                                 ; 23/02/2015
                                 ; 06/02/2015
                                 ; 07/09/2014
00000724 31DB                    xor   ebx, ebx
00000726 8A1D[D6700000]          mov   bl, [ptty]  ; active_page
0000072C 89DE                    mov   esi, ebx
0000072E 66D1E6                  shl   si, 1
00000731 81C6[D8700000]          add   esi, ttychr
00000737 668B06                  mov   ax, [esi]
0000073A 6621C0                  and   ax, ax
                                 ;jz   short _c8
0000073D 74E4                    jz    short hang
0000073F 66C7060000              mov   word [esi], 0
00000744 80FB03                  cmp   bl, 3       ; Video page 3
                                 ;jb   short _c8
00000747 72DA                    jb    short hang
                                 ;
                                 ; 02/09/2014
00000749 B40E                    mov   ah, 0Eh           ; Yellow character
                                                   ; on black background
                                 ; 07/09/2014
                                nxtl:
0000074B 6653                    push  bx
                                 ;
                                 ;xor  bx, bx            ; bl = 0 (video page 0)
                                                   ; bh = 0 (video mode)
                                                   ; Retro UNIX 386 v1 - Video Mode 0
                                                   ; (PC/AT Video Mode 3 - 80x25 Alpha.)
0000074D 6650                    push  ax
0000074F E8C90D0000              call  write_tty
00000754 6658                    pop   ax
00000756 665B                    pop   bx ; 07/09/2014
00000758 3C0D                    cmp   al, 0Dh           ; carriage return (enter)
                                 ;jne  short _c8
0000075A 75C7                    jne   short hang
0000075C B00A                    mov   al, 0Ah           ; next line
0000075E EBEB                    jmp   short nxtl

                                ;_c8:
                                ;; 25/08/2014
                                ;cli                     ; Disable interrupts
                                ;mov   al, [scounter + 1]
                                ;and   al, al
                                ;jnz   hang
                                ;call  rtc_p
                                ;jmp     hang


                                 ; 27/08/2014
                                 ; 20/08/2014
                                printk:
                                        ;mov    edi, [scr_row]
                                pkl:
00000760 AC                      lodsb
00000761 08C0                    or    al, al
00000763 7404                    jz    short pkr
00000765 66AB                    stosw
00000767 EBF7                    jmp   short pkl
                                pkr:
00000769 C3                      retn
```

```
                                ; 25/07/2015
                                ; 14/05/2015 (multi tasking -time sharing- 'clock', x_timer)
                                ; 17/02/2015
                                ; 06/02/2015 (unix386.s)
                                ; 11/12/2014 - 22/12/2014 (dsectrm2.s)
                                ;
                                ; IBM PC-XT Model 286 Source Code - BIOS2.ASM (06/10/85)
                                ;
                                ;-- HARDWARE INT  08 H - ( IRQ LEVEL 0 ) --------------------------------------
                                ;THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF        :
                                ;THE 8254 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR        :
                                ;IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND.     :
                                ;                                                                :
                                ;THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE    :
                                ;POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.       :
                                ;THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40)  :
                                ;OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE          :
                                ;DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS.            :
                                ;THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH          :
                                ;INTERRUPT 1CH AT EVERY TIME TICK.  THE USER MUST CODE A          :
                                ;ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.       :
                                ;-----------------------------------------------------------------------------
                                ;

                                timer_int:   ; IRQ 0
                                ;int_08h:    ; Timer
                                 ; 14/10/2015
                                 ; Here, we are simulating system call entry (for task switch)
                                 ; (If multitasking is enabled,
                                 ; 'clock' procedure may jump to 'sysrelease')
0000076A 1E                      push  ds
0000076B 06                      push  es
0000076C 0FA0                    push  fs
0000076E 0FA8                    push  gs
00000770 60                      pushad  ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
00000771 66B91000                mov     cx, KDATA
00000775 8ED9                           mov     ds, cx
00000777 8EC1                           mov     es, cx
00000779 8EE1                           mov     fs, cx
0000077B 8EE9                           mov     gs, cx
                                 ;
0000077D 0F20D9                  mov   ecx, cr3
00000780 890D[1F080000]          mov   [cr3reg], ecx ; save current cr3 register value/content
                                 ;
00000786 3B0D[A8700000]          cmp   ecx, [k_page_dir]
0000078C 741F                    je    short T3
                                 ;
                                 ; timer interrupt has been occurred while OS is in user mode
0000078E A3[80740000]            mov   [u.r0], eax
00000793 89E1                    mov   ecx, esp
00000795 83C130                  add   ecx, ESPACE ; 4 * 12 (stack frame)
00000798 890D[78740000]          mov   [u.sp], ecx ; kernel stack pointer at the start of interrupt
0000079E 8925[7C740000]          mov   [u.usp], esp ; kernel stack points to user's registers
                                 ;
000007A4 8B0D[A8700000]          mov   ecx, [k_page_dir]
000007AA 0F22D9                  mov   cr3, ecx
                                T3:
000007AD FB                      sti                     ; INTERRUPTS BACK ON
000007AE 66FF05[24710000]        INC   word [TIMER_LOW]  ; INCREMENT TIME
000007B5 7507                    JNZ   short T4          ; GO TO TEST_DAY
000007B7 66FF05[26710000]        INC   word [TIMER_HIGH] ; INCREMENT HIGH WORD OF TIME
                                T4:                               ; TEST_DAY
000007BE 66833D[26710000]18      CMP   word [TIMER_HIGH],018H  ; TEST FOR COUNT EQUALING 24 HOURS
000007C6 7519                    JNZ   short T5          ; GO TO DISKETTE_CTL
000007C8 66813D[24710000]B0-     CMP   word [TIMER_LOW],0B0H
000007D0 00
000007D1 750E                    JNZ   short T5          ; GO TO DISKETTE_CTL

                                ;----- TIMER HAS GONE 24 HOURS
                                 ;;SUB AX,AX
                                 ;MOV  [TIMER_HIGH],AX
                                 ;MOV  [TIMER_LOW],AX
000007D3 29C0                    sub   eax, eax
000007D5 A3[24710000]            mov   [TIMER_LH], eax
                                 ;
000007DA C605[28710000]01        MOV   byte [TIMER_OFL],1

                                ;----- TEST FOR DISKETTE TIME OUT

                                T5:
                                 ; 23/12/2014
000007E1 EB1D                    jmp   short T6          ; will be replaced with nop, nop
                                                         ; (9090h) if a floppy disk
                                                         ; is detected.
                                 ;mov  al,[CS:MOTOR_COUNT]
000007E3 A0[2B710000]            mov   al, [MOTOR_COUNT]
000007E8 FEC8                    dec   al
                                 ;mov  [CS:MOTOR_COUNT], al    ; DECREMENT DISKETTE MOTOR CONTROL
000007EA A2[2B710000]            mov   [MOTOR_COUNT], al
                                 ;mov  [ORG_MOTOR_COUNT], al
000007EF 750F                    JNZ   short T6          ; RETURN IF COUNT NOT OUT
```

```
000007F1 B0F0                    mov   al,0F0h
                                 ;AND  [CS:MOTOR_STATUS],al    ; TURN OFF MOTOR RUNNING BITS
000007F3 2005[2A710000]          and   [MOTOR_STATUS], al
                                 ;and  [ORG_MOTOR_STATUS], al
000007F9 B00C                    MOV   AL,0CH                  ; bit 3 = enable IRQ & DMA,
                                                         ; bit 2 = enable controller
                                                         ;     1 = normal operation
                                                         ;     0 = reset
                                                         ; bit 0, 1 = drive select
                                                         ; bit 4-7 = motor running bits
000007FB 66BAF203                MOV   DX,03F2H          ; FDC CTL PORT
000007FF EE                      OUT   DX,AL             ; TURN OFF THE MOTOR
                                T6:
                                 ;inc  word [CS:wait_count]    ; 22/12/2014 (byte -> word)
                                                         ; TIMER TICK INTERRUPT
                                 ;;inc word [wait_count] ;;27/02/2015
                                 ;INT  1CH               ; TRANSFER CONTROL TO A USER ROUTINE
                                 ;;;;cli
                                 ;call       u_timer                 ; TRANSFER CONTROL TO A USER ROUTINE
00000800 FF15[1B080000]          call  [x_timer] ; 14/05/2015
                                T7:
                                 ; 14/10/2015
00000806 B020                    MOV   AL,EOI                  ; GET END OF INTERRUPT MASK
00000808 FA                      CLI                     ; DISABLE INTERRUPTS TILL STACK CLEARED
00000809 E620                    OUT   INTA00,AL         ; END OF INTERRUPT TO 8259 - 1
                                 ;
0000080B A1[1F080000]            mov   eax, [cr3reg]           ; previous value/content of cr3 register
00000810 0F22D8                  mov   cr3, eax  ; restore cr3 register content
                                 ;
00000813 61                      popad ; edi, esi, ebp, temp (icrement esp by 4), ebx, edx, ecx, eax
                                 ;
00000814 0FA9                    pop   gs
00000816 0FA1                    pop   fs
00000818 07                      pop   es
00000819 1F                      pop   ds
0000081A CF                      iretd ; return from interrupt


                                ; ////////////////

                                ; 14/05/2015 - Multi tasking 'clock' procedure (sys emt)
                                x_timer:
0000081B [23080000]              dd    u_timer                 ; 14/05/2015
                                 ;dd   clock

                                ; 14/10/2015
0000081F 00000000               cr3reg: dd 0

                                 ; 06/02/2015
                                 ; 07/09/2014
                                 ; 21/08/2014
                                u_timer:
                                ;timer_int:  ; IRQ 0
                                 ; 06/02/2015
                                 ;push eax
                                 ;push edx
                                 ;push ecx
                                 ;push ebx
                                 ;push ds
                                 ;push es
                                 ;mov  eax, KDATA
                                 ;mov  ds, ax
                                 ;mov  es, ax
00000823 FF05[EC700000]          inc   dword [tcount]
00000829 BB[D26B0000]            mov   ebx, tcountstr + 4
0000082E 66A1[EC700000]          mov   ax, [tcount]
00000834 B90A000000              mov   ecx, 10
                                rp_divtcnt:
00000839 31D2                    xor   edx, edx
0000083B F7F1                    div   ecx
0000083D 80C230                  add   dl, 30h
00000840 8813                    mov   [ebx], dl
00000842 6609C0                  or    ax, ax
00000845 7403                    jz    short print_lzero
00000847 4B                      dec   ebx
00000848 EBEF                    jmp   short rp_divtcnt
                                print_lzero:
0000084A 81FB[CE6B0000]          cmp   ebx, tcountstr
00000850 7606                    jna   short print_tcount
00000852 4B                      dec   ebx
00000853 C60330                  mov   byte [ebx], 30h
00000856 EBF2                    jmp   short print_lzero
                                print_tcount:
00000858 56                      push  esi
00000859 57                      push  edi
0000085A BE[AA6B0000]            mov   esi, timer_msg ; Timer interrupt message
                                 ; 07/09/2014
0000085F 66BB0100                mov   bx, 1       ; Video page 1
                                ptmsg:
00000863 AC                      lodsb
00000864 08C0                    or    al, al
00000866 740F                    jz    short ptmsg_ok
```

```
00000868 56                        push  esi
00000869 6653                      push  bx
0000086B B42F                             mov     ah,  2Fh ; Green background, white forecolor
0000086D E8AB0C0000                call  write_tty
00000872 665B                      pop   bx
00000874 5E                        pop   esi
00000875 EBEC                      jmp   short ptmsg
                                   ;; 27/08/2014
                                   ;mov     edi, 0B8000h + 0A0h ; Row 1
                                   ;call printk
                                   ;
                                 ptmsg_ok:
                                   ; 07/09/2014
00000877 6631D2                    xor   dx, dx           ; column 0, row 0
0000087A E8C20D0000                call  set_cpos    ; set cursor position to 0,0
                                   ; 23/02/2015
                                   ; 25/08/2014
                                   ;mov  ebx, scounter          ; (seconds counter)
                                   ;dec  byte [ebx+1]           ; (for reading real time clock)
                                 ;dec   byte [scounter+1]
                                 ;;     jns   short timer_eoi        ; 0 -> 0FFh ?
                                 ;jns   short u_timer_retn
                                   ; 26/02/2015
                                 ;call  rtc_p
                                 ;mov   ebx, scounter          ; (seconds counter)
                                 ;mov   byte [ebx+1], 18  ; (18.2 timer ticks per second)
                                 ;dec   byte [ebx]        ; 19+18+18+18+18 (5)
                                 ;jnz   short timer_eoi        ; (109 timer ticks in 5 seconds)
                                 ;jnz   short u_timer_retn ; 06/02/2015
                                 ;mov   byte [ebx], 5
                                 ;inc   byte [ebx+1] ; 19
                                 ;;timer_eoi:
                                 ;;     mov   al, 20h ; END OF INTERRUPT COMMAND TO 8259
                                 ;;     out   20h, al     ; 8259 PORT
                                   ;
                                 ;u_timer_retn:  ; 06/02/2015
0000087F 5F                        pop   edi
00000880 5E                        pop   esi
                                   ;pop  es
                                   ;pop  ds
                                   ;pop  ebx
                                   ;pop  ecx
                                   ;pop  edx
                                   ;pop  eax
                                   ;iret
00000881 C3                        retn  ; 06/02/2015

                                   ; 28/08/2014
                                 irq0:
00000882 6A00                             push dword 0
00000884 EB48                      jmp   short which_irq
                                 irq1:
00000886 6A01                             push dword 1
00000888 EB44                      jmp   short which_irq
                                 irq2:
0000088A 6A02                             push dword 2
0000088C EB40                      jmp   short which_irq
                                 irq3:
                                   ; 20/11/2015
                                   ; 24/10/2015
0000088E 2EFF15[7B3F0000]          call  dword [cs:com2_irq3]
00000895 6A03                      push  dword 3
00000897 EB35                      jmp   short which_irq
                                 irq4:
                                   ; 20/11/2015
                                   ; 24/10/2015
00000899 2EFF15[773F0000]          call  dword [cs:com1_irq4]
000008A0 6A04                             push dword 4
000008A2 EB2A                      jmp   short which_irq
                                 irq5:
000008A4 6A05                             push dword 5
000008A6 EB26                      jmp   short which_irq
                                 irq6:
000008A8 6A06                             push dword 6
000008AA EB22                      jmp   short which_irq
                                 irq7:
000008AC 6A07                             push dword 7
000008AE EB1E                      jmp   short which_irq
                                 irq8:
000008B0 6A08                             push dword 8
000008B2 EB1A                      jmp   short which_irq
                                 irq9:
000008B4 6A09                             push dword 9
000008B6 EB16                      jmp   short which_irq
                                 irq10:
000008B8 6A0A                             push dword 10
000008BA EB12                      jmp   short which_irq
                                 irq11:
000008BC 6A0B                             push dword 11
000008BE EB0E                      jmp   short which_irq
                                 irq12:
000008C0 6A0C                             push dword 12
```

```
000008C2 EB0A                    jmp   short which_irq
                                irq13:
000008C4 6A0D                           push dword 13
000008C6 EB06                    jmp   short which_irq
                                irq14:
000008C8 6A0E                           push dword 14
000008CA EB02                    jmp   short which_irq
                                irq15:
000008CC 6A0F                           push dword 15
                                 ;jmp  short which_irq

                                 ; 19/10/2015
                                 ; 29/08/2014
                                 ; 21/08/2014
                                which_irq:
000008CE 870424                  xchg  eax, [esp]  ; 28/08/2014
000008D1 53                      push  ebx
000008D2 56                      push  esi
000008D3 57                      push  edi
000008D4 1E                      push  ds
000008D5 06                      push  es
                                 ;
000008D6 88C3                    mov   bl, al
                                 ;
000008D8 B810000000              mov   eax, KDATA
000008DD 8ED8                    mov   ds, ax
000008DF 8EC0                    mov   es, ax
                                 ; 19/10/2015
000008E1 FC                      cld
                                        ; 27/08/2014
000008E2 8105[596B0000]A000-            add     dword [scr_row], 0A0h
000008EA 0000
                                 ;
000008EC B417                    mov   ah, 17h     ; blue (1) background,
                                             ; light gray (7) forecolor
000008EE 8B3D[596B0000]                 mov     edi, [scr_row]
000008F4 B049                    mov   al, 'I'
000008F6 66AB                    stosw
000008F8 B052                    mov   al, 'R'
000008FA 66AB                    stosw
000008FC B051                    mov   al, 'Q'
000008FE 66AB                    stosw
00000900 B020                    mov   al, ' '
00000902 66AB                    stosw
00000904 88D8                    mov   al, bl
00000906 3C0A                    cmp   al, 10
00000908 7208                    jb    short iix
0000090A B031                    mov   al, '1'
0000090C 66AB                    stosw
0000090E 88D8                    mov   al, bl
00000910 2C0A                    sub   al, 10
                                iix:
00000912 0430                    add   al, '0'
00000914 66AB                    stosw
00000916 B020                    mov   al, ' '
00000918 66AB                    stosw
0000091A B021                    mov   al, '!'
0000091C 66AB                    stosw
0000091E B020                    mov   al, ' '
00000920 66AB                    stosw
                                 ; 23/02/2015
00000922 80FB07                  cmp   bl, 7 ; check for IRQ 8 to IRQ 15
00000925 0F868D010000            jna   iiret
0000092B B020                    mov   al, 20h  ; END OF INTERRUPT COMMAND TO
0000092D E6A0                    out   0A0h, al ; the 2nd 8259
0000092F E984010000              jmp     iiret
                                 ;
                                 ; 22/08/2014
                                 ;mov  al, 20h ; END OF INTERRUPT COMMAND TO 8259
                                 ;out  20h, al     ; 8259 PORT
                                 ;
                                 ;pop  es
                                 ;pop  ds
                                 ;pop  edi
                                 ;pop  esi
                                 ;pop  ebx
                                 ;pop  eax
                                 ;iret

                                 ; 02/04/2015
                                 ; 25/08/2014
                                exc0:
00000934 6A00                           push dword 0
00000936 E990000000                     jmp     cpu_except
                                exc1:
0000093B 6A01                           push dword 1
0000093D E989000000                     jmp     cpu_except
                                exc2:
00000942 6A02                           push dword 2
00000944 E982000000                     jmp     cpu_except
                                exc3:
00000949 6A03                           push dword 3
```

```
0000094B EB7E                         jmp     cpu_except
                              exc4:
0000094D 6A04                         push dword 4
0000094F EB7A                         jmp     cpu_except
                              exc5:
00000951 6A05                         push dword 5
00000953 EB76                         jmp     cpu_except
                              exc6:
00000955 6A06                         push dword 6
00000957 EB72                         jmp     cpu_except
                              exc7:
00000959 6A07                         push dword 7
0000095B EB6E                         jmp     cpu_except
                              exc8:
                                ; [esp] = Error code
0000095D 6A08                         push dword 8
0000095F EB5C                         jmp     cpu_except_en
                              exc9:
00000961 6A09                         push dword 9
00000963 EB66                         jmp     cpu_except
                              exc10:
                                ; [esp] = Error code
00000965 6A0A                         push dword 10
00000967 EB54                         jmp     cpu_except_en
                              exc11:
                                ; [esp] = Error code
00000969 6A0B                         push dword 11
0000096B EB50                         jmp     cpu_except_en
                              exc12:
                                ; [esp] = Error code
0000096D 6A0C                         push dword 12
0000096F EB4C                         jmp     cpu_except_en
                              exc13:
                                ; [esp] = Error code
00000971 6A0D                         push dword 13
00000973 EB48                         jmp     cpu_except_en
                              exc14:
                                ; [esp] = Error code
00000975 6A0E                         push dword 14
00000977 EB44                   jmp   short cpu_except_en
                              exc15:
00000979 6A0F                         push dword 15
0000097B EB4E                         jmp     cpu_except
                              exc16:
0000097D 6A10                         push dword 16
0000097F EB4A                         jmp     cpu_except
                              exc17:
                                ; [esp] = Error code
00000981 6A11                         push dword 17
00000983 EB38                   jmp   short cpu_except_en
                              exc18:
00000985 6A12                         push dword 18
00000987 EB42                   jmp   short cpu_except
                              exc19:
00000989 6A13                         push dword 19
0000098B EB3E                   jmp   short cpu_except
                              exc20:
0000098D 6A14                         push dword 20
0000098F EB3A                   jmp   short cpu_except
                              exc21:
00000991 6A15                         push dword 21
00000993 EB36                   jmp   short cpu_except
                              exc22:
00000995 6A16                         push dword 22
00000997 EB32                   jmp   short cpu_except
                              exc23:
00000999 6A17                         push dword 23
0000099B EB2E                   jmp   short cpu_except
                              exc24:
0000099D 6A18                         push dword 24
0000099F EB2A                   jmp   short cpu_except
                              exc25:
000009A1 6A19                         push dword 25
000009A3 EB26                   jmp   short cpu_except
                              exc26:
000009A5 6A1A                         push dword 26
000009A7 EB22                   jmp   short cpu_except
                              exc27:
000009A9 6A1B                         push dword 27
000009AB EB1E                   jmp   short cpu_except
                              exc28:
000009AD 6A1C                         push dword 28
000009AF EB1A                   jmp   short cpu_except
                              exc29:
000009B1 6A1D                         push dword 29
000009B3 EB16                   jmp   short cpu_except
                              exc30:
000009B5 6A1E                         push dword 30
000009B7 EB04                   jmp   short cpu_except_en
                              exc31:
000009B9 6A1F                         push dword 31
000009BB EB0E                         jmp     short cpu_except
```

```

                                  ; 19/10/2015
                                  ; 19/09/2015
                                  ; 01/09/2015
                                  ; 28/08/2015
                                  ; 28/08/2014
                                 cpu_except_en:
000009BD 87442404                 xchg  eax, [esp+4] ; Error code
000009C1 36A3[48810000]           mov   [ss:error_code], eax
000009C7 58                       pop   eax  ; Exception number
000009C8 870424                   xchg  eax, [esp]
                                        ; eax = eax before exception
                                        ; [esp] -> exception number
                                        ; [esp+4] -> EIP to return
                                  ; 19/10/2015
                                  ; 19/09/2015
                                  ; 01/09/2015
                                  ; 28/08/2015
                                  ; 29/08/2014
                                  ; 28/08/2014
                                  ; 25/08/2014
                                  ; 21/08/2014
                                 cpu_except:  ; CPU Exceptions
000009CB FC                       cld
000009CC 870424                   xchg  eax, [esp]
                                        ; eax = Exception number
                                        ; [esp] = eax (before exception)
000009CF 53                       push  ebx
000009D0 56                       push  esi
000009D1 57                       push  edi
000009D2 1E                       push  ds
000009D3 06                       push  es
                                  ; 28/08/2015
000009D4 66BB1000                 mov   bx, KDATA
000009D8 8EDB                     mov   ds, bx
000009DA 8EC3                     mov   es, bx
000009DC 0F20DB                   mov   ebx, cr3
000009DF 53                       push  ebx ; (*) page directory
                                  ; 19/10/2015
000009E0 FC                       cld
                                  ; 25/03/2015
000009E1 8B1D[A8700000]           mov   ebx, [k_page_dir]
000009E7 0F22DB                   mov   cr3, ebx
                                  ; 28/08/2015
000009EA 83F80E                   cmp   eax, 0Eh ; 14, PAGE FAULT
000009ED 7512                     jne   short cpu_except_nfp
000009EF E815290000               call  page_fault_handler
000009F4 21C0                     and   eax, eax
000009F6 0F84B8000000                    jz   iiretp ; 01/09/2015
000009FC B80E000000               mov   eax, 0Eh ; 14
                                 cpu_except_nfp:
                                  ; 02/04/2015
00000A01 BB[23070000]             mov   ebx, hang
00000A06 875C241C                 xchg  ebx, [esp+28]
                                        ; EIP (points to instruction which faults)
                                        ; New EIP (hang)
00000A0A 891D[4C810000]           mov   [FaultOffset], ebx
00000A10 C744242008000000         mov   dword [esp+32], KCODE ; kernel's code segment
00000A18 814C242400020000         or    dword [esp+36], 200h ; enable interrupts (set IF)
                                  ;
00000A20 88C4                     mov   ah, al
00000A22 240F                     and   al, 0Fh
00000A24 3C09                     cmp   al, 9
00000A26 7602                     jna   short h1ok
00000A28 0407                     add   al, 'A'-':'
                                 h1ok:
00000A2A D0EC                     shr   ah, 1
00000A2C D0EC                     shr   ah, 1
00000A2E D0EC                     shr   ah, 1
00000A30 D0EC                     shr   ah, 1
00000A32 80FC09                   cmp   ah, 9
00000A35 7603                     jna   short h2ok
00000A37 80C407                   add   ah, 'A'-':'
                                 h2ok:
00000A3A 86E0                     xchg  ah, al
00000A3C 66053030                 add   ax, '00'
00000A40 66A3[E66B0000]           mov   [excnstr], ax
                                  ;
                                  ; 29/08/2014
00000A46 A1[4C810000]             mov   eax, [FaultOffset]
00000A4B 51                       push  ecx
00000A4C 52                       push  edx
00000A4D 89E3                     mov   ebx, esp
                                  ; 28/08/2015
00000A4F B910000000               mov   ecx, 16       ; divisor value to convert binary number
                                                ; to hexadecimal string
                                  ;mov  ecx, 10         ; divisor to convert
                                                  ; binary number to decimal string
                                 b2d1:
00000A54 31D2                     xor   edx, edx
00000A56 F7F1                     div   ecx
00000A58 6652                     push  dx
```

```
00000A5A 39C8                     cmp   eax, ecx
00000A5C 73F6                     jnb   short b2d1
00000A5E BF[F16B0000]             mov   edi, EIPstr ; EIP value
                                                  ; points to instruction which faults
                                  ; 28/08/2015
00000A63 89C2                     mov   edx, eax
                                 b2d2:
                                  ;add  al, '0'
00000A65 8A82[31190000]           mov   al, [edx+hexchrs]
00000A6B AA                       stosb           ; write hexadecimal digit to its place
00000A6C 39E3                     cmp   ebx, esp
00000A6E 7606                     jna   short b2d3
00000A70 6658                     pop   ax
00000A72 88C2                     mov   dl, al
00000A74 EBEF                     jmp   short b2d2
                                 b2d3:
00000A76 B068                     mov   al, 'h' ; 28/08/2015
00000A78 AA                       stosb
00000A79 B020                     mov   al, 20h         ; space
00000A7B AA                       stosb
00000A7C 30C0                     xor   al, al          ; to do it an ASCIIZ string
00000A7E AA                       stosb
                                  ;
00000A7F 5A                       pop   edx
00000A80 59                       pop   ecx
                                  ;
00000A81 B44F                     mov   ah, 4Fh     ; red (4) background,
                                              ; white (F) forecolor
00000A83 BE[D66B0000]             mov   esi, exc_msg ; message offset
                                  ;
00000A88 EB11                     jmp   short piemsg
                                  ;
                                         ;add    dword [scr_row], 0A0h
                                         ;mov    edi, [scr_row]
                                         ;
                                  ;call       printk
                                  ;
                                  ;mov  al, 20h ; END OF INTERRUPT COMMAND TO 8259
                                  ;out  20h, al     ; 8259 PORT
                                  ;
                                  ;pop  es
                                  ;pop  ds
                                  ;pop  edi
                                  ;pop  esi
                                  ;pop  eax
                                  ;iret

                                  ; 28/08/2015
                                  ; 23/02/2015
                                  ; 20/08/2014
                                 ignore_int:
00000A8A 50                       push  eax
00000A8B 53                       push  ebx ; 23/02/2015
00000A8C 56                       push  esi
00000A8D 57                       push  edi
00000A8E 1E                       push  ds
00000A8F 06                       push  es
                                  ; 28/08/2015
00000A90 0F20D8                   mov   eax, cr3
00000A93 50                       push  eax ; (*) page directory
                                  ;
00000A94 B467                     mov   ah, 67h     ; brown (6) background,
                                              ; light gray (7) forecolor
00000A96 BE[946B0000]             mov   esi, int_msg ; message offset
                                 piemsg:
                                         ; 27/08/2014
00000A9B 8105[596B0000]A000-             add     dword [scr_row], 0A0h
00000AA3 0000
00000AA5 8B3D[596B0000]                  mov     edi, [scr_row]
                                         ;
00000AAB E8B0FCFFFF               call  printk
                                  ;
                                  ; 23/02/2015
00000AB0 B020                     mov   al, 20h  ; END OF INTERRUPT COMMAND TO
00000AB2 E6A0                     out   0A0h, al ; the 2nd 8259
                                 iiretp: ; 01/09/2015
                                  ; 28/08/2015
00000AB4 58                       pop   eax ; (*) page directory
00000AB5 0F22D8                   mov   cr3, eax
                                  ;
                                 iiret:
                                  ; 22/08/2014
00000AB8 B020                     mov   al, 20h ; END OF INTERRUPT COMMAND TO 8259
00000ABA E620                     out   20h, al     ; 8259 PORT
                                  ;
00000ABC 07                       pop   es
00000ABD 1F                       pop   ds
00000ABE 5F                       pop   edi
00000ABF 5E                       pop   esi
00000AC0 5B                       pop   ebx ; 29/08/2014
00000AC1 58                       pop   eax
00000AC2 CF                       iretd
```

```

                                 ; 26/02/2015
                                 ; 07/09/2014
                                 ; 25/08/2014
                                rtc_int:       ; Real Time Clock Interrupt (IRQ 8)
                                 ; 22/08/2014
00000AC3 50                      push  eax
00000AC4 53                      push  ebx ; 29/08/2014
00000AC5 56                      push  esi
00000AC6 57                      push  edi
00000AC7 1E                      push  ds
00000AC8 06                      push  es
                                 ;
00000AC9 B810000000              mov   eax, KDATA
00000ACE 8ED8                    mov   ds, ax
00000AD0 8EC0                    mov   es, ax
                                 ;
                                 ; 25/08/2014
00000AD2 E884000000              call  rtc_p
                                 ;
                                 ; 22/02/2015 - dsectpm.s
                                 ; [ source: http://wiki.osdev.org/RTC ]
                                 ; read status register C to complete procedure
                                 ;(it is needed to get a next IRQ 8)
00000AD7 B00C                    mov   al, 0Ch ;
00000AD9 E670                    out   70h, al ; select register C
00000ADB 90                      nop
00000ADC E471                    in    al, 71h ; just throw away contents
                                 ; 22/02/2015
00000ADE B020                    MOV   AL,EOI            ; END OF INTERRUPT
00000AE0 E6A0                    OUT   INTB00,AL   ; FOR CONTROLLER #2
                                 ;
00000AE2 EBD4                    jmp   short iiret

                                 ; 22/08/2014
                                 ; IBM PC/AT BIOS source code ----- 10/06/85 (bios.asm)
                                 ; (INT 1Ah)
                                 ;; Linux (v0.12) source code (main.c) by Linus Torvalds (1991)
                                time_of_day:
00000AE4 E866010000              call  UPD_IPR                 ; WAIT TILL UPDATE NOT IN PROGRESS
00000AE9 726F                           jc      short rtc_retn
00000AEB B000                    mov   al, CMOS_SECONDS
00000AED E845010000              call  CMOS_READ
00000AF2 A2[1C710000]            mov   [time_seconds], al
00000AF7 B002                    mov   al, CMOS_MINUTES
00000AF9 E839010000              call  CMOS_READ
00000AFE A2[1D710000]            mov   [time_minutes], al
00000B03 B004                    mov   al, CMOS_HOURS
00000B05 E82D010000              call  CMOS_READ
00000B0A A2[1E710000]                   mov     [time_hours], al
00000B0F B006                    mov   al, CMOS_DAY_WEEK
00000B11 E821010000              call  CMOS_READ
00000B16 A2[1F710000]            mov   [date_wday], al
00000B1B B007                    mov   al, CMOS_DAY_MONTH
00000B1D E815010000              call  CMOS_READ
00000B22 A2[20710000]            mov   [date_day], al
00000B27 B008                    mov   al, CMOS_MONTH
00000B29 E809010000              call  CMOS_READ
00000B2E A2[21710000]            mov   [date_month], al
00000B33 B009                    mov   al, CMOS_YEAR
00000B35 E8FD000000              call  CMOS_READ
00000B3A A2[22710000]            mov   [date_year], al
00000B3F B032                    mov   al, CMOS_CENTURY
00000B41 E8F1000000              call  CMOS_READ
00000B46 A2[23710000]            mov   [date_century], al
                                 ;
00000B4B B000                    mov   al, CMOS_SECONDS
00000B4D E8E5000000              call  CMOS_READ
00000B52 3A05[1C710000]          cmp   al, [time_seconds]
00000B58 758A                    jne   short time_of_day

                                rtc_retn:
00000B5A C3                      retn

                                rtc_p:
                                 ; 07/09/2014
                                 ; 29/08/2014
                                 ; 27/08/2014
                                 ; 25/08/2014
                                 ; Print Real Time Clock content
                                 ;
                                 ;
00000B5B E884FFFFFF              call  time_of_day
00000B60 72F8                    jc    short rtc_retn
                                 ;
00000B62 3A05[486C0000]          cmp   al, [ptime_seconds]
00000B68 74F0                           je      short rtc_retn ; 29/08/2014
                                 ;
00000B6A A2[486C0000]            mov   [ptime_seconds], al
                                 ;
00000B6F A0[23710000]            mov   al, [date_century]
00000B74 E8F1000000              call  bcd_to_ascii
```

```
00000B79 66A3[156C0000]           mov   [datestr+6], ax
00000B7F A0[22710000]             mov   al, [date_year]
00000B84 E8E1000000               call  bcd_to_ascii
00000B89 66A3[176C0000]           mov   [datestr+8], ax
00000B8F A0[21710000]             mov   al, [date_month]
00000B94 E8D1000000               call  bcd_to_ascii
00000B99 66A3[126C0000]           mov   [datestr+3], ax
00000B9F A0[20710000]             mov   al, [date_day]
00000BA4 E8C1000000               call  bcd_to_ascii
00000BA9 66A3[0F6C0000]           mov   [datestr], ax
                                  ;
00000BAF 0FB61D[1F710000]         movzx ebx, byte [date_wday]
00000BB6 C0E302                   shl   bl, 2
00000BB9 81C3[286C0000]           add   ebx, daytmp
00000BBF 8B03                     mov   eax, [ebx]
00000BC1 A3[1A6C0000]             mov   [daystr], eax
                                  ;
00000BC6 A0[1E710000]             mov   al, [time_hours]
00000BCB E89A000000               call  bcd_to_ascii
00000BD0 66A3[1E6C0000]           mov   [timestr], ax
00000BD6 A0[1D710000]             mov   al, [time_minutes]
00000BDB E88A000000               call  bcd_to_ascii
00000BE0 66A3[216C0000]           mov   [timestr+3], ax
00000BE6 A0[1C710000]             mov   al, [time_seconds]
00000BEB E87A000000               call  bcd_to_ascii
00000BF0 66A3[246C0000]           mov   [timestr+6], ax
                                  ;
00000BF6 BE[FD6B0000]             mov   esi, rtc_msg ; message offset
                                  ; 23/02/2015
00000BFB 52                       push  edx
00000BFC 51                       push  ecx
                                  ; 07/09/2014
00000BFD 66BB0200                 mov   bx, 2       ; Video page 2
                                 prtmsg:
00000C01 AC                       lodsb
00000C02 08C0                     or    al, al
00000C04 740F                     jz    short prtmsg_ok
00000C06 56                       push  esi
00000C07 6653                     push  bx
00000C09 B43F                            mov  ah, 3Fh     ; cyan (6) background,
                                              ; white (F) forecolor
00000C0B E80D090000               call  write_tty
00000C10 665B                     pop   bx
00000C12 5E                       pop   esi
00000C13 EBEC                     jmp   short prtmsg
                                  ;
                                  ;mov  edi, 0B8000h+0A0h+0A0h ; Row 2
                                  ;call printk
                                 prtmsg_ok:
                                  ; 07/09/2014
00000C15 6631D2                   xor   dx, dx            ; column 0, row 0
00000C18 E8240A0000               call  set_cpos    ; set curspor position to 0,0
                                  ; 23/02/2015
00000C1D 59                       pop   ecx
00000C1E 5A                       pop   edx
00000C1F C3                       retn

                                 ; Default IRQ 7 handler against spurious IRQs (from master PIC)
                                 ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
                                 default_irq7:
00000C20 6650                     push  ax
00000C22 B00B                     mov   al, 0Bh  ; In-Service register
00000C24 E620                     out   20h, al
00000C26 EB00                            jmp short $+2
00000C28 EB00                     jmp short $+2
00000C2A E420                     in    al, 20h
00000C2C 2480                     and   al, 80h ; bit 7 (is it real IRQ 7 or fake?)
00000C2E 7404                            jz      short irq7_iret ; Fake (spurious) IRQ, do not send EOI
00000C30 B020                            mov     al, 20h ; EOI
00000C32 E620                     out   20h, al
                                 irq7_iret:
00000C34 6658                     pop   ax
00000C36 CF                       iretd

                                  ; 22/08/2014
                                  ; IBM PC/AT BIOS source code ----- 10/06/85 (test4.asm)
                                 CMOS_READ:
00000C37 9C                       pushf       ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
00000C38 D0C0                     rol   al, 1 ; MOVE NMI BIT TO LOW POSITION
00000C3A F9                       stc         ; FORCE NMI BIT ON IN CARRY FLAG
00000C3B D0D8                     rcr   al, 1 ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
00000C3D FA                       cli         ; DISABLE INTERRUPTS
00000C3E E670                     out   CMOS_PORT, al     ; ADDRESS LOCATION AND DISABLE NMI
00000C40 90                       nop         ; I/O DELAY
00000C41 E471                     in    al, CMOS_DATA     ; READ THE REQUESTED CMOS LOCATION
00000C43 6650                     push  ax    ; SAVE (AH) REGISTER VALUE AND CMOS BYTE
                                  ; 15/03/2015 ; IBM PC/XT Model 286 BIOS source code
                                             ; ----- 10/06/85 (test4.asm)
00000C45 B01E                     mov   al, CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
                                  ;mov  al, CMOS_REG_D*2 ; GET ADDRESS OF DEFAULT LOCATION
00000C47 D0D8                     rcr   al, 1 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
00000C49 E670                     out   CMOS_PORT, al     ; SET DEFAULT TO READ ONLY REGISTER
```

```
00000C4B 6658                     pop   ax    ; RESTORE (AH) AND (AL), CMOS BYTE
00000C4D 9D                       popf
00000C4E C3                       retn        ; RETURN WITH FLAGS RESTORED

                                  ; 22/08/2014
                                  ; IBM PC/AT BIOS source code ----- 10/06/85 (bios2.asm)
                                 UPD_IPR:                       ; WAIT TILL UPDATE NOT IN PROGRESS
00000C4F 51                       push  ecx
00000C50 B9FFFF0000               mov   ecx, 65535        ; SET TIMEOUT LOOP COUNT (= 800)
                                        ; mov cx, 800
                                 UPD_10:
00000C55 B00A                     mov   al, CMOS_REG_A          ; ADDRESS STATUS REGISTER A
00000C57 FA                       cli                     ; NO TIMER INTERRUPTS DURING UPDATES
00000C58 E8DAFFFFFF               call  CMOS_READ         ; READ UPDATE IN PROCESS FLAG
00000C5D A880                     test  al, 80h                 ; IF UIP BIT IS ON ( CANNOT READ TIME )
00000C5F 7406                     jz    short UPD_90            ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
00000C61 FB                       sti                     ; ALLOW INTERRUPTS WHILE WAITING
00000C62 E2F1                     loop  UPD_10                  ; LOOP TILL READY OR TIMEOUT
00000C64 31C0                     xor   eax, eax          ; CLEAR RESULTS IF ERROR
                                        ; xor ax, ax
00000C66 F9                       stc                     ; SET CARRY FOR ERROR
                                 UPD_90:
00000C67 59                       pop   ecx               ; RESTORE CALLERS REGISTER
00000C68 FA                       cli                     ; INTERRUPTS OFF DURING SET
00000C69 C3                       retn                    ; RETURN WITH CY FLAG SET

                                 bcd_to_ascii:
                                  ; 25/08/2014
                                  ; INPUT ->
                                  ;     al = Packed BCD number
                                  ; OUTPUT ->
                                  ;     ax  = ASCII word/number
                                  ;
                                  ; Erdogan Tan - 1998 (proc_hex) - TRDOS.ASM (2004-2011)
                                  ;
00000C6A D410                     db 0D4h,10h                     ; Undocumented inst. AAM
                                                          ; AH = AL / 10h
                                                          ; AL = AL MOD 10h
00000C6C 660D3030                 or ax,'00'                      ; Make it ASCII based

00000C70 86E0                            xchg ah, al

00000C72 C3                       retn


                                 %include 'keyboard.inc' ; 07/03/2015
                             <1> ; Retro UNIX 386 v1 Kernel - KEYBOARD.INC
                             <1> ; Last Modification: 17/10/2015
                             <1> ;          (Keyboard Data is in 'KYBDATA.INC')
                             <1> ;
                             <1> ; ///////// KEYBOARD FUNCTIONS (PROCEDURES) ///////////////
                             <1>
                             <1> ; 30/06/2015
                             <1> ; 11/03/2015
                             <1> ; 28/02/2015
                             <1> ; 25/02/2015
                             <1> ; 20/02/2015
                             <1> ; 18/02/2015
                             <1> ; 03/12/2014
                             <1> ; 07/09/2014
                             <1> ; KEYBOARD INTERRUPT HANDLER
                             <1> ; (kb_int - Retro UNIX 8086 v1 - U0.ASM, 30/06/2014)
                             <1>
                             <1> ;getch:
                             <1> ;; 18/02/2015
                             <1> ;; This routine will be replaced with Retro UNIX 386
                             <1> ;; version of Retro UNIX 8086 getch (tty input)
                             <1> ;; routine, later... (multi tasking ability)
                             <1> ;; 28/02/2015
                             <1> ;sti   ; enable interrupts
                             <1> ;;
                             <1> ;;push esi
                             <1> ;;push ebx
                             <1> ;;xor  ebx, ebx
                             <1> ;;mov  bl, [ptty]  ; active_page
                             <1> ;;mov  esi, ebx
                             <1> ;;shl  si, 1
                             <1> ;;add  esi, ttychr
                             <1> ;getch_1:
                             <1> ;;mov  ax, [esi]
                             <1> ;mov   ax, [ttychr] ; video page 0 (tty0)
                             <1> ;and   ax, ax
                             <1> ;jz    short getch_2
                             <1> ;mov   word [ttychr], 0
                             <1> ;;mov  word [esi], 0
                             <1> ;;pop  ebx
                             <1> ;;pop  esi
                             <1> ;retn
                             <1> ;getch_2:
                             <1> ;hlt   ; not proper for multi tasking!
                             <1> ;      ; (temporary halt for now)
                             <1> ;      ; 'sleep' on tty
                             <1> ;       ; will (must) be located here
                             <1> ;nop
                             <1> ;jmp   short getch_1
                             <1>
                             <1> keyb_int:
                             <1>  ; 30/06/2015
                             <1>  ; 25/02/2015
                             <1>  ; 20/02/2015
                             <1>  ; 03/12/2014 (getc_int - INT 16h modifications)
                             <1>  ; 07/09/2014 - Retro UNIX 386 v1
                             <1>  ; 30/06/2014
                             <1>  ; 10/05/2013
                             <1>        ; Retro Unix 8086 v1 feature only!
                             <1>  ; 03/03/2014
                             <1>
00000C73 1E                  <1>  push  ds
00000C74 53                  <1>  push  ebx
00000C75 50                  <1>  push  eax
                             <1>  ;
00000C76 66B81000            <1>  mov   ax, KDATA
00000C7A 8ED8                <1>  mov   ds, ax
                             <1>  ;
00000C7C 9C                  <1>  pushfd
00000C7D 0E                  <1>  push  cs
00000C7E E823020000          <1>  call  kb_int  ; int_09h
                             <1>  ;
00000C83 B411                <1>  mov   ah, 11h      ; 03/12/2014
                             <1>  ;call getc
00000C85 E856000000          <1>  call  int_16h ; 30/06/2015
00000C8A 7450                <1>  jz    short keyb_int4
                             <1>  ;
00000C8C B410                <1>  mov   ah, 10h      ; 03/12/2014
                             <1>  ;call getc
00000C8E E84D000000          <1>  call  int_16h ; 30/06/2015
                             <1>  ;
                             <1>  ; 20/02/2015
00000C93 0FB61D[D6700000]    <1>         movzx   ebx, byte [ptty]  ; active_page
                             <1>  ;
00000C9A 20C0                <1>  and   al, al
00000C9C 751E                <1>  jnz   short keyb_int1
                             <1>  ;
00000C9E 80FC68              <1>  cmp   ah, 68h     ; ALT + F1 key
00000CA1 7219                <1>  jb    short keyb_int1
00000CA3 80FC6F              <1>  cmp   ah, 6Fh  ; ALT + F8 key
00000CA6 7714                <1>  ja    short keyb_int1
                             <1>  ;
00000CA8 88D8                <1>  mov   al, bl
00000CAA 0468                <1>  add   al, 68h
00000CAC 38E0                <1>  cmp   al, ah
00000CAE 7409                <1>  je    short keyb_int0
00000CB0 88E0                <1>  mov   al, ah
00000CB2 2C68                <1>  sub   al, 68h
00000CB4 E8510B0000          <1>  call  tty_sw
                             <1>  ;movzx      ebx, [ptty]  ; active_page
                             <1> keyb_int0: ; 30/06/2015
00000CB9 6631C0              <1>  xor   ax, ax
                             <1> keyb_int1:
00000CBC D0E3                <1>  shl   bl, 1
00000CBE 81C3[D8700000]      <1>  add   ebx, ttychr
                             <1>  ;
00000CC4 6609C0              <1>  or    ax, ax
00000CC7 7406                <1>  jz    short keyb_int2
                             <1>  ;
00000CC9 66833B00            <1>  cmp   word [ebx], 0
00000CCD 7703                <1>         ja      short keyb_int3
                             <1> keyb_int2:
00000CCF 668903              <1>         mov  [ebx], ax  ; Save ascii code
                             <1>                    ; and scan code of the character
                             <1>                    ; for current tty (or last tty
                             <1>                    ; just before tty switch).
                             <1> keyb_int3:
00000CD2 A0[D6700000]        <1>         mov     al, [ptty]
00000CD7 E811480000          <1>  call  wakeup
                             <1>  ;
                             <1> keyb_int4:
00000CDC 58                  <1>  pop   eax
00000CDD 5B                  <1>  pop   ebx
00000CDE 1F                  <1>  pop   ds
00000CDF CF                  <1>  iret
                             <1>
                             <1> ; 18/02/2015
                             <1> ; REMINDER: Only 'keyb_int' (IRQ 9) must call getc.
                             <1> ; 'keyb_int' always handles 'getc' at 1st and puts the
                             <1> ; scancode and ascii code of the character
                             <1> ; in the tty input (ttychr) buffer.
                             <1> ; Test procedures must call 'getch' for tty input
                             <1> ; otherwise, 'getc' will not be able to return to the caller
                             <1> ; due to infinite (key press) waiting loop.
                             <1> ;
                             <1> ; 03/12/2014
                             <1> ; 26/08/2014
                             <1> ; KEYBOARD I/O
                             <1> ; (INT_16h - Retro UNIX 8086 v1 - U9.ASM, 30/06/2014)
                             <1>
                             <1> ;NOTE: 'k0' to 'k7' are name of OPMASK registers.
                             <1> ;(The reason of using '_k' labels!!!) (27/08/2014)
                             <1> ;NOTE: 'NOT' keyword is '~' unary operator in NASM.
                             <1> ;('NOT LC_HC' --> '~LC_HC') (bit reversing operator)
                             <1>
                             <1> int_16h: ; 30/06/2015
                             <1> ;getc:
00000CE0 9C                  <1>  pushfd      ; 28/08/2014
00000CE1 0E                  <1>  push  cs
00000CE2 E801000000          <1>  call  getc_int
00000CE7 C3                  <1>  retn
                             <1>
                             <1> getc_int:
                             <1>  ; 28/02/2015
                             <1>  ; 03/12/2014 (derivation from pc-xt-286 bios source code -1986-,
                             <1>  ;           instead of pc-at bios - 1985-)
                             <1>  ; 28/08/2014 (_k1d)
                             <1>  ; 30/06/2014
                             <1>  ; 03/03/2014
                             <1>  ; 28/02/2014
                             <1>  ; Derived from "KEYBOARD_IO_1" procedure of IBM "pc-xt-286"
                             <1>  ; rombios source code (21/04/1986)
                             <1>  ;      'keybd.asm', INT 16H, KEYBOARD_IO
                             <1>  ;
                             <1>  ; KYBD --- 03/06/86  KEYBOARD BIOS
                             <1>  ;
                             <1>  ;--- INT 16 H ------------------------------------------------------------------
                             <1>  ; KEYBOARD I/O                                                   :
                             <1>  ;     THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT                      :
                             <1>  ; INPUT                                                          :
                             <1>  ;     (AH)= 00H  READ THE NEXT ASCII CHARACTER ENTERED FROM THE KEYBOARD,   :
                             <1>  ;              RETURN THE RESULT IN (AL), SCAN CODE IN (AH).             :
                             <1>  ;              THIS IS THE COMPATIBLE READ INTERFACE, EQUIVALENT TO THE  :
                             <1>  ;                  STANDARD PC OR PCAT KEYBOARD                         :
                             <1>  ;------------------------------------------------------------------------------:
                             <1>  ;     (AH)= 01H  SET THE ZERO FLAG TO INDICATE IF AN ASCII CHARACTER IS     :
                             <1>  ;              AVAILABLE TO BE READ FROM THE KEYBOARD BUFFER.           :
                             <1>  ;              (ZF)= 1 -- NO CODE AVAILABLE                              :
                             <1>  ;              (ZF)= 0 -- CODE IS AVAILABLE  (AX)= CHARACTER              :
                             <1>  ;              IF (ZF)= 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS :
                             <1>  ;              IN (AX), AND THE ENTRY REMAINS IN THE BUFFER.              :
                             <1>  ;              THIS WILL RETURN ONLY PC/PCAT KEYBOARD COMPATIBLE CODES    :
                             <1>  ;------------------------------------------------------------------------------:

                             <1>  ;     (AH)= 02H  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER              :
                             <1>  ;              THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE        :
                             <1>  ;              EQUATES FOR @KB_FLAG                                     :
                             <1>  ;------------------------------------------------------------------------------:

                             <1>  ;     (AH)= 03H  SET TYPAMATIC RATE AND DELAY                                :
                             <1>  ;           (AL) = 05H                                                       :
                             <1>  ;           (BL) = TYPAMATIC RATE (BITS 5 - 7 MUST BE RESET TO 0)            :
                             <1>  ;                                                                :
                             <1>  ;                     REGISTER     RATE      REGISTER     RATE                 :
                             <1>  ;                      VALUE     SELECTED     VALUE     SELECTED               :
                             <1>  ;                     --------------------------------------------             :
                             <1>  ;                 00H        30.0        10H        7.5                 :
                             <1>  ;                 01H        26.7        11H        6.7                 :
                             <1>  ;                 02H        24.0        12H        6.0                 :
                             <1>  ;                 03H        21.8        13H        5.5                 :
                             <1>  ;                 04H        20.0        14H        5.0                 :
                             <1>  ;                 05H        18.5        15H        4.6                 :
                             <1>  ;                 06H        17.1        16H        4.3                 :
                             <1>  ;                 07H        16.0        17H        4.0                 :
                             <1>  ;                 08H        15.0        18H        3.7                 :
                             <1>  ;                 09H        13.3        19H        3.3                 :
                             <1>  ;                 0AH        12.0        1AH        3.0                 :
                             <1>  ;                 0BH        10.9        1BH        2.7                 :
                             <1>        ;                 0CH        10.0        1CH        2.5                 :
                             <1>  ;                 0DH         9.2        1DH        2.3                 :
                             <1>  ;                 0EH         8.6        1EH        2.1                 :
                             <1>  ;                 0FH         8.0        1FH        2.0                 :
                             <1>  ;                                                           :
                             <1>  ;           (BH) = TYPAMATIC DELAY  (BITS 2 - 7 MUST BE RESET TO 0)         :
                             <1>  ;                                                                :
                             <1>  ;                     REGISTER     DELAY                                       :
                             <1>  ;                      VALUE       VALUE                                       :
                             <1>  ;                     ------------------                                       :
                             <1>  ;                 00H        250 ms                                     :
                             <1>  ;                 01H        500 ms                                     :
                             <1>  ;                 02H        750 ms                                     :
                             <1>  ;                 03H       1000 ms                                     :
                             <1>  ;------------------------------------------------------------------------------:
                             <1>  ;     (AH)= 05H  PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD    :
                             <1>  ;              BUFFER AS IF STRUCK FROM KEYBOARD                         :
                             <1>  ;              ENTRY:  (CL) = ASCII CHARACTER                      :
                             <1>  ;                      (CH) = SCAN CODE                                  :
                             <1>  ;              EXIT:   (AH) = 00H = SUCCESSFUL OPERATION                 :
                             <1>  ;                      (AL) = 01H = UNSUCCESSFUL - BUFFER FULL           :
                             <1>  ;              FLAGS:  CARRY IF ERROR                                    :
                             <1>  ;----------------------------------------------------------------------------:
                             <1>  ;     (AH)= 10H  EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD,         :
                             <1>  ;              OTHERWISE SAME AS FUNCTION AH=0                             :
                             <1>  ;----------------------------------------------------------------------------:
                             <1>  ;     (AH)= 11H  EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,          :
                             <1>  ;              OTHERWISE SAME AS FUNCTION AH=1                             :
                             <1>  ;----------------------------------------------------------------------------:
                             <1>  ;     (AH)= 12H  RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER           :
                             <1>  ;              AL = BITS FROM KB_FLAG, AH = BITS FOR LEFT AND RIGHT        :
                             <1>  ;              CTL AND ALT KEYS FROM KB_FLAG_1 AND KB_FLAG_3               :
                             <1>  ; OUTPUT                                                           :
                             <1>  ;     AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED                          :
                             <1>  ;     ALL REGISTERS RETAINED                                              :
                             <1>  ;-----------------------------------------------------------------------------
                             <1>
00000CE8 FB                     <1>  sti                     ; INTERRUPTS BACK ON
00000CE9 1E                     <1>  push  ds                ; SAVE CURRENT DS
00000CEA 53                     <1>  push  ebx               ; SAVE BX TEMPORARILY
                             <1>  ;push ecx                ; SAVE CX TEMPORARILY
00000CEB 66BB1000               <1>         mov     bx, KDATA
00000CEF 8EDB                   <1>  mov   ds, bx                  ; PUT SEGMENT VALUE OF DATA AREA INTO DS
00000CF1 08E4                   <1>  or    ah, ah                  ; CHECK FOR (AH)= 00H
00000CF3 7439                   <1>  jz    short _K1         ; ASCII_READ
00000CF5 FECC                   <1>  dec   ah                       ; CHECK FOR (AH)= 01H
00000CF7 7452                   <1>         jz      short _K2              ; ASCII_STATUS
00000CF9 FECC                   <1>  dec   ah                ; CHECK FOR (AH)= 02H
00000CFB 0F8485000000           <1>         jz      _K3                    ; SHIFT STATUS
00000D01 FECC                   <1>  dec   ah                ; CHECK FOR (AH)= 03H
00000D03 0F8484000000           <1>         jz      _K300                  ; SET TYPAMATIC RATE/DELAY
00000D09 80EC02                 <1>  sub   ah, 2             ; CHECK FOR (AH)= 05H
00000D0C 0F84A1000000           <1>         jz      _K500                  ; KEYBOARD WRITE
                             <1> _KIO1:
00000D12 80EC0B                 <1>  sub   ah, 11                  ; AH =  10H
00000D15 740B                   <1>  jz    short _K1E        ; EXTENDED ASCII READ
00000D17 FECC                   <1>  dec   ah                ; CHECK FOR (AH)= 11H
00000D19 7421                   <1>  jz    short _K2E        ; EXTENDED_ASCII_STATUS
00000D1B FECC                   <1>  dec   ah                ; CHECK FOR (AH)= 12H
00000D1D 7449                   <1>  jz    short _K3E        ; EXTENDED_SHIFT_STATUS
                             <1> _KIO_EXIT:
                             <1>  ;pop  ecx               ; RECOVER REGISTER
00000D1F 5B                     <1>  pop   ebx               ; RECOVER REGISTER
00000D20 1F                     <1>  pop   ds                ; RECOVER SEGMENT
00000D21 CF                     <1>  iretd                   ; INVALID COMMAND, EXIT
                             <1>
                             <1>  ;-----      ASCII CHARACTER
                             <1> _K1E:
00000D22 E8B9000000             <1>  call  _K1S              ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
00000D27 E82E010000             <1>  call  _KIO_E_XLAT       ; ROUTINE TO XLATE FOR EXTENDED CALLS
00000D2C EBF1                   <1>  jmp   short _KIO_EXIT         ; GIVE IT TO THE CALLER
                             <1> _K1:
00000D2E E8AD000000             <1>  call  _K1S              ; GET A CHARACTER FROM THE BUFFER
00000D33 E82D010000             <1>  call  _KIO_S_XLAT       ; ROUTINE TO XLATE FOR STANDARD CALLS
00000D38 72F4                   <1>  jc    short _K1         ; CARRY SET MEANS TROW CODE AWAY
                             <1> _K1A:
00000D3A EBE3                   <1>  jmp   short _KIO_EXIT         ; RETURN TO CALLER
                             <1>
                             <1>  ;-----      ASCII STATUS
                             <1> _K2E:
00000D3C E8EA000000             <1>  call  _K2S              ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
00000D41 7420                   <1>  jz    short _K2B        ; RETURN IF BUFFER EMPTY
00000D43 9C                     <1>  pushf                   ; SAVE ZF FROM TEST
00000D44 E811010000             <1>  call  _KIO_E_XLAT       ; ROUTINE TO XLATE FOR EXTENDED CALLS
00000D49 EB17                   <1>  jmp   short _K2A          ; GIVE IT TO THE CALLER
                             <1> _K2:
00000D4B E8DB000000             <1>  call  _K2S              ; TEST FOR CHARACTER IN BUFFER
00000D50 7411                   <1>  jz    short _K2B        ; RETURN IF BUFFER EMPTY
00000D52 9C                     <1>  pushf                   ; SAVE ZF FROM TEST
00000D53 E80D010000             <1>  call  _KIO_S_XLAT       ; ROUTINE TO XLATE FOR STANDARD CALLS
00000D58 7308                   <1>  jnc   short _K2A          ; CARRY CLEAR MEANS PASS VALID CODE
00000D5A 9D                     <1>  popf                    ; INVALID CODE FOR THIS TYPE OF CALL
00000D5B E880000000             <1>  call  _K1S              ; THROW THE CHARACTER AWAY
00000D60 EBE9                   <1>  jmp   short _K2         ; GO LOOK FOR NEXT CHAR, IF ANY
                             <1> _K2A:
00000D62 9D                     <1>  popf                    ; RESTORE ZF FROM TEST
                             <1> _K2B:
                             <1>  ;pop  ecx               ; RECOVER REGISTER
00000D63 5B                     <1>  pop   ebx               ; RECOVER REGISTER
00000D64 1F                     <1>  pop   ds                ; RECOVER SEGMENT
00000D65 CA0400                 <1>  retf  4                 ; THROW AWAY (e)FLAGS
                             <1>
                             <1>  ;-----      SHIFT STATUS
                             <1> _K3E:                                          ; GET THE EXTENDED SHIFT STATUS FLAGS
00000D68 8A25[946A0000]         <1>  mov   ah, [KB_FLAG_1]         ; GET SYSTEM SHIFT KEY STATUS
00000D6E 80E404                 <1>  and   ah, SYS_SHIFT           ; MASK ALL BUT SYS KEY BIT
                             <1>  ;mov  cl, 5             ; SHIFT THEW SYSTEMKEY BIT OVER TO
                             <1>  ;shl  ah, cl                  ; BIT 7 POSITION
00000D71 C0E405                 <1>         shl  ah, 5
00000D74 A0[946A0000]           <1>  mov   al, [KB_FLAG_1]         ; GET SYSTEM SHIFT STATES BACK
00000D79 2473                   <1>  and   al, 01110011b           ; ELIMINATE SYS SHIFT, HOLD_STATE AND INS_SHIFT
00000D7B 08C4                   <1>  or    ah, al                   ; MERGE REMAINING BITS INTO AH
```

```
00000D7D A0[966A0000]          <1>  mov   al, [KB_FLAG_3]         ; GET RIGHT CTL AND ALT
00000D82 240C                  <1>  and   al, 00001100b           ; ELIMINATE LC_E0 AND LC_E1
00000D84 08C4                  <1>  or    ah, al                  ; OR THE SHIFT FLAGS TOGETHER
                               <1> _K3:
00000D86 A0[936A0000]          <1>  mov   al, [KB_FLAG]           ; GET THE SHIFT STATUS FLAGS
00000D8B EB92                  <1>  jmp   short _KIO_EXIT         ; RETURN TO CALLER
                               <1>
                               <1>  ;-----      SET TYPAMATIC RATE AND DELAY
                               <1> _K300:
00000D8D 3C05                  <1>  cmp   al, 5           ; CORRECT FUNCTION CALL?
00000D8F 758E                  <1>  jne   short _KIO_EXIT         ; NO, RETURN
00000D91 F6C3E0                <1>        test  bl, 0E0h          ; TEST FOR OUT-OF-RANGE RATE
00000D94 7589                  <1>  jnz   short _KIO_EXIT         ; RETURN IF SO
00000D96 F6C7FC                <1>  test  BH, 0FCh        ; TEST FOR OUT-OF-RANGE DELAY
00000D99 7584                  <1>  jnz   short _KIO_EXIT         ; RETURN IF SO
00000D9B B0F3                  <1>  mov   al, KB_TYPA_RD          ; COMMAND FOR TYPAMATIC RATE/DELAY
00000D9D E8B6060000            <1>  call  SND_DATA        ; SEND TO KEYBOARD
                               <1>  ;mov  cx, 5           ; SHIFT COUNT
                               <1>  ;shl  bh, cl                  ; SHIFT DELAY OVER
00000DA2 C0E705                <1>  shl   bh, 5
00000DA5 88D8                  <1>  mov   al, bl                  ; PUT IN RATE
00000DA7 08F8                  <1>  or    al, bh                  ; AND DELAY
00000DA9 E8AA060000            <1>  call  SND_DATA        ; SEND TO KEYBOARD
00000DAE E96CFFFFFF            <1>         jmp     _KIO_EXIT               ; RETURN TO CALLER
                               <1>
                               <1>  ;-----      WRITE TO KEYBOARD BUFFER
                               <1> _K500:
00000DB3 56                    <1>  push  esi             ; SAVE SI (esi)
00000DB4 FA                    <1>  cli                   ;
00000DB5 8B1D[A46A0000]        <1>        mov   ebx, [BUFFER_TAIL]      ; GET THE 'IN TO' POINTER TO THE BUFFER
00000DBB 89DE                  <1>  mov   esi, ebx        ; SAVE A COPY IN CASE BUFFER NOT FULL
00000DBD E8D3000000            <1>  call  _K4             ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
00000DC2 3B1D[A06A0000]        <1>  cmp   ebx, [BUFFER_HEAD]      ; WILL THE BUFFER OVERRUN IF WE STORE THIS?
00000DC8 740D                  <1>  je    short _K502     ; YES - INFORM CALLER OF ERROR
00000DCA 66890E                <1>  mov   [esi], cx       ; NO - PUT ASCII/SCAN CODE INTO BUFFER
00000DCD 891D[A46A0000]        <1>  mov   [BUFFER_TAIL], ebx      ; ADJUST 'IN TO' POINTER TO REFLECT CHANGE
00000DD3 28C0                  <1>  sub   al, al                  ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
00000DD5 EB02                  <1>  jmp   short _K504     ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
                               <1> _K502:
00000DD7 B001                  <1>  mov   al, 01h                 ; BUFFER FULL INDICATION
                               <1> _K504:
00000DD9 FB                    <1>  sti
00000DDA 5E                    <1>  pop   esi             ; RECOVER SI (esi)
00000DDB E93FFFFFFF            <1>         jmp     _KIO_EXIT               ; RETURN TO CALLER WITH STATUS IN AL
                               <1>
                               <1>  ;-----      READ THE KEY TO FIGURE OUT WHAT TO DO -----
                               <1> _K1S:
00000DE0 FA                    <1>  cli   ; 03/12/2014
00000DE1 8B1D[A06A0000]        <1>         mov     ebx, [BUFFER_HEAD]   ; GET POINTER TO HEAD OF BUFFER
00000DE7 3B1D[A46A0000]        <1>         cmp     ebx, [BUFFER_TAIL]   ; TEST END OF BUFFER
                               <1>  ;jne  short _K1U       ; IF ANYTHING IN BUFFER SKIP INTERRUPT
00000DED 750F                  <1>  jne   short _k1x ; 03/12/2014
                               <1>  ;
                               <1>  ; 03/12/2014
                               <1>  ; 28/08/2014
                               <1>  ; PERFORM OTHER FUNCTION ?? here !
                               <1>  ;; MOV      AX, 9002h         ; MOVE IN WAIT CODE & TYPE
                               <1>  ;; INT      15H               ; PERFORM OTHER FUNCTION
                               <1> _K1T:                                   ; ASCII READ
00000DEF FB                    <1>  sti                   ; INTERRUPTS BACK ON DURING LOOP
00000DF0 90                    <1>  nop                   ; ALLOW AN INTERRUPT TO OCCUR
                               <1> _K1U:
00000DF1 FA                    <1>  cli                   ; INTERRUPTS BACK OFF
00000DF2 8B1D[A06A0000]        <1>         mov        ebx, [BUFFER_HEAD]      ; GET POINTER TO HEAD OF BUFFER
00000DF8 3B1D[A46A0000]        <1>         cmp     ebx, [BUFFER_TAIL]   ; TEST END OF BUFFER
                               <1> _k1x:
00000DFE 53                    <1>  push  ebx             ; SAVE ADDRESS
00000DFF 9C                    <1>  pushf                 ; SAVE FLAGS
00000E00 E80B070000            <1>  call  MAKE_LED        ; GO GET MODE INDICATOR DATA BYTE
00000E05 8A1D[956A0000]        <1>  mov   bl, [KB_FLAG_2]   ; GET PREVIOUS BITS
00000E0B 30C3                  <1>  xor   bl, al                  ; SEE IF ANY DIFFERENT
00000E0D 80E307                <1>  and   bl, 07h     ; KB_LEDS   ; ISOLATE INDICATOR BITS
00000E10 7406                  <1>  jz    short _K1V      ; IF NO CHANGE BYPASS UPDATE
00000E12 E8A5060000            <1>  call  SND_LED1
00000E17 FA                    <1>  cli                   ; DISABLE INTERRUPTS
                               <1> _K1V:
00000E18 9D                    <1>  popf                  ; RESTORE FLAGS
00000E19 5B                    <1>  pop   ebx             ; RESTORE ADDRESS
00000E1A 74D3                  <1>         je      short _K1T              ; LOOP UNTIL SOMETHING IN BUFFER
                               <1>  ;
00000E1C 668B03                <1>  mov   ax, [ebx]       ; GET SCAN CODE AND ASCII CODE
00000E1F E871000000            <1>         call    _K4                     ; MOVE POINTER TO NEXT POSITION
00000E24 891D[A06A0000]        <1>         mov     [BUFFER_HEAD], ebx      ; STORE VALUE IN VARIABLE
00000E2A C3                    <1>  retn                  ; RETURN
                               <1>
                               <1>  ;-----      READ THE KEY TO SEE IF ONE IS PRESENT -----
                               <1> _K2S:
00000E2B FA                    <1>  cli                   ; INTERRUPTS OFF
00000E2C 8B1D[A06A0000]        <1>         mov     ebx, [BUFFER_HEAD]      ; GET HEAD POINTER
00000E32 3B1D[A46A0000]        <1>         cmp     ebx, [BUFFER_TAIL]      ; IF EQUAL (Z=1) THEN NOTHING THERE
00000E38 668B03                <1>  mov   ax, [ebx]
00000E3B 9C                    <1>  pushf                 ; SAVE FLAGS
```

```
00000E3C 6650                <1>  push  ax                ; SAVE CODE
00000E3E E8CD060000          <1>  call  MAKE_LED          ; GO GET MODE INDICATOR DATA BYTE
00000E43 8A1D[956A0000]      <1>  mov   bl, [KB_FLAG_2]   ; GET PREVIOUS BITS
00000E49 30C3                <1>  xor   bl, al                  ; SEE IF ANY DIFFERENT
00000E4B 80E307              <1>  and   bl, 07h ; KB_LEDS ; ISOLATE INDICATOR BITS
00000E4E 7405                <1>  jz    short _K2T        ; IF NO CHANGE BYPASS UPDATE
00000E50 E850060000          <1>  call  SND_LED                 ; GO TURN ON MODE INDICATORS
                             <1> _K2T:
00000E55 6658                <1>  pop   ax                ; RESTORE CODE
00000E57 9D                  <1>  popf                    ; RESTORE FLAGS
00000E58 FB                  <1>  sti                     ; INTERRUPTS BACK ON
00000E59 C3                  <1>  retn                    ; RETURN
                             <1>
                             <1>  ;-----      ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS -----
                             <1> _KIO_E_XLAT:
00000E5A 3CF0                <1>  cmp   al, 0F0h          ; IS IT ONE OF THE FILL-INs?
00000E5C 7506                <1>  jne   short _KIO_E_RET  ; NO, PASS IT ON
00000E5E 08E4                <1>         or   ah, ah                  ; AH = 0 IS SPECIAL CASE
00000E60 7402                <1>         jz   short _KIO_E_RET        ; PASS THIS ON UNCHANGED
00000E62 30C0                <1>  xor   al, al                  ; OTHERWISE SET AL = 0
                             <1> _KIO_E_RET:
00000E64 C3                  <1>  retn                    ; GO BACK
                             <1>
                             <1>  ;-----      ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS -----
                             <1> _KIO_S_XLAT:
00000E65 80FCE0              <1>  cmp   ah, 0E0h          ; IS IT KEYPAD ENTER OR / ?
00000E68 750F                <1>  jne   short _KIO_S2           ; NO, CONTINUE
00000E6A 3C0D                <1>  cmp   al, 0Dh                 ; KEYPAD ENTER CODE?
00000E6C 7408                <1>         je   short _KIO_S1          ; YES, MASSAGE A BIT
00000E6E 3C0A                <1>  cmp   al, 0Ah                 ; CTRL KEYPAD ENTER CODE?
00000E70 7404                <1>         je   short _KIO_S1          ; YES, MASSAGE THE SAME
00000E72 B435                <1>  mov   ah, 35h                 ; NO, MUST BE KEYPAD /
                             <1> _kio_ret: ; 03/12/2014
00000E74 F8                  <1>  clc
00000E75 C3                  <1>  retn
                             <1>  ;jmp  short _KIO_USE          ; GIVE TO CALLER
                             <1> _KIO_S1:
00000E76 B41C                <1>  mov   ah, 1Ch                 ; CONVERT TO COMPATIBLE OUTPUT
                             <1>  ;jmp  short _KIO_USE          ; GIVE TO CALLER
00000E78 C3                  <1>  retn
                             <1> _KIO_S2:
00000E79 80FC84              <1>  cmp   ah, 84h                 ; IS IT ONE OF EXTENDED ONES?
00000E7C 7715                <1>  ja    short _KIO_DIS          ; YES, THROW AWAY AND GET ANOTHER CHAR
00000E7E 3CF0                <1>  cmp   al, 0F0h          ; IS IT ONE OF THE FILL-INs?
00000E80 7506                <1>         jne  short _KIO_S3          ; NO, TRY LAST TEST
00000E82 08E4                <1>  or    ah, ah                  ; AH = 0 IS SPECIAL CASE
00000E84 740C                <1>         jz   short _KIO_USE         ; PASS THIS ON UNCHANGED
00000E86 EB0B                <1>  jmp   short _KIO_DIS          ; THROW AWAY THE REST
                             <1> _KIO_S3:
00000E88 3CE0                <1>  cmp   al, 0E0h          ; IS IT AN EXTENSION OF A PREVIOUS ONE?
                             <1>  ;jne  short _KIO_USE          ; NO, MUST BE A STANDARD CODE
00000E8A 75E8                <1>  jne   short _kio_ret
00000E8C 08E4                <1>  or    ah, ah                  ; AH = 0 IS SPECIAL CASE
00000E8E 7402                <1>         jz   short _KIO_USE         ; JUMP IF AH = 0
00000E90 30C0                <1>  xor   al, al                  ; CONVERT TO COMPATIBLE OUTPUT
                             <1>  ;jmp  short _KIO_USE          ; PASS IT ON TO CALLER
                             <1> _KIO_USE:
                             <1>  ;clc                    ; CLEAR CARRY TO INDICATE GOOD CODE
00000E92 C3                  <1>  retn                    ; RETURN
                             <1> _KIO_DIS:
00000E93 F9                  <1>  stc                     ; SET CARRY TO INDICATE DISCARD CODE
00000E94 C3                  <1>  retn                    ; RETURN
                             <1>
                             <1>  ;-----      INCREMENT BUFFER POINTER ROUTINE -----
                             <1> _K4:
00000E95 43                  <1>  inc     ebx
00000E96 43                  <1>  inc   ebx               ; MOVE TO NEXT WORD IN LIST
00000E97 3B1D[9C6A0000]      <1>         cmp     ebx, [BUFFER_END]    ; AT END OF BUFFER?
                             <1>         ;jne    short _K5               ; NO, CONTINUE
00000E9D 7206                <1>  jb    short _K5
00000E9F 8B1D[986A0000]      <1>         mov     ebx, [BUFFER_START]     ; YES, RESET TO BUFFER BEGINNING
                             <1> _K5:
00000EA5 C3                  <1>  retn
                             <1>
                             <1> ; 20/02/2015
                             <1> ; 05/12/2014
                             <1> ; 26/08/2014
                             <1> ; KEYBOARD (HARDWARE) INTERRUPT -  IRQ LEVEL 1
                             <1> ; (INT_09h - Retro UNIX 8086 v1 - U9.ASM, 07/03/2014)
                             <1> ;
                             <1> ; Derived from "KB_INT_1" procedure of IBM "pc-at"
                             <1> ; rombios source code (06/10/1985)
                             <1> ; 'keybd.asm', HARDWARE INT 09h - (IRQ Level 1)
                             <1>
                             <1> ;--------- 8042 COMMANDS -------------------------------------------------------
                             <1> ENA_KBD      equ   0AEh  ; ENABLE KEYBOARD COMMAND
                             <1> DIS_KBD      equ   0ADh  ; DISABLE KEYBOARD COMMAND
                             <1> SHUT_CMD     equ   0FEh  ; CAUSE A SHUTDOWN COMMAND
                             <1> ;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------
                             <1> STATUS_PORT  equ   064h  ; 8042 STATUS PORT
                             <1> INPT_BUF_FULLequ   00000010b ; 1 = +INPUT BUFFER FULL
                             <1> PORT_A       equ   060h  ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
```

```
                       <1> ;---------- 8042 KEYBOARD RESPONSE ---------------------------------------------
                       <1> KB_ACK       equ   0FAh  ; ACKNOWLEDGE PROM TRANSMISSION
                       <1> KB_RESEND    equ   0FEh  ; RESEND REQUEST
                       <1> KB_OVER_RUN  equ   0FFh  ; OVER RUN SCAN CODE
                       <1> ;---------- KEYBOARD/LED COMMANDS ----------------------------------------------
                       <1> KB_ENABLE    equ   0F4h        ; KEYBOARD ENABLE
                       <1> LED_CMD      equ   0EDh        ; LED WRITE COMMAND
                       <1> KB_TYPA_RD   equ   0F3h        ; TYPAMATIC RATE/DELAY COMMAND
                       <1> ;---------- KEYBOARD SCAN CODES ------------------------------------------------
                       <1> NUM_KEY      equ   69          ; SCAN CODE FOR    NUMBER LOCK KEY
                       <1> SCROLL_KEY   equ   70          ; SCAN CODE FOR    SCROLL LOCK KEY
                       <1> ALT_KEY      equ   56          ; SCAN CODE FOR    ALTERNATE SHIFT KEY
                       <1> CTL_KEY      equ   29          ; SCAN CODE FOR    CONTROL KEY
                       <1> CAPS_KEY     equ   58          ; SCAN CODE FOR    SHIFT LOCK KEY
                       <1> DEL_KEY      equ   83          ; SCAN CODE FOR    DELETE KEY
                       <1> INS_KEY      equ   82          ; SCAN CODE FOR    INSERT KEY
                       <1> LEFT_KEY     equ   42          ; SCAN CODE FOR    LEFT SHIFT
                       <1> RIGHT_KEY    equ   54          ; SCAN CODE FOR    RIGHT SHIFT
                       <1> SYS_KEY      equ   84          ; SCAN CODE FOR    SYSTEM KEY
                       <1> ;---------- ENHANCED KEYBOARD SCAN CODES ---------------------------------------
                       <1> ID_1         equ   0ABh        ; 1ST ID CHARACTER FOR KBX
                       <1> ID_2         equ   041h        ; 2ND ID CHARACTER FOR KBX
                       <1> ID_2A        equ   054h        ; ALTERNATE 2ND ID CHARACTER FOR KBX
                       <1> F11_M        equ   87          ; F11 KEY MAKE
                       <1> F12_M        equ   88          ; F12 KEY MAKE
                       <1> MC_E0        equ   224         ; GENERAL MARKER CODE
                       <1> MC_E1        equ   225         ; PAUSE KEY MARKER CODE
                       <1> ;---------- FLAG EQUATES WITHIN @KB_FLAG---------------------------------------
                       <1> RIGHT_SHIFT  equ   00000001b   ; RIGHT SHIFT KEY DEPRESSED
                       <1> LEFT_SHIFT   equ   00000010b   ; LEFT SHIFT KEY DEPRESSED
                       <1> CTL_SHIFT    equ   00000100b   ; CONTROL SHIFT KEY DEPRESSED
                       <1> ALT_SHIFT    equ   00001000b   ; ALTERNATE SHIFT KEY DEPRESSED
                       <1> SCROLL_STATE equ   00010000b   ; SCROLL LOCK STATE IS ACTIVE
                       <1> NUM_STATE    equ   00100000b   ; NUM LOCK STATE IS ACTIVE
                       <1> CAPS_STATE   equ   01000000b   ; CAPS LOCK STATE IS ACTIVE
                       <1> INS_STATE    equ   10000000b   ; INSERT STATE IS ACTIVE
                       <1> ;---------- FLAG EQUATES WITHIN      @KB_FLAG_1 --------------------------------
---
                       <1> L_CTL_SHIFT  equ   00000001b   ; LEFT CTL KEY DOWN
                       <1> L_ALT_SHIFT  equ   00000010b   ; LEFT ALT KEY DOWN
                       <1> SYS_SHIFT    equ   00000100b   ; SYSTEM KEY DEPRESSED AND HELD
                       <1> HOLD_STATE   equ   00001000b   ; SUSPEND KEY HAS BEEN TOGGLED
                       <1> SCROLL_SHIFT equ   00010000b   ; SCROLL LOCK KEY IS DEPRESSED
                       <1> NUM_SHIFT    equ   00100000b   ; NUM LOCK KEY IS DEPRESSED
                       <1> CAPS_SHIFT   equ   01000000b   ; CAPS LOCK KEY IS DEPRE55ED
                       <1> INS_SHIFT    equ   10000000b   ; INSERT KEY IS DEPRESSED
                       <1> ;---------- FLAGS EQUATES WITHIN @KB_FLAG_2 --------------------------------
                       <1> KB_LEDS      equ   00000111b   ; KEYBOARD LED STATE BITS
                       <1> ;      equ   00000001b   ; SCROLL LOCK INDICATOR
                       <1> ;      equ   00000010b   ; NUM LOCK INDICATOR
                       <1> ;      equ   00000100b   ; CAPS LOCK INDICATOR
                       <1> ;      equ   00001000b   ; RESERVED (MUST BE ZERO)
                       <1> KB_FA        equ   00010000b   ; ACKNOWLEDGMENT RECEIVED
                       <1> KB_FE        equ   00100000b   ; RESEND RECEIVED FLAG
                       <1> KB_PR_LED    equ   01000000b   ; MODE INDICATOR UPDATE
                       <1> KB_ERR       equ   10000000b   ; KEYBOARD TRANSMIT ERROR FLAG
                       <1> ;----------- FLAGS EQUATES WITHIN @KB_FLAG_3 ----------------------------------
                       <1> LC_E1        equ   00000001b   ; LAST CODE WAS THE E1 HIDDEN CODE
                       <1> LC_E0        equ   00000010b   ; LAST CODE WAS THE E0 HIDDEN CODE
                       <1> R_CTL_SHIFT  equ   00000100b   ; RIGHT CTL KEY DOWN
                       <1> R_ALT_SHIFT  equ   00001000b   ; RIGHT ALT KEY DOWN
                       <1> GRAPH_ON     equ   00001000b   ; ALT GRAPHICS KEY DOWN (WT ONLY)
                       <1> KBX          equ   00010000b   ; ENHANCED KEYBOARD INSTALLED
                       <1> SET_NUM_LK   equ   00100000b   ; FORCE NUM LOCK IF READ ID AND KBX
                       <1> LC_AB        equ   01000000b   ; LAST CHARACTER WAS FIRST ID CHARACTER
                       <1> RD_ID        equ   10000000b   ; DOING A READ ID (MUST BE BIT0)
                       <1> ;
                       <1> ;----------- INTERRUPT EQUATES ------------------------------------------------
                       <1> EOI          equ   020h        ; END OF INTERRUPT COMMAND TO 8259
                       <1> INTA00       equ   020h        ; 8259 PORT
                       <1>
                       <1>
                       <1> kb_int:
                       <1>
                       <1> ; 17/10/2015 ('ctrlbrk')
                       <1> ; 05/12/2014
                       <1> ; 04/12/2014 (derivation from pc-xt-286 bios source code -1986-,
                       <1>  ;           instead of pc-at bios - 1985-)
                       <1> ; 26/08/2014
                       <1> ;
                       <1> ; 03/06/86  KEYBOARD BIOS
                       <1> ;
                       <1> ;--- HARDWARE INT 09H -- (IRQ LEVEL 1) -------------------------------------------
                       <1> ;                                                      ;
                       <1> ;KEYBOARD INTERRUPT ROUTINE                                    ;
                       <1> ;                                                      ;
                       <1> ;--------------------------------------------------------------------------------
                       <1>
                       <1> KB_INT_1:
00000EA6 FB            <1>  sti                     ; ENABLE INTERRUPTS
                       <1>  ;push ebp
00000EA7 50            <1>  push  eax
```

```
00000EA8 53                   <1>  push  ebx
00000EA9 51                   <1>  push  ecx
00000EAA 52                   <1>  push  edx
00000EAB 56                   <1>  push  esi
00000EAC 57                   <1>  push  edi
00000EAD 1E                   <1>  push  ds
00000EAE 06                   <1>  push  es
00000EAF FC                   <1>  cld                     ; FORWARD DIRECTION
00000EB0 66B81000             <1>  mov   ax, KDATA
00000EB4 8ED8                 <1>  mov   ds, ax
00000EB6 8EC0                 <1>  mov   es, ax
                              <1>  ;
                              <1>  ;-----      WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
00000EB8 B0AD                 <1>  mov   al, DIS_KBD       ; DISABLE THE KEYBOARD COMMAND
00000EBA E885050000           <1>  call  SHIP_IT                 ; EXECUTE DISABLE
00000EBF FA                   <1>  cli                     ; DISABLE INTERRUPTS
00000EC0 B900000100           <1>  mov   ecx, 10000h       ; SET MAXIMUM TIMEOUT
                              <1> KB_INT_01:
00000EC5 E464                 <1>  in    al, STATUS_PORT         ; READ ADAPTER STATUS
00000EC7 A802                 <1>  test  al, INPT_BUF_FULL ; CHECK INPUT BUFFER FULL STATUS BIT
00000EC9 E0FA                 <1>  loopnz      KB_INT_01         ; WAIT FOR COMMAND TO BE ACCEPTED
                              <1>  ;
                              <1>  ;-----      READ CHARACTER FROM KEYBOARD INTERFACE
00000ECB E460                 <1>  in    al, PORT_A        ; READ IN THE CHARACTER
                              <1>  ;
                              <1>  ;-----      SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INT LEVEL 9H)
                              <1>  ;MOV  AH, 04FH          ; SYSTEM INTERCEPT - KEY CODE FUNCTION
                              <1>  ;STC                    ; SET CY=1 (IN CASE OF IRET)
                              <1>  ;INT  15H               ; CASETTE CALL (AL)=KEY SCAN CODE
                              <1>  ;                       ; RETURNS CY=1 FOR INVALID FUNCTION
                              <1>  ;JC   KB_INT_02         ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
                              <1>  ;JMP  K26               ; EXIT IF SYSTEM HANDLES SCAN CODE
                              <1>  ;                       ; EX¦T HANDLES HARDWARE EOI AND ENABLE
                              <1>  ;
                              <1>  ;-----      CHECK FOR A RESEND COMMAND TO KEYBOARD
                              <1> KB_INT_02:                    ;       (AL)= SCAN CODE
00000ECD FB                   <1>  sti                     ; ENABLE INTERRUPTS AGAIN
00000ECE 3CFE                 <1>  cmp   al, KB_RESEND           ; IS THE INPUT A RESEND
00000ED0 7411                 <1>         je      short KB_INT_4          ; GO IF RESEND
                              <1>  ;
                              <1>  ;-----      CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
00000ED2 3CFA                 <1>  cmp   al, KB_ACK        ; IS THE INPUT AN ACKNOWLEDGE
00000ED4 751A                 <1>         jne     short KB_INT_2          ; GO IF NOT
                              <1>  ;
                              <1>  ;-----      A COMMAND TO THE KEYBOARD WAS ISSUED
00000ED6 FA                   <1>  cli                     ; DISABLE INTERRUPTS
00000ED7 800D[956A0000]10     <1>  or    byte [KB_FLAG_2], KB_FA ; INDICATE ACK RECEIVED
00000EDE E97A020000           <1>         jmp     K26                     ; RETURN IF NOT (ACK RETURNED FOR DATA)
                              <1>  ;
                              <1>  ;-----      RESEND THE LAST BYTE
                              <1> KB_INT_4:
00000EE3 FA                   <1>  cli                     ; DISABLE INTERRUPTS
00000EE4 800D[956A0000]20     <1>  or    byte [KB_FLAG_2], KB_FE ; INDICATE RESEND RECEIVED
00000EEB E96D020000           <1>         jmp     K26                     ; RETURN IF NOT ACK RETURNED FOR DATA)
                              <1>  ;
                              <1> ;----- UPDATE MODE INDICATORS IF CHANGE IN STATE
                              <1> KB_INT_2:
00000EF0 6650                 <1>  push  ax                ; SAVE DATA IN
00000EF2 E819060000           <1>  call  MAKE_LED          ; GO GET MODE INDICATOR DATA BYTE
00000EF7 8A1D[956A0000]       <1>  mov   bl, [KB_FLAG_2]   ; GET PREVIOUS BITS
00000EFD 30C3                 <1>  xor   bl, al                  ; SEE IF ANY DIFFERENT
00000EFF 80E307               <1>  and   bl, KB_LEDS       ; ISOLATE INDICATOR BITS
00000F02 7405                 <1>  jz    short UP0         ; IF NO CHANGE BYPASS UPDATE
00000F04 E89C050000           <1>  call  SND_LED                 ; GO TURN ON MODE INDICATORS
                              <1> UP0:
00000F09 6658                 <1>  pop   ax                ; RESTORE DATA IN
                              <1> ;---------------------------------------------------------------------
                              <1> ;START OF KEY PROCESSING                                   ;
                              <1> ;---------------------------------------------------------------------
00000F0B 88C4                 <1>  mov   ah, al                  ; SAVE SCAN CODE IN AH ALSO
                              <1>  ;
                              <1>  ;-----      TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
00000F0D 3CFF                 <1>  cmp   al, KB_OVER_RUN         ; IS THIS AN OVERRUN CHAR
00000F0F 0F841B050000         <1>         je      K62                  ; BUFFER_FULL_BEEP
                              <1>  ;
                              <1> K16:
00000F15 8A3D[966A0000]       <1>  mov   bh, [KB_FLAG_3]         ; LOAD FLAGS FOR TESTING
                              <1>  ;
                              <1>  ;-----      TEST TO SEE IF A READ_ID IS IN PROGRESS
00000F1B F6C7C0               <1>  test  bh, RD_ID+LC_AB   ; ARE WE DOING A READ ID?
00000F1E 7449                 <1>  jz    short NOT_ID            ; CONTINUE IF NOT
00000F20 7917                 <1>  jns   short TST_ID_2          ; IS THE RD_ID FLAG ON?
00000F22 3CAB                 <1>  cmp   al, ID_1          ; IS THIS THE 1ST ID CHARACTER?
00000F24 7507                 <1>  jne   short RST_RD_ID
00000F26 800D[966A0000]40     <1>  or    byte [KB_FLAG_3], LC_AB ; INDICATE 1ST ID WAS OK
                              <1> RST_RD_ID:
00000F2D 8025[966A0000]7F     <1>  and   byte [KB_FLAG_3], ~RD_ID ; RESET THE READ ID FLAG
                              <1>         ;jmp    short ID_EX          ; AND EXIT
00000F34 E924020000           <1>  jmp   K26
                              <1>  ;
                              <1> TST_ID_2:
00000F39 8025[966A0000]BF     <1>  and   byte [KB_FLAG_3], ~LC_AB ; RESET FLAG
```

```
00000F40 3C54                <1>  cmp   al, ID_2A         ; IS THIS THE 2ND ID CHARACTER?
00000F42 7419                <1>         je   short KX_BIT            ; JUMP IF SO
00000F44 3C41                <1>  cmp   al, ID_2          ; IS THIS THE 2ND ID CHARACTER?
                             <1>         ;jne short ID_EX       ; LEAVE IF NOT
00000F46 0F8511020000        <1>  jne   K26
                             <1>  ;
                             <1>  ;-----      A READ ID SAID THAT IT WAS ENHANCED KEYBOARD
00000F4C F6C720              <1>  test  bh, SET_NUM_LK          ; SHOULD WE SET NUM LOCK?
00000F4F 740C                <1>         jz      short KX_BIT         ; EXIT IF NOT
00000F51 800D[936A0000]20    <1>  or    byte [KB_FLAG], NUM_STATE ; FORCE NUM LOCK ON
00000F58 E848050000          <1>  call  SND_LED                 ; GO SET THE NUM LOCK INDICATOR
                             <1> KX_BIT:
00000F5D 800D[966A0000]10    <1>  or    byte [KB_FLAG_3], KBX   ; INDICATE ENHANCED KEYBOARD WAS FOUND
00000F64 E9F4010000          <1> ID_EX: jmp     K26             ; EXIT
                             <1>  ;
                             <1> NOT_ID:
00000F69 3CE0                <1>  cmp   al, MC_E0         ; IS THIS THE GENERAL MARKER CODE?
00000F6B 750C                <1>  jne   short TEST_E1
00000F6D 800D[966A0000]12    <1>  or    byte [KB_FLAG_3], LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
                             <1>  ;jmp  short EXIT        ; THROW AWAY THIS CODE
00000F74 E9EB010000          <1>  jmp   K26A
                             <1> TEST_E1:
00000F79 3CE1                <1>  cmp   al, MC_E1         ; IS THIS THE PAUSE KEY?
00000F7B 750C                <1>  jne   short NOT_HC
00000F7D 800D[966A0000]11    <1>  or    byte [KB_FLAG_3], LC_E1+KBX ; SET FLAG BIT, SET KBX, AND
00000F84 E9DB010000          <1> EXIT:  jmp   K26A              ; THROW AWAY THIS CODE
                             <1>  ;
                             <1> NOT_HC:
00000F89 247F                <1>  and   al, 07Fh          ; TURN OFF THE BREAK BIT
00000F8B F6C702              <1>  test  bh, LC_E0         ; LAST CODE THE E0 MARKER CODE
00000F8E 7414                <1>  jz    short NOT_LC_E0         ; JUMP IF NOT
                             <1>  ;
00000F90 BF[7E690000]        <1>  mov   edi, _K6+6        ; IS THIS A SHIFT KEY?
00000F95 AE                  <1>  scasb
00000F96 0F84C1010000        <1>         je      K26 ; K16B              ; YES, THROW AWAY & RESET FLAG
00000F9C AE                  <1>  scasb
00000F9D 757C                <1>  jne   short K16A        ; NO, CONTINUE KEY PROCESSING
                             <1>  ;jmp  short K16B        ; YES, THROW AWAY & RESET FLAG
00000F9F E9B9010000          <1>  jmp   K26
                             <1>  ;
                             <1> NOT_LC_E0:
00000FA4 F6C701              <1>  test  bh, LC_E1         ; LAST CODE THE E1 MARKER CODE?
00000FA7 7435                <1>  jz    short T_SYS_KEY         ; JUMP IF NOT
00000FA9 B904000000          <1>  mov   ecx, 4                  ; LENGHT OF SEARCH
00000FAE BF[7C690000]        <1>  mov   edi, _K6+4        ; IS THIS AN ALT, CTL, OR SHIFT?
00000FB3 F2AE                <1>  repne scasb             ; CHECK IT
                             <1>  ;je   short EXIT        ; THROW AWAY IF SO
00000FB5 0F84A9010000        <1>  je    K26A
                             <1>  ;
00000FBB 3C45                <1>  cmp   al, NUM_KEY       ; IS IT THE PAUSE KEY?
                             <1>  ;jne  short K16B        ; NO, THROW AWAY & RESET FLAG
00000FBD 0F859A010000        <1>  jne   K26
00000FC3 F6C480              <1>  test  ah, 80h                 ; YES, IS IT THE BREAK OF THE KEY?
                             <1>  ;jnz  short K16B        ; YES, THROW THIS AWAY, TOO
00000FC6 0F8591010000        <1>  jnz   K26
                             <1>         ; 20/02/2015
00000FCC F605[946A0000]08    <1>  test  byte [KB_FLAG_1],HOLD_STATE ;  NO, ARE WE PAUSED ALREADY?
                             <1>  ;jnz  short K16B        ;  YES, THROW AWAY
00000FD3 0F8584010000        <1>  jnz   K26
00000FD9 E9E1020000          <1>  jmp     K39P                    ; NO, THIS IS THE REAL PAUSE STATE
                             <1>  ;
                             <1>  ;-----      TEST FOR SYSTEM KEY
                             <1> T_SYS_KEY:
00000FDE 3C54                <1>  cmp   al, SYS_KEY       ; IS IT THE SYSTEM KEY?
00000FE0 7539                <1>  jnz   short K16A        ; CONTINUE IF NOT
                             <1>  ;
00000FE2 F6C480              <1>  test  ah, 80h                 ; CHECK IF THIS A BREAK CODE
00000FE5 7524                <1>  jnz   short K16C        ; DO NOT TOUCH SYSTEM INDICATOR IF TRUE
                             <1>  ;
00000FE7 F605[946A0000]04    <1>  test  byte [KB_FLAG_1], SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
                             <1>         ;jnz short K16B        ; IF YES, DO NOT PROCESS SYSTEM INDICATOR
00000FEE 0F8569010000        <1>  jnz     K26
                             <1>  ;
00000FF4 800D[946A0000]04    <1>  or    byte [KB_FLAG_1], SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
00000FFB B020                <1>  mov   al, EOI                 ; END OF INTERRUPT COMMAND
00000FFD E620                <1>  out   20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
                             <1>                          ; INTERRUPT-RETURN-NO-EOI
00000FFF B0AE                <1>  mov   al, ENA_KBD       ; INSURE KEYBOARD IS ENABLED
00001001 E83E040000          <1>  call  SHIP_IT                 ; EXECUTE ENABLE
                             <1>  ; !!! SYSREQ !!! function/system call (INTERRUPT) must be here !!!
                             <1>  ;MOV  AL, 8500H         ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
                             <1>  ;STI                    ; MAKE SURE INTERRUPTS ENABLED
                             <1>  ;INT  15H               ; USER INTERRUPT
00001006 E965010000          <1>         jmp     K27A                    ; END PROCESSING
                             <1>  ;
                             <1> ;K16B: jmp   K26               ; IGNORE SYSTEM KEY
                             <1>  ;
                             <1> K16C:
0000100B 8025[946A0000]FB    <1>  and   byte [KB_FLAG_1], ~SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
00001012 B020                <1>  mov   al, EOI                 ; END OF INTERRUPT COMMAND
00001014 E620                <1>  out   20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
                             <1>                          ; INTERRUPT-RETURN-NO-EOI
                             <1>  ;MOV  AL, ENA_KBD       ; INSURE KEYBOARD IS ENABLED
                             <1>  ;CALL SHIP_IT                 ; EXECUTE ENABLE
                             <1>  ;
                             <1>  ;MOV  AX, 8501H         ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
                             <1>  ;STI                    ; MAKE SURE INTERRUPTS ENABLED
                             <1>  ;INT  15H               ; USER INTERRUPT
                             <1>  ;JMP  K27A              ; INGONRE SYSTEM KEY
                             <1>  ;
00001016 E94E010000          <1>  jmp     K27             ; IGNORE SYSTEM KEY
                             <1>  ;
                             <1>  ;-----      TEST FOR SHIFT KEYS
                             <1> K16A:
0000101B 8A1D[936A0000]      <1>  mov   bl, [KB_FLAG]           ; PUT STATE FLAGS IN BL
00001021 BF[78690000]        <1>  mov   edi, _K6          ; SHIFT KEY TABLE offset
00001026 B908000000          <1>  mov   ecx, _K6L         ; LENGTH
0000102B F2AE                <1>  repne scasb             ; LOOK THROUGH THE TABLE FOR A MATCH
0000102D 88E0                <1>  mov   al, ah                  ; RECOVER SCAN CODE
0000102F 0F8510010000        <1>         jne     K25                      ; IF NO MATCH, THEN SHIFT NOT FOUND
                             <1>  ;
                             <1>  ;------     SHIFT KEY FOUND
                             <1> K17:
00001035 81EF[79690000]      <1>         sub     edi, _K6+1              ; ADJUST PTR TO SCAN CODE MATCH
0000103B 8AA7[80690000]      <1>        mov     ah, [edi+_K7]        ; GET MASK INTO AH
00001041 B102                <1>  mov   cl, 2             ; SETUP COUNT FOR FLAG SHIFTS
00001043 A880                <1>  test  al, 80h                 ; TEST FOR BREAK KEY
00001045 0F8596000000        <1>         jnz     K23                      ; JUMP OF BREAK
                             <1>  ;
                             <1>  ;-----      SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
                             <1> K17C:
0000104B 80FC10              <1>  cmp   ah, SCROLL_SHIFT
0000104E 732B                <1>  jae   short K18         ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
                             <1>  ;
                             <1>  ;-----      PLAIN SHIFT KEY, SET SHIFT ON
00001050 0825[936A0000]      <1>  or    [KB_FLAG], ah           ; TURN ON SHIFT BIT
00001056 A80C                <1>         test al, CTL_SHIFT+ALT_SHIFT ; IS IT ALT OR CTRL?
                             <1>  ;jnz  short K17D        ; YES, MORE FLAGS TO SET
00001058 0F84FF000000        <1>  jz    K26               ; NO, INTERRUPT RETURN
                             <1> K17D:
0000105E F6C702              <1>  test  bh, LC_E0         ; IS THIS ONE OF NEW KEYS?
00001061 740B                <1>  jz    short K17E        ; NO, JUMP
00001063 0825[966A0000]      <1>  or    [KB_FLAG_3], ah         ; SET BITS FOR RIGHT CTRL, ALT
00001069 E9EF000000          <1>  jmp   K26               ; INTERRUPT RETURN
                             <1> K17E:
0000106E D2EC                <1>  shr   ah, cl                  ; MOVE FLAG BITS TWO POSITIONS
00001070 0825[946A0000]      <1>  or    [KB_FLAG_1], ah         ; SET BITS FOR LEFT CTRL, ALT
00001076 E9E2000000          <1>  jmp   K26
                             <1>  ;
                             <1>  ;-----      TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
                             <1> K18:                           ; SHIFT-TOGGLE
0000107B F6C304              <1>  test  bl, CTL_SHIFT           ; CHECK CTL SHIFT STATE
                             <1>        ;jz        short K18A              ; JUMP IF NOT CTL STATE
0000107E 0F85C1000000        <1>         jnz     K25                      ; JUMP IF CTL STATE
                             <1> K18A:
00001084 3C52                <1>  cmp   al, INS_KEY       ; CHECK FOR INSERT KEY
00001086 7524                <1>  jne   short K22         ; JUMP IF NOT INSERT KEY
00001088 F6C308              <1>  test  bl, ALT_SHIFT           ; CHECK FOR ALTERNATE SHIFT
                             <1>        ;jz   short K18B        ; JUMP IF NOT ALTERNATE SHIFT
0000108B 0F85B4000000        <1>         jnz     K25                      ; JUMP IF ALTERNATE SHIFT
                             <1> K18B:
00001091 F6C702              <1>  test  bh, LC_E0 ;20/02/2015   ; IS THIS NEW INSERT KEY?
00001094 7516                <1>  jnz   short K22         ; YES, THIS ONE'S NEVER A '0'
                             <1> K19:
00001096 F6C320              <1>  test  bl, NUM_STATE           ; CHECK FOR BASE STATE
00001099 750C                <1>  jnz   short K21         ; JUMP IF NUM LOCK IS ON
0000109B F6C303              <1>  test  bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
0000109E 740C                <1>  jz    short K22         ; JUMP IF BASE STATE
                             <1> K20:                           ; NUMERIC ZERO, NOT INSERT KEY
000010A0 88C4                <1>  mov   ah, al                  ; PUT SCAN CODE BACK IN AH
000010A2 E99E000000          <1>         jmp     K25                      ; NUMERAL '0', STNDRD. PROCESSING
                             <1> K21:                           ; MIGHT BE NUMERIC
000010A7 F6C303              <1>  test  bl, LEFT_SHIFT+RIGHT_SHIFT
000010AA 74F4                <1>  jz    short K20         ; IS NUMERIC, STD. PROC.
                             <1>  ;
                             <1> K22:                           ; SHIFT TOGGLE KEY HIT; PROCESS IT
000010AC 8425[946A0000]      <1>  test  ah, [KB_FLAG_1]   ; IS KEY ALREADY DEPRESSED
000010B2 0F85A5000000        <1>         jnz     K26                      ; JUMP IF KEY ALREADY DEPRESSED
                             <1> K22A:
000010B8 0825[946A0000]      <1>         or      [KB_FLAG_1], ah      ; INDICATE THAT THE KEY IS DEPRESSED
000010BE 3025[936A0000]      <1>  xor   [KB_FLAG], ah           ; TOGGLE THE SHIFT STATE
                             <1>  ;
                             <1>  ;-----      TOGGLE LED IF CAPS, NUM  OR SCROLL KEY DEPRESSED
000010C4 F6C470              <1>  test  ah, CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
000010C7 7409                <1>  jz    short K22B        ; GO IF NOT
                             <1>  ;
000010C9 6650                <1>  push  ax                ; SAVE SCAN CODE AND SHIFT MASK
000010CB E8D5030000          <1>  call  SND_LED                 ; GO TURN MODE INDICATORS ON
000010D0 6658                <1>  pop   ax                ; RESTORE SCAN CODE
                             <1> K22B:
000010D2 3C52                <1>  cmp   al, INS_KEY       ; TEST FOR 1ST MAKE OF INSERT KEY
000010D4 0F8583000000        <1>         jne     K26                      ; JUMP IF NOT INSERT KEY
000010DA 88C4                <1>  mov   ah, al                    ; SCAN CODE IN BOTH HALVES OF AX
000010DC E999000000          <1>         jmp     K28                      ; FLAGS UPDATED, PROC. FOR BUFFER
```

```
                             <1>  ;
                             <1>  ;-----      BREAK SHIFT FOUND
                             <1> K23:                           ; BREAK-SHIFT-FOUND
000010E1 80FC10              <1>  cmp   ah, SCROLL_SHIFT  ; IS THIS A TOGGLE KEY
000010E4 F6D4                <1>  not   ah                ; INVERT MASK
000010E6 7355                <1>  jae   short K24         ; YES, HANDLE BREAK TOGGLE
000010E8 2025[936A0000]      <1>  and   [KB_FLAG], ah           ; TURN OFF SHIFT BIT
000010EE 80FCFB              <1>  cmp   ah, ~CTL_SHIFT          ; IS THIS ALT OR CTL?
000010F1 7730                <1>  ja    short K23D        ; NO, ALL DONE
                             <1>  ;
000010F3 F6C702              <1>  test  bh, LC_E0         ; 2ND ALT OR CTL?
000010F6 7408                <1>  jz    short K23A        ; NO, HANSLE NORMALLY
000010F8 2025[966A0000]      <1>  and   [KB_FLAG_3], ah         ; RESET BIT FOR RIGHT ALT OR CTL
000010FE EB08                <1>  jmp   short K23B        ; CONTINUE
                             <1> K23A:
00001100 D2FC                <1>  sar   ah, cl                  ; MOVE THE MASK BIT TWO POSITIONS
00001102 2025[946A0000]      <1>  and   [KB_FLAG_1], ah         ; RESET BIT FOR LEFT ALT AND CTL
                             <1> K23B:
00001108 88C4                <1>  mov   ah, al                  ; SAVE SCAN CODE
0000110A A0[966A0000]        <1>  mov   al, [KB_FLAG_3]         ; GET RIGHT ALT & CTRL FLAGS
0000110F D2E8                <1>  shr   al, cl                  ; MOVE TO BITS 1 & 0
00001111 0A05[946A0000]      <1>  or    al, [KB_FLAG_1]         ; PUT IN LEFT ALÌT & CTL FLAGS
00001117 D2E0                <1>  shl   al, cl                  ; MOVE BACK TO BITS 3 & 2
00001119 240C                <1>  and   al, ALT_SHIFT+CTL_SHIFT ; FILTER OUT OTHER GARBAGE
0000111B 0805[936A0000]      <1>  or    [KB_FLAG], al           ; PUT RESULT IN THE REAL FLAGS
00001121 88E0                <1>  mov   al, ah
                             <1> K23D:
00001123 3CB8                <1>  cmp   al, ALT_KEY+80h         ; IS THIS ALTERNATE SHIFT RELEASE
00001125 7536                <1>  jne   short K26         ; INTERRUPT RETURN
                             <1>  ;
                             <1>  ;-----      ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
00001127 A0[976A0000]        <1>  mov   al, [ALT_INPUT]
0000112C B400                <1>  mov   ah, 0             ; SCAN CODE OF 0
0000112E 8825[976A0000]      <1>  mov   [ALT_INPUT], ah   ; ZERO OUT THE FIELD
00001134 3C00                <1>  cmp   al, 0             ; WAS THE INPUT = 0?
00001136 7425                <1>  je    short K26         ; INTERRUPT_RETURN
00001138 E9D0020000          <1>         jmp     K61                     ; IT WASN'T, SO PUT IN BUFFER
                             <1>  ;
                             <1> K24:                           ; BREAK-TOGGLE
0000113D 2025[946A0000]      <1>  and   [KB_FLAG_1], ah   ; INDICATE NO LONGER DEPRESSED
00001143 EB18                <1>  jmp   short K26         ; INTERRUPT_RETURN
                             <1>  ;
                             <1>  ;-----      TEST FOR HOLD STATE
                             <1>                          ; AL, AH = SCAN CODE
                             <1> K25:                           ; NO-SHIFT-FOUND
00001145 3C80                <1>  cmp   al, 80h                 ; TEST FOR BREAK KEY
00001147 7314                <1>  jae   short K26         ; NOTHING FOR BREAK CHARS FROM HERE ON
00001149 F605[946A0000]08    <1>  test  byte [KB_FLAG_1], HOLD_STATE ; ARE WE IN HOLD STATE
00001150 7428                <1>  jz    short K28         ; BRANCH AROUND TEST IF NOT
00001152 3C45                <1>  cmp   al, NUM_KEY
00001154 7407                <1>  je    short K26         ; CAN'T END HOLD ON NUM_LOCK
00001156 8025[946A0000]F7    <1>  and   byte [KB_FLAG_1], ~HOLD_STATE ; TURN OFF THE HOLD STATE BIT
                             <1>  ;
                             <1> K26:
0000115D 8025[966A0000]FC    <1>  and   byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
                             <1> K26A:                          ; INTERRUPT-RETURN
00001164 FA                  <1>  cli                     ; TURN OFF INTERRUPTS
00001165 B020                <1>  mov   al, EOI                 ; END OF INTERRUPT COMMAND
00001167 E620                <1>  out   20h, al     ;out INTA00, al   ; SEND COMMAND TO INTERRUPT CONTROL PORT
                             <1> K27:                           ; INTERRUPT-RETURN-NO-EOI
00001169 B0AE                <1>  mov   al, ENA_KBD       ; INSURE KEYBOARD IS ENABLED
0000116B E8D4020000          <1>  call  SHIP_IT                 ; EXECUTE ENABLE
                             <1> K27A:
00001170 FA                  <1>  cli                     ; DISABLE INTERRUPTS
00001171 07                  <1>  pop   es                ; RESTORE REGISTERS
00001172 1F                  <1>  pop   ds
00001173 5F                  <1>  pop   edi
00001174 5E                  <1>  pop   esi
00001175 5A                  <1>  pop   edx
00001176 59                  <1>  pop   ecx
00001177 5B                  <1>  pop   ebx
00001178 58                  <1>  pop   eax
                             <1>  ;pop  ebp
00001179 CF                  <1>  iret                    ; RETURN
                             <1>
                             <1>  ;-----      NOT IN      HOLD STATE
                             <1> K28:                           ; NO-HOLD-STATE
0000117A 3C58                <1>  cmp   al, 88                  ; TEST FOR OUT-OF-RANGE SCAN CODES
0000117C 77DF                <1>  ja    short K26         ; IGNORE IF OUT-OF-RANGE
                             <1>  ;
0000117E F6C308              <1>  test  bl, ALT_SHIFT           ; ARE WE IN ALTERNATE SHIFT
                             <1>        ;jz  short K28A         ; IF NOT ALTERNATE
00001181 0F84F1000000        <1>         jz      K38
                             <1>  ;
00001187 F6C710              <1>  test  bh, KBX                 ; IS THIS THE ENCHANCED KEYBOARD?
0000118A 740D                <1>  jz    short K29         ; NO, ALT STATE IS REAL
                             <1>   ;28/02/2015
0000118C F605[946A0000]04    <1>  test  byte [KB_FLAG_1], SYS_SHIFT ; YES, IS SYSREQ KEY DOWN?
                             <1>  ;jz   short K29         ;  NO, ALT STATE IS REAL
00001193 0F85DF000000        <1>  jnz   K38               ; YES, THIS IS PHONY ALT STATE
                             <1>         ;                      ; DUE TO PRESSING SYSREQ
                             <1> ;K28A: jmp   short K38
                             <1>  ;
                             <1>  ;-----      TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
                             <1> K29:                           ; TEST-RESET
00001199 F6C304              <1>  test  bl, CTL_SHIFT           ; ARE WE IN CONTROL SHIFT ALSO?
0000119C 740B                <1>  jz    short K31         ; NO_RESET
0000119E 3C53                <1>  cmp   al, DEL_KEY       ; CTL-ALT STATE, TEST FOR DELETE KEY
000011A0 7507                <1>  jne   short K31         ; NO_RESET, IGNORE
                             <1>  ;
                             <1>  ;-----      CTL-ALT-DEL HAS BEEN FOUND
                             <1>  ; 26/08/2014
                             <1> cpu_reset:
                             <1>  ; IBM PC/AT ROM BIOS source code - 10/06/85 (TEST4.ASM - PROC_SHUTDOWN)
                             <1>  ; Send FEh (system reset command) to the keyboard controller.
000011A2 B0FE                <1>  mov   al, SHUT_CMD            ; SHUTDOWN COMMAND
000011A4 E664                <1>  out   STATUS_PORT, al         ; SEND TO KEYBOARD CONTROL PORT
                             <1> khere:
000011A6 F4                  <1>  hlt                     ; WAIT FOR 80286 RESET
000011A7 EBFD                <1>  jmp   short khere       ; INSURE HALT
                             <1>
                             <1>  ;
                             <1>  ;-----      IN ALTERNATE SHIFT, RESET NOT FOUND
                             <1> K31:                           ; NO-RESET
000011A9 3C39                <1>  cmp   al, 57                  ; TEST FOR SPACE KEY
000011AB 7507                <1>  jne   short K311        ; NOT THERE
000011AD B020                <1>  mov   al, ' '                 ; SET SPACE CHAR
000011AF E948020000          <1>        jmp     K57                     ; BUFFER_FILL
                             <1> K311:
000011B4 3C0F                <1>  cmp   al, 15                  ; TEST FOR TAB KEY
000011B6 7509                <1>  jne   short K312        ; NOT THERE
000011B8 66B800A5            <1>  mov   ax, 0A500h        ; SET SPECIAL CODE FOR ALT-TAB
000011BC E93B020000          <1>        jmp     K57                     ; BUFFER_FILL
                             <1> K312:
000011C1 3C4A                <1>  cmp   al, 74                  ; TEST FOR KEY PAD -
000011C3 0F84A2000000        <1>        je      K37B                    ; GO PROCESS
000011C9 3C4E                <1>  cmp   al, 78                  ; TEST FOR KEY PAD +
000011CB 0F849A000000        <1>        je      K37B                    ; GO PROCESS
                             <1>  ;
                             <1>  ;-----      LOOK FOR KEY PAD ENTRY
                             <1> K32:                           ; ALT-KEY-PAD
000011D1 BF[54690000]        <1>  mov   edi, K30          ; ALT-INPUT-TABLE offset
000011D6 B90A000000          <1>  mov   ecx, 10                 ; LOOK FOR ENTRY USING KEYPAD
000011DB F2AE                <1>  repne scasb             ; LOOK FOR MATCH
000011DD 7525                <1>  jne   short K33         ; NO_ALT_KEYPAD
000011DF F6C702              <1>  test  bh, LC_E0         ; IS THIS ONE OF THE NEW KEYS?
000011E2 0F858A000000        <1>        jnz     K37C                    ; YES, JUMP, NOT NUMPAD KEY
000011E8 81EF[55690000]      <1>  sub   edi, K30+1        ; DI NOW HAS ENTRY VALUE
000011EE A0[976A0000]        <1>  mov   al, [ALT_INPUT]   ; GET THE CURRENT BYTE
000011F3 B40A                <1>  mov   ah, 10                  ; MULTIPLY BY 10
000011F5 F6E4                <1>  mul   ah
000011F7 6601F8              <1>  add   ax, di                  ; ADD IN THE LATEST ENTRY
000011FA A2[976A0000]        <1>  mov   [ALT_INPUT], al   ; STORE IT AWAY
                             <1> ;K32A:
000011FF E959FFFFFF          <1>        jmp     K26                     ; THROW AWAY THAT KEYSTROKE
                             <1>  ;
                             <1>  ;-----      LOOK FOR SUPERSHIFT ENTRY
                             <1> K33:                           ; NO-ALT-KEYPAD
00001204 C605[976A0000]00    <1>        mov     byte [ALT_INPUT], 0     ; ZERO ANY PREVIOUS ENTRY INTO INPUT
0000120B B91A000000          <1>  mov   ecx, 26                 ; (DI),(ES) ALREADY POINTING
00001210 F2AE                <1>  repne scasb             ; LOOK FOR MATCH IN ALPHABET
00001212 7450                <1>  je    short K37A        ; MATCH FOUND, GO FILLL THE BUFFER
                             <1>  ;
                             <1>  ;-----      LOOK FOR TOP ROW OF ALTERNATE SHIFT
                             <1> K34:                           ; ALT-TOP-ROW
00001214 3C02                <1>  cmp   al, 2             ; KEY WITH '1' ON IT
00001216 7253                <1>  jb    short K37B        ; MUST BE ESCAPE
00001218 3C0D                <1>  cmp   al, 13                  ; IS IT IN THE REGION
0000121A 7705                <1>  ja    short K35         ; NO, ALT SOMETHING ELSE
0000121C 80C476              <1>  add   ah, 118                 ; CONVERT PSEUDO SCAN CODE TO RANGE
0000121F EB43                <1>  jmp   short K37A        ; GO FILL THE BUFFER
                             <1>  ;
                             <1>  ;-----      TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
                             <1> K35:                           ; ALT-FUNCTION
00001221 3C57                <1>  cmp   al, F11_M         ; IS IT F11?
00001223 7209                <1>  jb    short K35A ; 20/02/2015 ; NO, BRANCH
00001225 3C58                <1>  cmp   al, F12_M         ; IS IT F12?
00001227 7705                <1>  ja    short K35A ; 20/02/2015 ; NO, BRANCH
00001229 80C434              <1>  add   ah, 52                  ; CONVERT TO PSEUDO SCAN CODE
0000122C EB36                <1>  jmp   short K37A        ; GO FILL THE BUFFER
                             <1> K35A:
0000122E F6C702              <1>  test  bh, LC_E0         ; DO WE HAVE ONE OF THE NEW KEYS?
00001231 7422                <1>  jz    short K37         ; NO, JUMP
00001233 3C1C                <1>  cmp   al, 28                  ; TEST FOR KEYPAD ENTER
00001235 7509                <1>        jne     short K35B              ; NOT THERE
00001237 66B800A6            <1>  mov   ax, 0A600h        ; SPECIAL CODE
0000123B E9BC010000          <1>  jmp   K57               ; BUFFER FILL
                             <1> K35B:
00001240 3C53                <1>  cmp   al, 83                  ; TEST FOR DELETE KEY
00001242 742E                <1>  je    short K37C        ; HANDLE WITH OTHER EDIT KEYS
00001244 3C35                <1>  cmp   al, 53                  ; TEST FOR KEYPAD /
                             <1>  ;jne  short K32A        ; NOT THERE, NO OTHER E0 SPECIALS
00001246 0F8511FFFFFF        <1>        jne     K26
0000124C 66B800A4            <1>  mov   ax, 0A400h        ; SPECIAL CODE
```

```
00001250 E9A7010000            <1>  jmp   K57               ; BUFFER FILL
                               <1> K37:
00001255 3C3B                  <1>  cmp   al, 59                  ; TEST FOR FUNCTION KEYS (F1)
00001257 7212                  <1>        jb      short K37B           ; NO FN, HANDLE W/OTHER EXTENDED
00001259 3C44                  <1>  cmp   al, 68                  ; IN KEYPAD REGION?
                               <1>        ;ja  short K32A        ; IF SO, IGNORE
0000125B 0F87FCFEFFFF          <1>        ja      K26
00001261 80C42D                <1>  add   ah, 45                  ; CONVERT TO PSEUDO SCAN CODE
                               <1> K37A:
00001264 B000                  <1>  mov   al, 0             ; ASCII CODE OF ZERO
00001266 E991010000            <1>        jmp     K57                     ; PUT IT IN THE BUFFER
                               <1> K37B:
0000126B B0F0                  <1>  mov   al, 0F0h          ; USE SPECIAL ASCII CODE
0000126D E98A010000            <1>  jmp     K57                     ; PUT IT IN THE BUFFER
                               <1> K37C:
00001272 0450                  <1>  add   al, 80                  ; CONVERT SCAN CODE (EDIT KEYS)
00001274 88C4                  <1>  mov   ah, al                  ; (SCAN CODE NOT IN AH FOR INSERT)
00001276 EBEC                  <1>  jmp     short K37A              ; PUT IT IN THE BUFFER
                               <1>  ;
                               <1>  ;-----      NOT IN ALTERNATE SHIFT
                               <1> K38:                           ; NOT-ALT-SHIFT
                               <1>                        ; BL STILL HAS SHIFT FLAGS
00001278 F6C304                <1>  test  bl, CTL_SHIFT           ; ARE WE IN CONTROL SHIFT?
                               <1>  ;jnz  short K38A        ; YES, START PROCESSING
0000127B 0F84B0000000          <1>        jz      K44                     ; NOT-CTL-SHIFT
                               <1>  ;
                               <1>  ;-----      CONTROL SHIFT, TEST SPECIAL CHARACTERS
                               <1>  ;-----      TEST FOR BREAK
                               <1> K38A:
00001281 3C46                  <1>  cmp   al, SCROLL_KEY          ; TEST FOR BREAK
00001283 7531                  <1>  jne   short K39         ; JUMP, NO-BREAK
00001285 F6C710                <1>  test  bh, KBX                 ; IS THIS THE ENHANCED KEYBOARD?
00001288 7405                  <1>  jz    short K38B        ; NO, BREAK IS VALID
0000128A F6C702                <1>  test  bh, LC_E0        ; YES, WAS LAST CODE AN E0?
0000128D 7427                  <1>  jz    short K39         ; NO-BREAK, TEST FOR PAUSE
                               <1> K38B:
0000128F 8B1D[A06A0000]        <1>  mov   ebx, [BUFFER_HEAD]      ; RESET BUFFER TO EMPTY
00001295 891D[A46A0000]        <1>  mov   [BUFFER_TAIL], ebx
0000129B C605[926A0000]80      <1>  mov   byte [BIOS_BREAK], 80h  ; TURN ON BIOS_BREAK BIT
                               <1>  ;
                               <1>  ;-----      ENABLE KEYBOARD
000012A2 B0AE                  <1>  mov   al, ENA_KBD       ; ENABLE KEYBOARD
000012A4 E89B010000            <1>  call  SHIP_IT                 ; EXECUTE ENABLE
                               <1>  ;
                               <1>  ; CTRL+BREAK code here !!!
                               <1>  ;INT  1BH               ; BREAK INTERRUPT VECTOR
                               <1>  ; 17/10/2015
000012A9 E89E260000            <1>  call  ctrlbrk ; control+break subroutine
                               <1>  ;
000012AE 6629C0                <1>  sub   ax, ax                  ; PUT OUT DUMMY CHARACTER
000012B1 E946010000            <1>        jmp     K57                     ; BUFFER_FILL
                               <1>  ;
                               <1>  ;-----      TEST FOR PAUSE
                               <1> K39:                           ; NO_BREAK
000012B6 F6C710                <1>  test  bh, KBX                 ; IS THIS THE ENHANCED KEYBOARD?
000012B9 7537                  <1>  jnz   short K41         ; YES, THEN THIS CAN'T BE PAUSE
000012BB 3C45                  <1>  cmp   al, NUM_KEY       ; LOOK FOR PAUSE KEY
000012BD 7533                  <1>  jne   short K41         ; NO-PAUSE
                               <1> K39P:
000012BF 800D[946A0000]08      <1>  or    byte [KB_FLAG_1], HOLD_STATE ; TURN ON THE HOLD FLAG
                               <1>  ;
                               <1>  ;-----      ENABLE KEYBOARD
000012C6 B0AE                  <1>  mov   al, ENA_KBD       ; ENABLE KEYBOARD
000012C8 E877010000            <1>  call  SHIP_IT                 ; EXECUTE ENABLE
                               <1> K39A:
000012CD B020                  <1>  mov   al, EOI                 ; END OF INTERRUPT TO CONTROL PORT
000012CF E620                  <1>  out   20h, al ;out INTA00, al ; ALLOW FURTHER KEYSTROKE INTERRUPTS
                               <1>  ;
                               <1>  ;-----      DURING PAUSE INTERVAL, TURN COLOR CRT BACK ON
000012D1 803D[906A0000]07      <1>        cmp     byte [CRT_MODE], 7      ; IS THIS BLACK AND WHITE CARD
000012D8 740A                  <1>        je      short K40               ; YES, NOTHING TO DO
000012DA 66BAD803              <1>  mov   dx, 03D8h         ; PORT FOR COLOR CARD
000012DE A0[916A0000]          <1>        mov     al, [CRT_MODE_SET]   ; GET THE VALUE OF THE CURRENT MODE
000012E3 EE                    <1>  out   dx, al                  ; SET THE CRT MODE, SO THAT CRT IS ON
                               <1>  ;
                               <1> K40:                           ; PAUSE-LOOP
000012E4 F605[946A0000]08      <1>        test    byte [KB_FLAG_1], HOLD_STATE ; CHECK HOLD STATE FLAG
000012EB 75F7                  <1>  jnz   short K40         ; LOOP UNTIL FLAG TURNED OFF
                               <1>  ;
000012ED E977FEFFFF            <1>        jmp     K27                     ; INTERRUPT_RETURN_NO_EOI
                               <1>        ;
                               <1>  ;-----      TEST SPECIAL CASE KEY 55
                               <1> K41:                           ; NO-PAUSE
000012F2 3C37                  <1>  cmp   al, 55                  ; TEST FOR */PRTSC KEY
000012F4 7513                  <1>  jne   short K42         ; NOT-KEY-55
000012F6 F6C710                <1>  test  bh, KBX                 ; IS THIS THE ENHANCED KEYBOARD?
000012F9 7405                  <1>  jz    short K41A        ; NO, CTL-PRTSC IS VALID
000012FB F6C702                <1>  test  bh, LC_E0         ; YES, WAS LAST CODE AN E0?
000012FE 7421                  <1>  jz    short K42B        ; NO, TRANSLATE TO A FUNCTION
                               <1> K41A:
00001300 66B80072              <1>  mov   ax, 114*256       ; START/STOP PRINTING SWITCH
00001304 E9F3000000            <1>        jmp     K57                     ; BUFFER_FILL
```

```
                             <1>  ;
                             <1>  ;-----      SET UP TO TRANSLATE CONTROL SHIFT
                             <1> K42:                           ; NOT-KEY-55
00001309 3C0F                <1>  cmp   al, 15                  ; IS IT THE TAB KEY?
0000130B 7414                <1>  je    short K42B        ; YES, XLATE TO FUNCTION CODE
0000130D 3C35                <1>  cmp   al, 53                  ; IS IT THE / KEY?
0000130F 750E                <1>  jne   short K42A        ; NO, NO MORE SPECIAL CASES
00001311 F6C702              <1>  test  bh, LC_E0         ; YES, IS IT FROM THE KEY PAD?
00001314 7409                <1>  jz    short K42A        ; NO, JUST TRANSLATE
00001316 66B80095            <1>  mov   ax, 9500h         ; YES, SPECIAL CODE FOR THIS ONE
0000131A E9DD000000          <1>  jmp   K57               ; BUFFER FILL
                             <1> K42A:
                             <1>  ;;mov ebx, _K8          ; SET UP TO TRANSLATE CTL
0000131F 3C3B                <1>  cmp   al, 59                  ; IS IT IN CHARACTER TABLE?
                             <1>        ;jb  short K45F                ; YES, GO TRANSLATE CHAR
                             <1>  ;;jb  K56 ; 20/02/2015
                             <1>  ;;jmp K64 ; 20/02/2015
                             <1> K42B:
00001321 BB[88690000]        <1>  mov   ebx, _K8          ; SET UP TO TRANSLATE CTL
00001326 0F82AE000000        <1>  jb    K56 ;; 20/02/2015
0000132C E9B9000000          <1>  jmp   K64
                             <1>         ;
                             <1>  ;-----      NOT IN CONTROL SHIFT
                             <1> K44:                           ; NOT-CTL-SHIFT
00001331 3C37                <1>  cmp   al, 55                  ; PRINT SCREEN KEY?
00001333 7528                <1>  jne   short K45         ; NOT PRINT SCREEN
00001335 F6C710              <1>  test  bh, KBX                 ; IS THIS ENHANCED KEYBOARD?
00001338 7407                <1>  jz    short K44A        ; NO, TEST FOR SHIFT STATE
0000133A F6C702              <1>  test  bh, LC_E0         ; YES, LAST CODE A MARKER?
0000133D 7507                <1>  jnz   short K44B        ; YES, IS PRINT SCREEN
0000133F EB41                <1>  jmp   short K45C        ; NO, TRANSLATE TO '*' CHARACTER
                             <1> K44A:
00001341 F6C303              <1>  test  bl, LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
00001344 743C                <1>  jz    short K45C        ; NO, TRANSLATE TO '*' CHARACTER
                             <1>  ;
                             <1>  ;-----      ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
                             <1> K44B:
00001346 B0AE                <1>  mov   al, ENA_KBD       ; INSURE KEYBOARD IS ENABLED
00001348 E8F7000000          <1>  call  SHIP_IT                 ; EXECUTE ENABLE
0000134D B020                <1>  mov   al, EOI                 ; END OF CURRENT INTERRUPT
0000134F E620                <1>  out   20h, al ;out INTA00, al ; SO FURTHER THINGS CAN HAPPEN
                             <1>  ; Print Screen !!!            ; ISSUE PRINT SCREEN INTERRUPT (INT 05h)
                             <1>  ;PUSH       BP                ; SAVE POINTER
                             <1>  ;INT  5H                ; ISSUE PRINT SCREEN INTERRUPT
                             <1>  ;POP  BP                ; RESTORE POINTER
00001351 8025[966A0000]FC    <1>         and     byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; ZERO OUT THESE FLAGS
00001358 E90CFEFFFF          <1>         jmp     K27                      ; GO BACK WITHOUT EOI OCCURRING
                             <1>  ;
                             <1>  ;-----      HANDLE IN-CORE KEYS
                             <1> K45:                           ; NOT-PRINT-SCREEN
0000135D 3C3A                <1>  cmp   al, 58                  ; TEST FOR IN-CORE AREA
0000135F 7734                <1>  ja    short K46         ; JUMP IF NOT
00001361 3C35                <1>  cmp   al, 53                  ; IS THIS THE '/' KEY?
00001363 7505                <1>  jne   short K45A        ; NO, JUMP
00001365 F6C702              <1>  test  bh, LC_E0         ; WAS THE LAST CODE THE MARKER?
00001368 7518                <1>  jnz   short K45C        ; YES, TRANSLATE TO CHARACTER
                             <1> K45A:
0000136A B91A000000          <1>  mov   ecx, 26                 ; LENGHT OF SEARCH
0000136F BF[5E690000]        <1>  mov   edi, K30+10       ; POINT TO TABLE OF A-Z CHARS
00001374 F2AE                <1>  repne scasb             ; IS THIS A LETTER KEY?
                             <1>       ; 20/02/2015
00001376 7505                <1>  jne   short K45B               ; NO, SYMBOL KEY
                             <1>  ;
00001378 F6C340              <1>  test  bl, CAPS_STATE          ; ARE WE IN CAPS_LOCK?
0000137B 750C                <1>  jnz   short K45D        ; TEST FOR SURE
                             <1> K45B:
0000137D F6C303              <1>  test  bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
00001380 750C                <1>  jnz   short K45E        ; YES, UPPERCASE
                             <1>                          ; NO, LOWERCASE
                             <1> K45C:
00001382 BB[E0690000]        <1>  mov   ebx, K10          ; TRANSLATE TO LOWERCASE LETTERS
00001387 EB51                <1>  jmp   short K56
                             <1> K45D:                          ; ALMOST-CAPS-STATE
00001389 F6C303              <1>  test  bl, LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
0000138C 75F4                <1>  jnz   short K45C        ; SHIFTED TEMP OUT OF CAPS STATE
                             <1> K45E:
0000138E BB[386A0000]        <1>  mov   ebx, K11          ; TRANSLATE TO UPPER CASE LETTERS
00001393 EB45                <1> K45F:  jmp   short K56
                             <1>  ;
                             <1>  ;-----      TEST FOR KEYS F1 - F10
                             <1> K46:                           ; NOT IN-CORE AREA
00001395 3C44                <1>  cmp   al, 68                  ; TEST FOR F1 - F10
                             <1>  ;ja   short K47         ; JUMP IF NOT
                             <1>  ;jmp short K53          ; YES, GO DO FN KEY PROCESS
00001397 7635                <1>  jna   short K53
                             <1>  ;
                             <1>  ;-----      HANDLE THE NUMERIC PAD KEYS
                             <1> K47:                           ; NOT F1 - F10
00001399 3C53                <1>  cmp   al, 83                  ; TEST NUMPAD KEYS
0000139B 772D                <1>  ja    short K52         ; JUMP IF NOT
                             <1>  ;
                             <1>  ;-----      KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
                             <1> K48:
0000139D 3C4A                <1>  cmp   al , 74                 ; SPECIAL CASE FOR MINUS
0000139F 74ED                <1>  je    short K45E        ; GO TRANSLATE
000013A1 3C4E                <1>  cmp   al , 78                 ; SPECIAL CASE FOR PLUS
000013A3 74E9                <1>  je    short K45E        ; GO TRANSLATE
000013A5 F6C702              <1>  test  bh, LC_E0         ; IS THIS ONE OFTHE NEW KEYS?
000013A8 750A                <1>  jnz   short K49         ; YES, TRANSLATE TO BASE STATE
                             <1>  ;
000013AA F6C320              <1>  test  bl, NUM_STATE           ; ARE WE IN NUM LOCK
000013AD 7514                <1>  jnz   short K50         ; TEST FOR SURE
000013AF F6C303              <1>  test  bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
                             <1>  ;jnz  short K51         ; IF SHIFTED, REALLY NUM STATE
000013B2 75DA                <1>  jnz   short K45E
                             <1>  ;
                             <1>  ;-----      BASE CASE FOR KEYPAD
                             <1> K49:
000013B4 3C4C                <1>  cmp   al, 76                  ; SPECIAL CASE FOR BASE STATE 5
000013B6 7504                <1>  jne   short K49A        ; CONTINUE IF NOT KEYPAD 5
000013B8 B0F0                <1>  mov   al, 0F0h          ; SPECIAL ASCII CODE
000013BA EB40                <1>  jmp   short K57         ; BUFFER FILL
                             <1> K49A:
000013BC BB[E0690000]        <1>  mov   ebx, K10          ; BASE CASE TABLE
000013C1 EB27                <1>  jmp   short K64         ; CONVERT TO PSEUDO SCAN
                             <1>  ;
                             <1>  ;-----      MIGHT BE NUM LOCK, TEST SHIFT STATUS
                             <1> K50:                           ; ALMOST-NUM-STATE
000013C3 F6C303              <1>         test    bl, LEFT_SHIFT+RIGHT_SHIFT
000013C6 75EC                <1>  jnz   short K49         ; SHIFTED TEMP OUT OF NUM STATE
000013C8 EBC4                <1> K51:    jmp   short K45E        ; REALLY NUM STATE
                             <1>  ;
                             <1>  ;-----      TEST FOR THE NEW KEYS ON WT KEYBOARDS
                             <1> K52:                           ; NOT A NUMPAD KEY
000013CA 3C56                <1>  cmp   al, 86                  ; IS IT THE NEW WT KEY?
                             <1>  ;jne  short K53         ; JUMP IF NOT
                             <1>  ;jmp  short K45B        ; HANDLE WITH REST OF LETTER KEYS
000013CC 74AF                <1>  je    short K45B
                             <1>  ;
                             <1>  ;-----      MUST BE F11 OR F12
                             <1> K53:                           ; F1 - F10 COME HERE, TOO
000013CE F6C303              <1>  test  bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST SHIFT STATE
000013D1 74E1                <1>  jz    short K49         ; JUMP, LOWER CASE PSEUDO SC'S
                             <1>        ; 20/02/2015
000013D3 BB[386A0000]        <1>  mov   ebx, K11          ; UPPER CASE PSEUDO SCAN CODES
000013D8 EB10                <1>  jmp   short K64         ; TRANSLATE SCAN
                             <1>  ;
                             <1>  ;-----      TRANSLATE THE CHARACTER
                             <1> K56:                           ; TRANSLATE-CHAR
000013DA FEC8                <1>  dec   al                ; CONVERT ORIGIN
000013DC D7                  <1>  xlat                    ; CONVERT THE SCAN CODE TO ASCII
000013DD F605[966A0000]02    <1>  test  byte [KB_FLAG_3], LC_E0 ; IS THIS A NEW KEY?
000013E4 7416                <1>  jz    short K57         ; NO, GO FILL BUFFER
000013E6 B4E0                <1>  mov   ah, MC_E0         ; YES, PUT SPECIAL MARKER IN AH
000013E8 EB12                <1>  jmp   short K57         ; PUT IT INTO THE BUFFER
                             <1>  ;
                             <1>  ;-----      TRANSLATE SCAN FOR PSEUDO SCAN CODES
                             <1> K64:                           ; TRANSLATE-SCAN-ORGD
000013EA FEC8                <1>  dec   al                ; CONVERT ORIGIN
000013EC D7                  <1>        xlat                           ; CTL TABLE SCAN
000013ED 88C4                <1>  mov   ah, al                  ; PUT VALUE INTO AH
000013EF B000                <1>  mov   al, 0             ; ZERO ASCII CODE
000013F1 F605[966A0000]02    <1>  test  byte [KB_FLAG_3], LC_E0 ; IS THIS A NEW KEY?
000013F8 7402                <1>  jz    short K57         ; NO, GO FILL BUFFER
000013FA B0E0                <1>  mov   al, MC_E0         ; YES, PUT SPECIAL MARKER IN AL
                             <1>  ;
                             <1>  ;-----      PUT CHARACTER INTO BUFFER
                             <1> K57:                           ; BUFFER_FILL
000013FC 3CFF                <1>  cmp   al, -1                  ; IS THIS AN IGNORE CHAR
                             <1>        ;je  short K59          ; YES, DO NOTHING WITH IT
000013FE 0F8459FDFFFF        <1>  je      K26             ; YES, DO NOTHING WITH IT
00001404 80FCFF              <1>  cmp   ah, -1                  ; LOOK FOR -1 PSEUDO SCAN
                             <1>        ;jne short K61          ; NEAR_INTERRUPT_RETURN
00001407 0F8450FDFFFF        <1>  je      K26             ; INTERRUPT_RETURN
                             <1> ;K59:                          ; NEAR_INTERRUPT_RETURN
                             <1> ;jmp   K26               ; INTERRUPT_RETURN
                             <1> K61:                           ; NOT-CAPS-STATE
0000140D 8B1D[A46A0000]      <1>  mov   ebx, [BUFFER_TAIL]      ; GET THE END POINTER TO THE BUFFER
00001413 89DE                <1>  mov   esi, ebx          ; SAVE THE VALUE
00001415 E87BFAFFFF          <1>  call  _K4               ; ADVANCE THE TAIL
0000141A 3B1D[A06A0000]      <1>  cmp   ebx, [BUFFER_HEAD]      ; HAS THE BUFFER WRAPPED AROUND
00001420 740E                <1>  je    short K62         ; BUFFER_FULL_BEEP
00001422 668906              <1>  mov   [esi], ax         ; STORE THE VALUE
00001425 891D[A46A0000]      <1>  mov   [BUFFER_TAIL], ebx      ; MOVE THE POINTER UP
0000142B E92DFDFFFF          <1>  jmp   K26
                             <1>  ;;cli                   ; TURN OFF INTERRUPTS
                             <1>  ;;mov al, EOI                 ; END OF INTERRUPT COMMAND
                             <1>  ;;out INTA00, al        ; SEND COMMAND TO INTERRUPT CONTROL PORT
                             <1>  ;MOV  AL, ENA_KBD       ; INSURE KEYBOARD IS ENABLED
                             <1>  ;CALL SHIP_IT                 ; EXECUTE ENABLE
                             <1>  ;MOV  AX, 9102H         ; MOVE IN POST CODE & TYPE
                             <1>  ;INT  15H               ; PERFORM OTHER FUNCTION
                             <1>  ;;and byte [KB_FLAG_3],~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
                             <1>  ;JMP  K27A              ; INTERRUPT_RETURN
                             <1>  ;;jmp   K27
                             <1>  ;
                             <1>  ;-----      BUFFER IS FULL SOUND THE BEEPER
                             <1> K62:
00001430 B020                <1>  mov   al, EOI                 ; ENABLE INTERRUPT CONTROLLER CHIP
00001432 E620                <1>  out   INTA00, al
00001434 66B9A602            <1>  mov   cx, 678                 ; DIVISOR FOR 1760 HZ
00001438 B304                <1>  mov   bl, 4             ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
0000143A E8A1010000          <1>  call  beep              ; GO TO COMMON BEEP HANDLER
0000143F E925FDFFFF          <1>  jmp     K27             ; EXIT
                             <1>
                             <1> SHIP_IT:
                             <1>  ;--------------------------------------------------------------------------------
-
                             <1>  ; SHIP_IT
                             <1>  ;     THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
                             <1>  ;     TO THE KEYBOARD CONTROLLER.
                             <1>  ;--------------------------------------------------------------------------------
-
                             <1>  ;
00001444 6650                <1>  push  ax                ; SAVE DATA TO SEND
                             <1>
                             <1>  ;-----      WAIT FOR COMMAND TO ACCEPTED
00001446 FA                  <1>  cli                     ; DISABLE INTERRUPTS TILL DATA SENT
                             <1>  ; xor ecx, ecx          ; CLEAR TIMEOUT COUNTER
00001447 B900000100          <1>  mov   ecx, 10000h
                             <1> S10:
0000144C E464                <1>  in    al, STATUS_PORT         ; READ KEYBOARD CONTROLLER STATUS
0000144E A802                <1>  test  al, INPT_BUF_FULL ; CHECK FOR ITS INPUT BUFFER BUSY
00001450 E0FA                <1>  loopnz      S10               ; WAIT FOR COMMAND TO BE ACCEPTED
                             <1>
00001452 6658                <1>  pop   ax                ; GET DATA TO SEND
00001454 E664                <1>  out   STATUS_PORT, al         ; SEND TO KEYBOARD CONTROLLER
00001456 FB                  <1>  sti                     ; ENABLE INTERRUPTS AGAIN
00001457 C3                  <1>  retn                    ; RETURN TO CALLER
                             <1>
                             <1> SND_DATA:
                             <1>  ; ------------------------------------------------------------------------------
--
                             <1>  ; SND_DATA
                             <1>  ;     THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
                             <1>  ;     TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS. IT ALSO
                             <1>  ;     HANDLES ANY RETRIES IF REQUIRED
                             <1>  ; ------------------------------------------------------------------------------
--
                             <1>  ;
00001458 6650                <1>  push  ax                ; SAVE REGISTERS
0000145A 6653                <1>  push  bx
0000145C 51                  <1>  push  ecx
0000145D 88C7                <1>  mov   bh, al                  ; SAVE TRANSMITTED BYTE FOR RETRIES
0000145F B303                <1>  mov   bl, 3             ; LOAD RETRY COUNT
                             <1> SD0:
00001461 FA                  <1>  cli                     ; DISABLE INTERRUPTS
00001462 8025[956A0000]CF    <1>  and   byte [KB_FLAG_2], ~(KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
                             <1>  ;
                             <1>  ;-----      WAIT FOR COMMAND TO BE ACCEPTED
00001469 B900000100          <1>  mov   ecx, 10000h       ; MAXIMUM WAIT COUNT
                             <1> SD5:
0000146E E464                <1>  in    al, STATUS_PORT         ; READ KEYBOARD PROCESSOR STATUS PORT
00001470 A802                <1>  test  al, INPT_BUF_FULL ; CHECK FOR ANY PENDING COMMAND
00001472 E0FA                <1>  loopnz      SD5               ; WAIT FOR COMMAND TO BE ACCEPTED
                             <1>  ;
00001474 88F8                <1>  mov   al, bh                  ; REESTABLISH BYTE TO TRANSMIT
00001476 E660                <1>  out   PORT_A, al        ; SEND BYTE
00001478 FB                  <1>  sti                     ; ENABLE INTERRUPTS
                             <1>  ;mov  cx, 01A00h        ; LOAD COUNT FOR 10 ms+
00001479 B9FFFF0000          <1>  mov   ecx, 0FFFFh
                             <1> SD1:
0000147E F605[956A0000]30    <1>  test  byte [KB_FLAG_2], KB_FE+KB_FA ; SEE IF EITHER BIT SET
00001485 750F                <1>  jnz   short SD3         ; IF SET, SOMETHING RECEIVED GO PROCESS
00001487 E2F5                <1>  loop  SD1               ; OTHERWISE WAIT
                             <1> SD2:
00001489 FECB                <1>  dec   bl                ; DECREMENT RETRY COUNT
0000148B 75D4                <1>  jnz   short SD0         ; RETRY TRANSMISSION
0000148D 800D[956A0000]80    <1>  or    byte [KB_FLAG_2], KB_ERR ; TURN ON TRANSMIT ERROR FLAG
00001494 EB09                <1>  jmp   short SD4         ; RETRIES EXHAUSTED FORGET TRANSMISSION
                             <1> SD3:
00001496 F605[956A0000]10    <1>  test  byte [KB_FLAG_2], KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
0000149D 74EA                <1>  jz    short SD2         ; IF NOT, GO RESEND
                             <1> SD4:
0000149F 59                  <1>  pop   ecx               ; RESTORE REGISTERS
000014A0 665B                <1>  pop   bx
000014A2 6658                <1>  pop   ax
000014A4 C3                  <1>  retn                    ; RETURN, GOOD TRANSMISSION
                             <1>
                             <1> SND_LED:
                             <1>  ; ------------------------------------------------------------------------------
--
                             <1>  ; SND_LED
                             <1>  ;     THIS ROUTINES TURNS ON THE MODE INDICATORS.
                             <1>  ;
                             <1>  ;---------------------------------------------------------------------------------
--
                             <1>  ;
000014A5 FA                  <1>  cli                      ; TURN OFF INTERRUPTS
000014A6 F605[956A0000]40    <1>  test  byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
000014AD 755F                <1>  jnz   short SL1         ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
                             <1>  ;
000014AF 800D[956A0000]40    <1>  or    byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
000014B6 B020                <1>  mov   al, EOI                 ; END OF INTERRUPT COMMAND
000014B8 E620                <1>  out   20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
000014BA EB11                <1>  jmp   short SL0         ; GO SEND MODE INDICATOR COMMAND
                             <1> SND_LED1:
000014BC FA                  <1>  cli                      ; TURN OFF INTERRUPTS
000014BD F605[956A0000]40    <1>  test  byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
000014C4 7548                <1>  jnz   short SL1         ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
                             <1>  ;
000014C6 800D[956A0000]40    <1>  or    byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
                             <1> SL0:
000014CD B0ED                <1>  mov   al, LED_CMD       ; LED CMD BYTE
000014CF E884FFFFFF          <1>  call  SND_DATA          ; SEND DATA TO KEYBOARD
000014D4 FA                  <1>  cli
000014D5 E836000000          <1>  call  MAKE_LED          ; GO FORM INDICATOR DATA BYTE
000014DA 8025[956A0000]F8    <1>  and   byte [KB_FLAG_2], 0F8h  ; ~KB_LEDS ; CLEAR MODE INDICATOR BITS
000014E1 0805[956A0000]      <1>  or    [KB_FLAG_2], al   ; SAVE PRESENT INDICATORS FOR NEXT TIME
000014E7 F605[956A0000]80    <1>  test  byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
000014EE 750F                <1>  jnz   short SL2         ; IF SO, BYPASS SECOND BYTE TRANSMISSION
                             <1>  ;
000014F0 E863FFFFFF          <1>  call  SND_DATA          ; SEND DATA TO KEYBOARD
000014F5 FA                  <1>  cli                      ; TURN OFF INTERRUPTS
000014F6 F605[956A0000]80    <1>  test  byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
000014FD 7408                <1>  jz    short SL3         ; IF NOT, DON'T SEND AN ENABLE COMMAND
                             <1> SL2:
000014FF B0F4                <1>  mov   al, KB_ENABLE           ; GET KEYBOARD CSA ENABLE COMMAND
00001501 E852FFFFFF          <1>  call  SND_DATA          ; SEND DATA TO KEYBOARD
00001506 FA                  <1>  cli                      ; TURN OFF INTERRUPTS
                             <1> SL3:
00001507 8025[956A0000]3F    <1>  and   byte [KB_FLAG_2], ~(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
                             <1> SL1:                           ; UPDATE AND TRANSMIT ERROR FLAG
0000150E FB                  <1>  sti                      ; ENABLE INTERRUPTS
0000150F C3                  <1>  retn                     ; RETURN TO CALLER
                             <1>
                             <1> MAKE_LED:
                             <1>  ;---------------------------------------------------------------------------------
-
                             <1>  ; MAKE_LED
                             <1>  ;     THIS ROUTINES FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF
                             <1>  ;     THE MODE INDICATORS.
                             <1>  ;---------------------------------------------------------------------------------
-
                             <1>  ;
                             <1>  ;push       cx                ; SAVE CX
00001510 A0[936A0000]        <1>  mov   al, [KB_FLAG]           ; GET CAPS & NUM LOCK INDICATORS
00001515 2470                <1>  and   al, CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
                             <1>  ;mov  cl, 4             ; SHIFT COUNT
                             <1>  ;rol  al, cl                  ; SHIFT BITS OVER TO TURN ON INDICATORS
00001517 C0C004              <1>  rol   al, 4 ; 20/02/2015
0000151A 2407                <1>  and   al, 07h                 ; MAKE SURE ONLY MODE BITS ON
                             <1>  ;pop  cx
0000151C C3                  <1>  retn                     ; RETURN TO CALLER
                             <1>
                             <1> ; % include 'kybdata.inc'   ; KEYBOARD DATA ; 11/03/2015
                             <1>
                             <1>
                             <1> ; /// End Of KEYBOARD FUNCTIONS ///

                                 %include 'video.inc' ; 07/03/2015
                             <1> ; Retro UNIX 386 v1 Kernel - VIDEO.INC
                             <1> ; Last Modification: 13/08/2015
                             <1> ;        (Video Data is in 'VIDATA.INC')
                             <1> ;
                             <1> ; ///////// VIDEO (CGA) FUNCTIONS ///////////////
                             <1>
                             <1> ; 30/06/2015
                             <1> ; 27/06/2015
                             <1> ; 11/03/2015
                             <1> ; 02/09/2014
                             <1> ; 30/08/2014
                             <1> ; VIDEO FUNCTIONS
                             <1> ; (write_tty - Retro UNIX 8086 v1 - U9.ASM, 01/02/2014)
                             <1>
                             <1> write_tty:
                             <1>  ; 13/08/2015
                             <1>  ; 02/09/2014
                             <1>  ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
                             <1>  ; 01/02/2014 (Retro UNIX 8086 v1 - last update)
                             <1>  ; 03/12/2013 (Retro UNIX 8086 v1 - beginning)
                             <1>  ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
                             <1>  ;
                             <1>  ; INPUT -> AH = Color (Forecolor, Backcolor)
                             <1>  ;        AL = Character to be written
                             <1>  ;        EBX = Video Page (0 to 7)
                             <1>  ;        (BH = 0 --> Video Mode 3)
                             <1>
                             <1> RVRT   equ   00001000b   ; VIDEO VERTICAL RETRACE BIT
                             <1> RHRZ   equ   00000001b   ; VIDEO HORIZONTAL RETRACE BIT
                             <1>
                             <1> ; Derived from "WRITE_TTY" procedure of IBM "pc-at" rombios source code
                             <1> ; (06/10/1985), 'video.asm', INT 10H, VIDEO_IO
                             <1> ;
                             <1> ; 06/10/85  VIDEO DISPLAY BIOS
                             <1> ;
                             <1> ;--- WRITE_TTY ----------------------------------------------------------------------
                             <1> ;                                                      :
                             <1> ;   THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE             :
                             <1> ;   VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT                :
                             <1> ;   CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.       :
                             <1> ;   IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN        :
                             <1> ;   IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW         :
                             <1> ;   ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,         :
                             <1> ;   FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.         :
                             <1> ;   WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE        :
                             <1> ;   NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS       :
                             <1> ;   LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,         :
                             <1> ;   THE 0 COLOR IS USED.                                       :
                             <1> ;   ENTRY --                                                   :
                             <1> ;     (AH) = CURRENT CRT MODE                                       :
                             <1> ;     (AL) = CHARACTER TO BE WRITTEN                                :
                             <1> ;    NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE      :
                             <1> ;    HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS     :
                             <1> ;     (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE  :
                             <1> ;   EXIT --                                                    :
                             <1> ;     ALL REGISTERS SAVED                                           :
                             <1> ;--------------------------------------------------------------------------------
                             <1>
0000151D FA                  <1>  cli
                             <1>  ;
                             <1>  ; READ CURSOR (04/12/2013)
                             <1>  ; Retro UNIX 386 v1 Modifications: 30/08/2014
0000151E 08FF                <1>  or    bh, bh
00001520 0F85AB000000        <1>  jnz   beeper
                             <1>  ; 01/09/2014
00001526 803D[906A0000]03    <1>  cmp   byte [CRT_MODE], 3
0000152D 7405                <1>  je    short m3
                             <1>  ;
0000152F E889020000          <1>  call  set_mode
                             <1> m3:
00001534 89DE                <1>  mov   esi, ebx ; 13/08/2015 (0 to 7)
00001536 66D1E6              <1>  shl   si, 1
00001539 81C6[C6700000]      <1>  add   esi, cursor_posn
0000153F 668B16              <1>  mov   dx, [esi]
                             <1>  ;
                             <1>  ; dx now has the current cursor position
                             <1>  ;
00001542 3C0D                <1>  cmp   al, 0Dh           ; is it carriage return or control character
00001544 764D                <1>  jbe   short u8
                             <1>  ;
                             <1>  ; write the char to the screen
                             <1> u0:
                             <1>  ; ah = attribute/color
                             <1>  ; al = character
                             <1>  ; bl = video page number (0 to 7)
                             <1>  ; bh = 0
                             <1>  ;
00001546 E841020000          <1>  call  write_c_current
                             <1>  ;
                             <1>  ; position the cursor for next char
0000154B FEC2                <1>  inc   dl          ; next column
                             <1>  ;cmp  dl, [CRT_COLS]
0000154D 80FA50              <1>  cmp   dl, 80            ; test for column overflow
00001550 0F85EB000000        <1>         jne     set_cpos
00001556 B200                <1>  mov   dl, 0       ; column = 0
                             <1> u10:                     ; (line feed found)
00001558 80FE18              <1>  cmp   dh, 25-1    ; check for last row
0000155B 722F                <1>  jb    short u6
                             <1>  ;
                             <1>  ; scroll required
                             <1> u1:
                             <1>  ; SET CURSOR POSITION (04/12/2013)
0000155D E8DF000000          <1>  call  set_cpos
                             <1>  ;
                             <1>  ; determine value to fill with during scroll
                             <1> u2:
                             <1>  ; READ_AC_CURRENT       :
                             <1>  ;   THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER
                             <1>  ;    AT THE CURRENT CURSOR POSITION
                             <1>  ;
                             <1>  ; INPUT
                             <1>  ;     (AH) = CURRENT CRT MODE
                             <1>  ;     (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
                             <1>  ;     (DS) = DATA SEGMENT
                             <1>  ;     (ES) = REGEN SEGMENT
                             <1>  ; OUTPUT
                             <1>  ;     (AL) = CHARACTER READ
                             <1>  ;     (AH) = ATTRIBUTE READ
                             <1>  ;
                             <1>  ; mov ah, [CRT_MODE] ; move current mode into ah
                             <1>  ;
                             <1>  ; bl = video page number
                             <1>  ;
00001562 E837010000          <1>  call  find_position     ; get regen location and port address
                             <1>  ; dx = status port
                             <1>  ; esi = cursor location/address
                             <1> p11:
00001567 FB                  <1>  sti               ; enable interrupts
00001568 90                  <1>  nop               ; allow for small interupts window
00001569 FA                  <1>  cli               ; blocks interrupts for single loop
0000156A EC                  <1>  in    al, dx            ; get status from adapter
0000156B A801                <1>  test  al, RHRZ    ; is horizontal retrace low
0000156D 75F8                <1>  jnz   short p11   ; wait until it is
                             <1> p12:                     ; now wait for either retrace high
0000156F EC                  <1>  in    al, dx            ; get status
00001570 A809                <1>  test  al, RVRT+RHRZ     ; is horizontal or vertical retrace high
00001572 74FB                <1>  jz    short p12   ; wait until either is active
                             <1> p13:
00001574 81C600800B00        <1>  add   esi, 0B8000h      ; 30/08/2014 (Retro UNIX 386 v1)
0000157A 668B06              <1>  mov   ax, [esi]   ; get the character and attribute
                             <1>  ;
                             <1>  ; al = character, ah = attribute
                             <1>  ;
0000157D FB                  <1>  sti
                             <1>  ; bl = video page number
                             <1> u3:
                             <1>  ;;mov ax, 0601h  ; scroll one line
                             <1>  ;;sub cx, cx            ; upper left corner
                             <1>  ;;mov dh, 25-1    ; lower right row
                             <1>  ;;;mov      dl, [CRT_COLS]
                             <1>  ;mov  dl, 80            ; lower right column
                             <1>  ;;dec dl
                             <1>  ;;mov dl, 79
                             <1>
                             <1>  ;;call      scroll_up   ; 04/12/2013
                             <1>  ;;; 11/03/2015
                             <1>  ; 02/09/2014
                             <1>  ;;;mov      cx, [crt_ulc] ; Upper left corner  (0000h)
                             <1>  ;;;mov      dx, [crt_lrc] ; Lower right corner (184Fh)
                             <1>  ; 11/03/2015
0000157E 6629C9              <1>  sub   cx, cx
00001581 66BA4F18            <1>  mov   dx, 184Fh ; dl= 79 (column), dh = 24 (row)
                             <1>  ;
00001585 B001                <1>  mov   al, 1       ; scroll 1 line up
                             <1>        ; ah = attribute
00001587 E93E010000          <1>  jmp   scroll_up
                             <1> ;u4:
                             <1>  ;;int 10h         ; video-call return
                             <1>                    ; scroll up the screen
                             <1>                    ; tty return
                             <1> ;u5:
                             <1>  ;retn             ; return to the caller
                             <1>
                             <1> u6:                      ; set-cursor-inc
0000158C FEC6                <1>  inc   dh          ; next row
                             <1>                    ; set cursor
                             <1> ;u7:
                             <1>  ;;mov ah, 02h
                             <1>  ;;jmp short u4    ; establish the new cursor
                             <1>  ;call set_cpos
                             <1>  ;jmp  short u5
0000158E E9AE000000          <1>  jmp     set_cpos
                             <1>
                             <1>  ; check for control characters
                             <1> u8:
00001593 7438                <1>  je    short u9
00001595 3C0A                <1>  cmp   al, 0Ah           ; is it a line feed (0Ah)
00001597 74BF                <1>  je    short u10
00001599 3C07                <1>  cmp   al, 07h     ; is it a bell
0000159B 7434                <1>  je    short u11
0000159D 3C08                <1>  cmp   al, 08h           ; is it a backspace
                             <1>  ;jne  short u0
0000159F 7424                <1>  je    short bs    ; 12/12/2013
                             <1>  ; 12/12/2013 (tab stop)
000015A1 3C09                <1>  cmp   al, 09h           ; is it a tab stop
000015A3 75A1                <1>  jne   short u0
000015A5 88D0                <1>  mov   al, dl
000015A7 6698                <1>  cbw
000015A9 B108                <1>  mov   cl, 8
000015AB F6F1                <1>  div   cl
000015AD 28E1                <1>  sub   cl, ah
                             <1> ts:
                             <1>  ; 02/09/2014
                             <1>  ; 01/09/2014
000015AF B020                <1>  mov   al, 20h
                             <1> tsloop:
000015B1 6651                <1>  push  cx
000015B3 6650                <1>  push  ax
000015B5 30FF                <1>  xor   bh, bh
                             <1>  ;mov  bl, [active_page]
```

```
000015B7 E878FFFFFF         <1>  call  m3
000015BC 6658               <1>  pop   ax  ; ah = attribute/color
000015BE 6659               <1>  pop   cx
000015C0 FEC9               <1>  dec   cl
000015C2 75ED               <1>  jnz   short tsloop
000015C4 C3                 <1>  retn
                            <1> bs:
                            <1>  ; back space found
                            <1>
000015C5 08D2               <1>  or    dl, dl            ; is it already at start of line
                            <1>  ;je   short u7    ; set_cursor
000015C7 7478               <1>  jz    short set_cpos
000015C9 664A               <1>  dec   dx                ; no -- just move it back
                            <1>  ;jmp  short u7
000015CB EB74               <1>  jmp   short set_cpos
                            <1>
                            <1>  ; carriage return found
                            <1> u9:
000015CD B200               <1>  mov   dl, 0             ; move to first column
                            <1>  ;jmp  short u7
000015CF EB70               <1>  jmp   short set_cpos
                            <1>
                            <1>  ; line feed found
                            <1> ;u10:
                            <1> ;cmp   dh, 25-1    ; bottom of screen
                            <1> ;jne   short u6    ; no, just set the cursor
                            <1> ;       jmp     u1              ; yes, scroll the screen
                            <1>
                            <1> beeper:
                            <1>  ; 30/08/2014 (Retro UNIX 386 v1)
                            <1>  ; 18/01/2014
                            <1>  ; 03/12/2013
                            <1>  ; bell found
                            <1> u11:
000015D1 FB                 <1>  sti
000015D2 3A1D[D6700000]     <1>  cmp   bl, [active_page]
000015D8 7551               <1>  jne   short u12   ; Do not sound the beep
                            <1>                    ; if it is not written on the active page
000015DA 66B93305           <1>  mov   cx, 1331    ; divisor for 896 hz tone
000015DE B31F               <1>  mov   bl, 31            ; set count for 31/64 second for beep
                            <1>  ;call beep        ; sound the pod bell
                            <1>  ;jmp  short u5    ; tty_return
                            <1>  ;retn
                            <1>
                            <1> TIMER  equ   040h              ; 8254 TIMER - BASE ADDRESS
                            <1> PORT_B equ   061h        ; PORT B READ/WRITE DIAGNOSTIC REGISTER
                            <1> GATE2  equ   00000001b   ; TIMER 2 INPUT CATE CLOCK BIT
                            <1> SPK2   equ   00000010b   ; SPEAKER OUTPUT DATA ENABLE BIT
                            <1>
                            <1> beep:
                            <1>  ; 07/02/2015
                            <1>  ; 30/08/2014 (Retro UNIX 386 v1)
                            <1>  ; 18/01/2014
                            <1>  ; 03/12/2013
                            <1>  ;
                            <1>  ; TEST4.ASM - 06/10/85  POST AND BIOS UTILITY ROUTINES
                            <1>  ;
                            <1>  ; ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE
                            <1>  ;
                            <1>  ; ENTRY:
                            <1>  ;    (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )
                            <1>  ;    (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)
                            <1>  ; EXIT:                         :
                            <1>  ;    (AX),(BL),(CX) MODIFIED.
                            <1>
000015E0 9C                 <1>  pushf  ; 18/01/2014     ; save interrupt status
000015E1 FA                 <1>  cli               ; block interrupts during update
000015E2 B0B6               <1>  mov   al, 10110110b     ; select timer 2, lsb, msb binary
000015E4 E643               <1>  out   TIMER+3, al       ; write timer mode register
000015E6 EB00               <1>  jmp   $+2         ; I/O delay
000015E8 88C8               <1>  mov   al, cl            ; divisor for hz (low)
000015EA E642               <1>  out   TIMER+2,AL  ; write timer 2 count - lsb
000015EC EB00               <1>  jmp   $+2         ; I/O delay
000015EE 88E8               <1>  mov   al, ch            ; divisor for hz (high)
000015F0 E642               <1>  out   TIMER+2, al ; write timer 2 count - msb
000015F2 E461               <1>  in    al, PORT_B  ; get current setting of port
000015F4 88C4               <1>  mov   ah, al            ; save that setting
000015F6 0C03               <1>  or    al, GATE2+SPK2    ; gate timer 2 and turn speaker on
000015F8 E661               <1>  out   PORT_B, al  ; and restore interrupt status
                            <1>  ;popf ; 18/01/2014
000015FA FB                 <1>  sti
                            <1> g7:                      ; 1/64 second per count (bl)
000015FB B90B040000         <1>  mov   ecx, 1035   ; delay count for 1/64 of a second
00001600 E827000000         <1>  call  waitf       ; go to beep delay 1/64 count
00001605 FECB               <1>  dec   bl          ; (bl) length count expired?
00001607 75F2               <1>  jnz   short g7    ; no - continue beeping speaker
                            <1>  ;
                            <1>  ;pushf                  ; save interrupt status
00001609 FA                 <1>  cli   ; 18/01/2014      ; block interrupts during update
0000160A E461               <1>  in    al, PORT_B  ; get current port value
                            <1>         ;or      al, not (GATE2+SPK2) ; isolate current speaker bits in case
0000160C 0CFC               <1>         or      al, ~(GATE2+SPK2)
```

```
4034 0000160E 20C4                <1>        and  ah, al           ; someone turned them off during beep
4035 00001610 88E0                <1>  mov   al, ah           ; recover value of port
4036                              <1>        ;or      al, not (GATE2+SPK2) ; force speaker data off
4037 00001612 0CFC                <1>  or    al, ~(GATE2+SPK2) ; isolate current speaker bits in case
4038 00001614 E661                <1>  out   PORT_B, al  ; and stop speaker timer
4039                              <1>  ;popf             ; restore interrupt flag state
4040 00001616 FB                  <1>  sti
4041 00001617 B90B040000          <1>  mov   ecx, 1035   ; force 1/64 second delay (short)
4042 0000161C E80B000000          <1>  call  waitf       ; minimum delay between all beeps
4043                              <1>  ;pushf                  ; save interrupt status
4044 00001621 FA                  <1>  cli               ; block interrupts during update
4045 00001622 E461                <1>  in    al, PORT_B  ; get current port value in case
4046 00001624 2403                <1>  and   al, GATE2+SPK2    ; someone turned them on
4047 00001626 08E0                <1>  or    al, ah           ; recover value of port_b
4048 00001628 E661                <1>  out   PORT_B, al  ; restore speaker status
4049 0000162A 9D                  <1>  popf              ; restore interrupt flag state
4050                              <1> u12:
4051 0000162B C3                  <1>  retn
4052                              <1>
4053                              <1> REFRESH_BIT equ    00010000b   ; REFRESH TEST BIT
4054                              <1>
4055                              <1> WAITF:
4056                              <1> waitf:
4057                              <1>  ; 30/08/2014 (Retro UNIX 386 v1)
4058                              <1>  ; 03/12/2013
4059                              <1>  ;
4060                              <1> ;push ax                ; save work register (ah)
4061                              <1> ;waitf1:
4062                              <1>                    ; use timer 1 output bits
4063                              <1> ;in    al, PORT_B  ; read current counter output status
4064                              <1> ;and   al, REFRESH_BIT   ; mask for refresh determine bit
4065                              <1> ;cmp   al, ah           ; did it just change
4066                              <1> ;je    short waitf1      ; wait for a change in output line
4067                              <1> ;;
4068                              <1> ;mov   ah, al           ; save new lflag state
4069                              <1> ;loop  waitf1           ; decrement half cycles till count end
4070                              <1> ;;
4071                              <1> ;pop   ax          ; restore (ah)
4072                              <1> ;retn              ; return (cx)=0
4073                              <1>
4074                              <1> ; 06/02/2015 (unix386.s <-- dsectrm2.s)
4075                              <1> ; 17/12/2014 (dsectrm2.s)
4076                              <1> ; WAITF
4077                              <1> ; /// IBM PC-XT Model 286 System BIOS Source Code - Test 4 - 06/10/85 ///
4078                              <1> ;
4079                              <1> ;---WAITF-----------------------------------------------------------------------
4080                              <1> ;FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)
4081                              <1> ; ENTRY:
4082                              <1> ;(CX) =      COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT
4083                              <1> ;            MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE
4084                              <1> ; EXIT:
4085                              <1> ;            AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS)
4086                              <1> ;(CX) = 0
4087                              <1> ;-------------------------------------------------------------------------------
4088                              <1>
4089                              <1> ; Refresh period: 30 micro seconds (15-80 us)
4090                              <1> ; (16/12/2014 - AWARDBIOS 1999 - ATORGS.ASM, WAIT_REFRESH)
4091                              <1>
4092                              <1> ;WAITF:                          ; DELAY FOR (CX)*15.085737 US
4093 0000162C 6650                <1>  PUSH  AX                ; SAVE WORK REGISTER (AH)
4094                              <1>  ; 16/12/2014
4095                              <1>  ;shr  cx, 1             ; convert to count of 30 micro seconds
4096 0000162E D1E9                <1>  shr   ecx, 1      ; 21/02/2015
4097                              <1> ;17/12/2014
4098                              <1> ;WAITF1:
4099                              <1> ;IN    AL, PORT_B   ;061h      ; READ CURRENT COUNTER OUTPUT STATUS
4100                              <1> ;AND   AL, REFRESH_BIT   ;00010000b ; MASK FOR REFRESH DETERMINE BIT
4101                              <1> ;CMP   AL, AH                  ; DID IT JUST CHANGE
4102                              <1> ;JE    short WAITF1            ; WAIT FOR A CHANGE IN OUTPUT LINE
4103                              <1> ;MOV   AH, AL                  ; SAVE NEW FLAG STATE
4104                              <1> ;LOOP  WAITF1                  ; DECREMENT HALF CYCLES TILL COUNT END
4105                              <1>  ;
4106                              <1>  ; 17/12/2014
4107                              <1>  ;
4108                              <1>  ; Modification from 'WAIT_REFRESH' procedure of AWARD BIOS - 1999
4109                              <1>  ;
4110                              <1> ;WAIT_REFRESH:  Uses port 61, bit 4 to have CPU speed independent waiting.
4111                              <1> ;       INPUT:  CX = number of refresh periods to wait
4112                              <1> ;              (refresh periods = 1 per 30 microseconds on most machines)
4113                              <1> WR_STATE_0:
4114 00001630 E461                <1>  IN    AL,PORT_B         ; IN AL,SYS1
4115 00001632 A810                <1>  TEST  AL,010H
4116 00001634 74FA                <1>  JZ    SHORT WR_STATE_0
4117                              <1> WR_STATE_1:
4118 00001636 E461                <1>  IN    AL,PORT_B         ; IN AL,SYS1
4119 00001638 A810                <1>  TEST  AL,010H
4120 0000163A 75FA                <1>  JNZ   SHORT WR_STATE_1
4121 0000163C E2F2                <1>        LOOP    WR_STATE_0
4122                              <1>  ;
4123 0000163E 6658                <1>  POP   AX                ; RESTORE (AH)
4124 00001640 C3                  <1>  RETn                    ; (CX) = 0
4125                              <1>
```

```
                             <1> set_cpos:
                             <1>  ; 27/06/2015
                             <1>  ; 01/09/2014
                             <1>  ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; 12/12/2013 (Retro UNIX 8086 v1 - last update)
                             <1>  ; 04/12/2013 (Retro UNIX 8086 v1 - beginning)
                             <1>  ;
                             <1>  ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>  ;
                             <1>  ; SET_CPOS
                             <1>  ;     THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
                             <1>  ;     NEW X-Y VALUES PASSED
                             <1>  ; INPUT
                             <1>  ;     DX - ROW,COLUMN OF NEW CURSOR
                             <1>  ;     BH - DISPLAY PAGE OF CURSOR
                             <1>  ; OUTPUT
                             <1>  ;     CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
                             <1>  ;
00001641 0FB6C3              <1>         movzx   eax, bl  ; BL = video page number ; 27/06/2015 (movzx)
00001644 D0E0                <1>         shl     al, 1   ; word offset
00001646 BE[C6700000]        <1>  mov   esi, cursor_posn
0000164B 01C6                <1>         add     esi, eax
0000164D 668916              <1>  mov   [esi], dx ; save the pointer
00001650 381D[D6700000]      <1>  cmp   [active_page], bl
00001656 7532                <1>  jne   short m17
                             <1>  ;call m18   ; CURSOR SET
                             <1> ;m17:              ; SET_CPOS_RETURN
                             <1>  ; 01/09/2014
                             <1> ;retn
                             <1>        ; DX  = row/column
                             <1> m18:
00001658 E832000000          <1>  call  position ; determine location in regen buffer
0000165D 668B0D[C4700000]    <1>  mov   cx, [CRT_START]
00001664 6601C1              <1>  add   cx, ax  ; add char position in regen buffer
                             <1>              ; to the start address (offset) for this page
00001667 66D1E9              <1>  shr   cx, 1 ; divide by 2 for char only count
0000166A B40E                <1>  mov   ah, 14      ; register number for cursor
                             <1>  ;call m16   ; output value to the 6845
                             <1>  ;retn
                             <1>
                             <1>  ;-----      THIS ROUTINE OUTPUTS THE CX REGISTER
                             <1>  ;     TO THE 6845 REGISTERS NAMED IN (AH)
                             <1> m16:
0000166C FA                  <1>  cli
                             <1>  ;mov  dx, [addr_6845] ; address register
0000166D 66BAD403            <1>  mov   dx, 03D4h ; I/O address of color card
00001671 88E0                <1>  mov   al, ah      ; get value
00001673 EE                  <1>  out   dx, al      ; register set
00001674 6642                <1>  inc   dx    ; data register
00001676 EB00                <1>  jmp   $+2   ; i/o delay
00001678 88E8                <1>  mov   al, ch      ; data
0000167A EE                  <1>  out   dx, al
0000167B 664A                <1>  dec   dx
0000167D 88E0                <1>  mov   al, ah
0000167F FEC0                <1>  inc   al    ; point to other data register
00001681 EE                  <1>  out   dx, al      ; set for second register
00001682 6642                <1>  inc   dx
00001684 EB00                <1>  jmp   $+2   ; i/o delay
00001686 88C8                <1>  mov   al, cl      ; second data value
00001688 EE                  <1>  out   dx, al
00001689 FB                  <1>  sti
                             <1> m17:
0000168A C3                  <1>  retn
                             <1>
                             <1>
                             <1> set_ctype:
                             <1>  ; 02/09/2014 (Retro UNIX 386 v1)
                             <1>  ;
                             <1>  ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>
                             <1> ;CH) = BITS 4-0 = START LINE FOR CURSOR
                             <1> ;       ** HARDWARE WILL ALWAYS CAUSE BLINK
                             <1> ;       ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
                             <1> ;          OR NO CURSOR AT ALL
                             <1> ;(CL) = BITS 4-0 = END LINE FOR CURSOR
                             <1>
                             <1> ;----------------------------------------------
                             <1> ; SET_CTYPE
                             <1> ;THIS ROUTINE SETS THE CURSOR VALUE
                             <1> ; INPUT
                             <1> ;(CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                             <1> ; OUTPUT
                             <1> ;NONE
                             <1> ;----------------------------------------------
                             <1>
0000168B B40A                <1>  mov   ah, 10      ; 6845 register for cursor set
                             <1>  ;mov  [CURSOR_MODE], cx ; save in data area
                             <1>  ;call m16   ; output cx register
                             <1>  ;retn
0000168D EBDD                <1>  jmp   m16
                             <1>
                             <1>
                             <1> position:
                             <1>  ; 27/06/2015
                             <1>  ; 02/09/2014
                             <1>  ; 30/08/2014 (Retro UNIX 386 v1)
                             <1>  ; 04/12/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>  ;
                             <1>  ; POSITION
                             <1>  ;     THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
                             <1>  ;     OF A CHARACTER IN THE ALPHA MODE
                             <1>  ; INPUT
                             <1>  ;     AX = ROW, COLUMN POSITION
                             <1>  ; OUTPUT
                             <1>  ;     AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
                             <1>
                             <1>        ; DX = ROW, COLUMN POSITION
                             <1>  ;movzx      eax, byte [CRT_COLS] ; 27/06/2015
0000168F 31C0                <1>  xor   eax, eax ; 02/09/2014
00001691 B050                <1>  mov   al, 80   ; determine bytes to row
00001693 F6E6                <1>  mul   dh ;   row value
00001695 30F6                <1>  xor   dh, dh   ; 0
00001697 6601D0              <1>  add   ax, dx       ; add column value to the result
0000169A 66D1E0              <1>  shl   ax, 1 ; * 2 for attribute bytes
                             <1>        ; EAX = AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
0000169D C3                  <1>  retn
                             <1>
                             <1> find_position:
                             <1>  ; 27/06/2015
                             <1>  ; 07/09/2014
                             <1>  ; 02/09/2014
                             <1>  ; 30/08/2014 (Retro UNIX 386 v1)
                             <1>  ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
0000169E 0FB6CB              <1>  movzx ecx, bl ; video page number ; 27/06/2015 (movzx)
000016A1 89CE                <1>  mov   esi, ecx
000016A3 66D1E6              <1>  shl   si, 1
000016A6 668B96[C6700000]    <1>  mov   dx, [esi + cursor_posn]
000016AD 740A                <1>  jz    short p21
000016AF 6631F6              <1>  xor   si, si
                             <1> p20:
                             <1>  ;add  si, [CRT_LEN]
000016B2 6681C6A00F          <1>  add   si, 80*25*2 ; add length of buffer for one page
000016B7 E2F9                <1>  loop  p20
                             <1> p21:
000016B9 6621D2              <1>  and   dx, dx
000016BC 7407                <1>  jz    short p22
000016BE E8CCFFFFFF          <1>  call  position ; determine location in regen in page
000016C3 01C6                <1>  add   esi, eax ; add location to start of regen page
                             <1> p22:
                             <1>  ;mov  dx, [addr_6845] ; get base address of active display
                             <1>  ;mov  dx, 03D4h ; I/O address of color card
                             <1>  ;add  dx, 6 ; point at status port
000016C5 66BADA03            <1>  mov   dx, 03DAh ; status port
                             <1>  ; cx = 0
000016C9 C3                  <1>  retn
                             <1>
                             <1> scroll_up:
                             <1>  ; 07/09/2014
                             <1>  ; 02/09/2014
                             <1>  ; 01/09/2014 (Retro UNIX 386 v1 - beginning)
                             <1>  ; 04/04/2014
                             <1>  ; 04/12/2013
                             <1>  ;
                             <1>  ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>  ;
                             <1>  ; SCROLL UP
                             <1>  ;     THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                             <1>  ;     ON THE SCREEN
                             <1>  ; INPUT
                             <1>  ;      (AH) = CURRENT CRT MODE
                             <1>  ;      (AL) = NUMBER OF ROWS TO SCROLL
                             <1>  ;      (CX) = ROW/COLUMN OF UPPER LEFT CORNER
                             <1>  ;      (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
                             <1>  ;      (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
                             <1>  ;      (DS) = DATA SEGMENT
                             <1>  ;      (ES) = REGEN BUFFER SEGMENT
                             <1>  ; OUTPUT
                             <1>  ;     NONE -- THE REGEN BUFFER IS MODIFIED
                             <1>  ;
                             <1>  ;     bh = 0  (02/09/2014)
                             <1>  ;
                             <1>  ; ((ah = 3))
                             <1>  ; cl = left upper column
                             <1>  ; ch = left upper row
                             <1>  ; dl = right lower column
                             <1>  ; dh = right lower row
                             <1>  ;
                             <1>  ; al = line count
                             <1>  ; ah = attribute to be used on blanked line
                             <1>  ; bl = video page number (0 to 7)
                             <1>  ;
                             <1>
                             <1>  ; Test      Line Count
000016CA 08C0                <1>  or    al, al
000016CC 740C                <1>  jz    short al_set
000016CE 88F7                <1>  mov   bh, dh      ; subtract lower row from upper row
000016D0 28EF                <1>  sub   bh, ch
000016D2 FEC7                <1>  inc   bh    ; adjust difference by 1
000016D4 38C7                <1>  cmp   bh, al      ; line count = amount of rows in window?
000016D6 7502                <1>  jne   short al_set ; if not the we're all set
000016D8 30C0                <1>  xor   al, al      ; otherwise set al to zero
                             <1> al_set:
000016DA 30FF                <1>  xor   bh, bh      ; 0
000016DC 6650                <1>  push  ax
                             <1>  ;mov  esi, [crt_base]
000016DE BE00800B00          <1>         mov     esi, 0B8000h
000016E3 3A1D[D6700000]      <1>         cmp     bl, [active_page]
000016E9 750B                <1>  jne   short n0
                             <1>  ;
000016EB 66A1[C4700000]      <1>         mov     ax, [CRT_START]
000016F1 6601C6              <1>         add     si, ax
000016F4 EB0F                <1>         jmp     short n1
                             <1> n0:
000016F6 20DB                <1>         and     bl, bl
000016F8 740B                <1>  jz    short n1
000016FA 88D8                <1>  mov   al, bl
                             <1> n0x:
                             <1>         ;add    si, [CRT_LEN]
                             <1>         ;add    esi, 80*25*2
000016FC 6681C6A00F          <1>         add     si, 80*25*2
00001701 FEC8                <1>         dec  al
00001703 75F7                <1>  jnz   short n0x
                             <1> n1:
                             <1>         ;Scroll position
00001705 6652                <1>  push  dx
00001707 6689CA              <1>  mov   dx, cx      ; now, upper left position in DX
0000170A E880FFFFFF          <1>  call  position
0000170F 01C6                <1>  add   esi, eax
00001711 89F7                <1>  mov   edi, esi
00001713 665A                <1>  pop   dx    ; lower right position in DX
00001715 6629CA              <1>  sub   dx, cx
00001718 FEC6                <1>  inc   dh    ; dh = #rows
0000171A FEC2                <1>  inc   dl    ; dl = #cols in block
0000171C 6658                <1>  pop   ax    ; al = line count, ah = attribute
0000171E 31C9                <1>  xor   ecx, ecx
00001720 6689C1              <1>  mov   cx, ax
                             <1>  ;mov  ah, [CRT_COLS]
00001723 B450                <1>  mov   ah, 80
00001725 F6E4                <1>  mul   ah    ; determine offset to from address
00001727 6601C0              <1>  add   ax, ax  ; *2 for attribute byte
                             <1>  ;
0000172A 6650                <1>  push  ax    ; offset
0000172C 6652                <1>  push  dx
                             <1>  ;
                             <1>  ; 04/04/2014
0000172E 66BADA03            <1>  mov   dx, 3DAh ; guaranteed to be color card here
                             <1> n8:                       ; wait_display_enable
00001732 EC                  <1>         in      al, dx   ; get port
00001733 A808                <1>  test  al, RVRT ; wait for vertical retrace
00001735 74FB                <1>  jz    short n8 ; wait_display_enable
00001737 B025                <1>  mov   al, 25h
00001739 B2D8                <1>  mov   dl, 0D8h ; address control port
0000173B EE                  <1>  out   dx, al      ; turn off video during vertical retrace
0000173C 665A                <1>  pop   dx    ; #rows, #cols
0000173E 6658                <1>         pop   ax    ; offset
00001740 6691                <1>  xchg  ax, cx      ;
                             <1>  ; ecx = offset, al = line count, ah = attribute
                             <1> ;n9:
00001742 08C0                <1>  or    al, al
00001744 7420                <1>         jz      short n3
00001746 01CE                <1>         add     esi, ecx ; from address for scroll
00001748 88F7                <1>  mov   bh, dh  ; #rows in block
0000174A 28C7                <1>  sub   bh, al      ; #rows to be moved
                             <1> n2:
                             <1>  ; Move rows
0000174C 88D1                <1>  mov   cl, dl      ; get # of cols to move
0000174E 56                  <1>  push  esi
0000174F 57                  <1>  push  edi   ; save start address
                             <1> n10:
00001750 66A5                <1>  movsw       ; move that line on screen
00001752 FEC9                <1>  dec   cl
00001754 75FA                <1>         jnz     short n10
00001756 5F                  <1>  pop   edi
00001757 5E                  <1>  pop   esi   ; recover addresses
                             <1>         ;mov    cl, [CRT_COLS]
                             <1>  ;add  cl, cl
                             <1>         ;mov    ecx, 80*2
00001758 66B9A000            <1>         mov     cx, 80*2
0000175C 01CE                <1>         add     esi, ecx  ; next line
0000175E 01CF                <1>         add     edi, ecx
00001760 FECF                <1>  dec   bh     ; count of lines to move
00001762 75E8                <1>  jnz   short n2 ; row loop
                             <1>  ; bh = 0
```

```
00001764 88C6                <1>  mov   dh, al       ; #rows
                             <1> n3:
                             <1>  ; attribute in ah
00001766 B020                <1>  mov   al, ' '      ; fill with blanks
                             <1>  ; Clear rows
                             <1>                  ; dh =  #rows
00001768 88D1                <1>         mov  cl, dl      ; get # of cols to clear
0000176A 57                  <1>         push    edi     ; save address
                             <1> n11:
0000176B 66AB                <1>         stosw           ; store fill character
0000176D FEC9                <1>  dec   cl
0000176F 75FA                <1>         jnz     short n11
00001771 5F                  <1>         pop     edi     ; recover address
                             <1>  ;mov  cl, [CRT_COLS]
                             <1>  ;add  cl, cl
                             <1>         ;mov    ecx, 80*2
00001772 B1A0                <1>         mov  cl, 80*2
00001774 01CE                <1>         add     esi, ecx  ; next line
00001776 01CF                <1>         add     edi, ecx
00001778 FECE                <1>  dec   dh
0000177A 75EA                <1>  jnz   short n3
                             <1>  ;
0000177C 3A1D[D6700000]      <1>  cmp   bl, [active_page]
00001782 7507                <1>  jne   short n6
                             <1>  ;mov  al, [CRT_MODE_SET] ; get the value of mode set
00001784 B029                <1>  mov   al, 29h ; (ORGS.ASM), M7 mode set table value for mode 3
00001786 66BAD803            <1>  mov   dx, 03D8h ; always set color card port
0000178A EE                  <1>  out   dx, al
                             <1> n6:
0000178B C3                  <1>  retn
                             <1>
                             <1>
                             <1> write_c_current:
                             <1>  ; 30/08/2014 (Retro UNIX 386 v1)
                             <1>  ; 18/01/2014
                             <1>  ; 04/12/2013
                             <1>  ;
                             <1>  ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>  ;
                             <1>  ; WRITE_C_CURRENT
                             <1>  ;     THIS ROUTINE WRITES THE CHARACTER AT
                             <1>  ;     THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
                             <1>  ; INPUT
                             <1>  ;     (AH) = CURRENT CRT MODE
                             <1>  ;     (BH) = DISPLAY PAGE
                             <1>  ;     (CX) = COUNT OF CHARACTERS TO WRITE
                             <1>  ;     (AL) = CHAR TO WRITE
                             <1>  ;     (DS) = DATA SEGMENT
                             <1>  ;     (ES) = REGEN SEGMENT
                             <1>  ; OUTPUT
                             <1>  ;     DISPLAY REGEN BUFFER UPDATED
                             <1>
0000178C FA                  <1>  cli
                             <1>  ; bl = video page
                             <1>  ; al = character
                             <1>  ; ah = color/attribute
0000178D 6652                <1>  push  dx
0000178F 6650                <1>  push  ax    ; save character & attribute/color
00001791 E808FFFFFF          <1>  call  find_position  ; get regen location and port address
                             <1>  ; esi = regen location
                             <1>  ; dx = status port
                             <1>  ;
                             <1>  ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE
                             <1>  ;
                             <1> p41:               ; wait for horizontal retrace is low or vertical
00001796 FB                  <1>  sti         ; enable interrupts first
00001797 3A1D[D6700000]      <1>         cmp     bl, [active_page]
0000179D 7510                <1>  jne   short p44
0000179F FA                  <1>  cli         ; block interrupts for single loop
000017A0 EC                  <1>  in    al, dx      ; get status from the adapter
000017A1 A808                <1>  test  al, RVRT ; check for vertical retrace first
000017A3 7509                <1>  jnz   short p43 ; Do fast write now if vertical retrace
000017A5 A801                <1>  test  al, RHRZ ; is horizontal retrace low
000017A7 75ED                <1>  jnz   short p41 ; wait until it is
                             <1> p42:              ;  wait for either retrace high
000017A9 EC                  <1>  in    al, dx ; get status again
000017AA A809                <1>  test  al, RVRT+RHRZ ; is horizontal or vertical retrace high
000017AC 74FB                <1>  jz    short p42 ; wait until either retrace active
                             <1> p43:
000017AE FB                  <1>  sti
                             <1> p44:
000017AF 6658                <1>  pop   ax    ; restore the character (al) & attribute (ah)
000017B1 81C600800B00        <1>  add   esi, 0B8000h ; 30/08/2014 (crt_base)
                             <1>                ; Retro UNIX 386 v1 feature only!
000017B7 668906              <1>  mov   [esi], ax
000017BA 665A                <1>  pop   dx
000017BC C3                  <1>  retn
                             <1>
                             <1> set_mode:
                             <1>  ; 02/09/2014 (Retro UNIX 386 v1)
                             <1>  ;
                             <1>  ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                             <1>
                             <1> ;-----------------------------------------------------
                             <1> ; SET MODE                                          :
                             <1> ;THIS ROUTINE INITIALIZES THE ATTACHMENT TO         :
                             <1> ;THE SELECTED MODE, THE SCREEN IS BLANKED.          :
                             <1> ; INPUT                                             :
                             <1> ;(AL) - MODE SELECTED (RANGE 0-7)                   :
                             <1> ; OUTPUT                                            :
                             <1> ;NONE                                      :
                             <1> ;-----------------------------------------------------
                             <1>
000017BD 53                  <1>  push  ebx
000017BE 52                  <1>  push  edx
000017BF 50                  <1>         push    eax
                             <1>
                             <1>  ;mov  dx, 03D4h   ; address or color card
000017C0 B003                <1>  mov   al, 3
                             <1> ;M8:
000017C2 A2[906A0000]        <1>  mov   [CRT_MODE], al  ; save mode in global variable
000017C7 B029                <1>  mov   al, 29h
                             <1>  ;mov  [CRT_MODE_SET], al ; save the mode set value
000017C9 2437                <1>  and   al, 037h    ; video off, save high resolution bit
                             <1>  ;push dx          ; save port value
                             <1>  ;add  dx, 4       ; point to control register
000017CB 66BAD803            <1>  mov   dx, 3D8h
000017CF EE                  <1>  out   dx, al            ; reset video to off to suppress rolling
                             <1>  ;pop  dx
                             <1> ;M9:
000017D0 30E4                <1>  xor   ah, ah
000017D2 BB[C86A0000]        <1>  mov   ebx, video_params ; initialization table
                             <1>  ;mov  ax, [ebx+10]      ; get the cursor mode from the table
                             <1>  ;xchg       ah, al
                             <1>  ;mov  [CURSOR_MODE], ax ; save cursor mode
000017D7 30E4                <1>  xor   ah, ah              ; ah is register number during loop
                             <1>
                             <1> ;----- LOOP THROUGH TABLE, OUTPUTTING REGISTER ADDRESS, THEN VALUE FROM TABLE
                             <1>
                             <1> M10:               ;  initialization loop
000017D9 88E0                <1>  mov   al, ah      ; get 6845 register number
000017DB EE                  <1>  out   dx, al
000017DC 6642                <1>  inc   dx      ; point to data port
000017DE FEC4                <1>  inc   ah    ; next register value
000017E0 8A03                <1>  mov   al, [ebx] ; get table value
000017E2 EE                  <1>  out   dx, al      ; out to chip
000017E3 43                  <1>  inc   ebx   ; next in table
000017E4 664A                <1>  dec   dx    ; back to pointer register
000017E6 E2F1                <1>  loop  M10   ; do the whole table
                             <1>
                             <1> ;----- FILL REGEN AREA WITH BLANK
                             <1>  ;xor  ax, ax
                             <1>  ;mov  [CRT_START], ax  ; start address saved in global
                             <1>  ;mov  [ACTIVE_PAGE], al ; 0 ; (re)set page value
                             <1>  ;mov  ecx, 8192 ; number of words in color card
                             <1>  ; black background, light gray characeter color, space character
                             <1>  ;mov  ax, 0720h ; fill char for alpha - attribute
                             <1> ;M13:                ; clear buffer
                             <1>  ;add  edi, 0B8000h ; [crt_base]
                             <1>  ;rep  stosw ; FILL THE REGEN BUFFER WITH BLANKS
                             <1>
                             <1> ;----- ENABLE VIDEO AND CORRECT PORT SETTING
                             <1>  ;mov  dx, 3D4h ; mov dx, word [ADDR_6845]
                             <1>              ; prepare to output to video enable port
                             <1>  ;add  dx,4   ; point to the mode control gerister
000017E8 66BAD803            <1>  mov   dx, 3D8h
                             <1>  ;mov  al, [CRT_MODE_SET] ; get the mode set value
000017EC B029                <1>  mov   al, 29h
000017EE EE                  <1>  out   dx, al       ; set video enable port
                             <1>
                             <1> ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
                             <1> ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
                             <1>  ;
                             <1>  ;mov byte [CRT_COLS], 80h ; initialize number of columns count
                             <1>  ;
                             <1> ;----- SET CURSOR POSITIONS
000017EF 57                  <1>  push  edi
                             <1>  ;mov  word [CRT_LEN], 80*25*2
000017F0 51                  <1>  push  ecx
000017F1 BF[C6700000]        <1>  mov   edi, cursor_posn
000017F6 B904000000          <1>  mov   ecx, 4       ; clear all cursor positions (16 bytes)
000017FB 31C0                <1>  xor   eax, eax
000017FD F3AB                <1>  rep   stosd ; fill with zeroes
000017FF 59                  <1>  pop   ecx
00001800 5F                  <1>  pop   edi
                             <1>
                             <1> ;----- SET UP OVERSCAN REGISTER
00001801 6642                <1>  inc   dx    ; set overscan port to a default
00001803 B030                <1>  mov   al, 30h     ; 30H valuye for all modes except 640X200 bw
                             <1> ;M14:
00001805 EE                  <1>  out   dx, al      ; output the correct value to 3D9 port
                             <1>  ;mov  [CRT_PALETTE], al ; save the value for future use
                             <1>
                             <1> ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
```

```
                              <1>  ;
00001806 58                   <1>  pop   eax
00001807 5A                   <1>  pop   edx
00001808 5B                   <1>  pop   ebx
00001809 C3                   <1>  retn
                              <1>
                              <1> tty_sw:
                              <1>  ; 30/06/2015
                              <1>  ; 27/06/2015
                              <1>  ; 07/09/2014
                              <1>  ; 02/09/2014 (Retro UNIX 386 v1 - beginning)
                              <1>  ;
                              <1>  ; (Modified registers : EAX)
                              <1>  ;
                              <1>         ;mov     byte [u.quant], 0  ; 04/03/2014
                              <1>  ;
                              <1> ;act_disp_page:
                              <1>  ; 30/06/2015
                              <1>  ; 04/03/2014  (act_disp_page --> tty_sw)
                              <1>  ; 10/12/2013
                              <1>  ; 04/12/2013
                              <1>  ;
                              <1>  ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
                              <1>  ;
                              <1>  ; ACT_DISP_PAGE
                              <1>  ;     THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
                              <1>  ;     THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
                              <1>  ; INPUT
                              <1>  ;     AL HAS THE NEW ACTIVE DISPLAY PAGE
                              <1>  ; OUTPUT
                              <1>  ;     THE 6845 IS RESET TO DISPLAY THAT PAGE
                              <1>
                              <1>  ;cli
                              <1>
0000180A 53                   <1>  push  ebx
0000180B 6651                 <1>  push  cx
0000180D 6652                 <1>  push  dx
                              <1>  ;
0000180F A2[D6700000]         <1>  mov   [active_page], al ; save active page value ; [ptty]
                              <1>  ;mov  cx, [CRT_LEN] ; get saved length of regen buffer
00001814 66B9A00F             <1>  mov   cx, 25*80*2
                              <1>  ; 27/06/2015
00001818 0FB6D8               <1>  movzx ebx, al
                              <1>  ;
0000181B 6698                 <1>  cbw   ; 07/09/2014 (ah=0)
0000181D 66F7E1               <1>  mul   cx    ; display page times regen length
                              <1>  ; 10/12/2013
00001820 66A3[C4700000]       <1>  mov   [CRT_START], ax ; save start address for later
00001826 6689C1               <1>  mov   cx, ax ; start address to cx
                              <1>  ;sar  cx, 1
00001829 66D1E9               <1>  shr   cx, 1 ; divide by 2 for 6845 handling
0000182C B40C                 <1>  mov   ah, 12      ; 6845 register for start address
0000182E E839FEFFFF           <1>  call  m16
                              <1>  ;sal  bx, 1
                              <1>  ; 01/09/2014
00001833 D0E3                 <1>  shl   bl, 1 ; *2 for word offset
00001835 81C3[C6700000]       <1>  add   ebx, cursor_posn
0000183B 668B13               <1>  mov   dx, [ebx] ; get cursor for this page
0000183E E815FEFFFF           <1>  call  m18
                              <1>  ;
00001843 665A                 <1>  pop   dx
00001845 6659                 <1>  pop   cx
00001847 5B                   <1>  pop   ebx
                              <1>  ;
                              <1>  ;sti
                              <1>  ;
00001848 C3                   <1>  retn
                              <1>
                              <1> ; % include 'vidata.inc' ; VIDEO DATA ; 11/03/2015
                              <1>
                              <1>
                              <1> ; /// End Of VIDEO FUNCTIONS ///

                                  setup_rtc_int:
                                  ; source: http://wiki.osdev.org/RTC
00001849 FA                        cli         ; disable interrupts
                                   ; default int frequency is 1024 Hz (Lower 4 bits of register A is 0110b or 6)
                                   ; in order to change this ...
                                   ; frequency  = 32768 >> (rate-1) --> 32768 >> 5 = 1024
                                   ; (rate must be above 2 and not over 15)
                                   ; new rate = 15 --> 32768 >> (15-1) = 2 Hz
0000184A B08A                      mov   al, 8Ah
0000184C E670                      out   70h, al ; set index to register A, disable NMI
0000184E 90                        nop
0000184F E471                      in    al, 71h ; get initial value of register A
00001851 88C4                      mov   ah, al
00001853 80E4F0                    and   ah, 0F0h
00001856 B08A                      mov   al, 8Ah
00001858 E670                      out   70h, al ; reset index to register A
0000185A 88E0                      mov   al, ah
0000185C 0C0F                      or    al, 0Fh     ; new rate (0Fh -> 15)
0000185E E671                      out   71h, al ; write only our rate to A. Note, rate is the bottom 4 bits.
```

```
                                 ; enable RTC interrupt
00001860 B08B                    mov   al, 8Bh ;
00001862 E670                    out   70h, al ; select register B and disable NMI
00001864 90                      nop
00001865 E471                    in    al, 71h ; read the current value of register B
00001867 88C4                    mov   ah, al  ;
00001869 B08B                    mov   al, 8Bh ;
0000186B E670                    out   70h, al ; set the index again (a read will reset the index to register B)

0000186D 88E0                    mov   al, ah  ;
0000186F 0C40                    or    al, 40h ;
00001871 E671                    out   71h, al ; write the previous value ORed with 0x40. This turns on bit 6 of
register B
00001873 FB                      sti
00001874 C3                      retn

                                ; Write memory information
                                ; Temporary Code
                                ; 06/11/2014
                                ; 14/08/2015
                                memory_info:
00001875 A1[AC700000]            mov   eax, [memory_size] ; in pages
0000187A 50                      push  eax
0000187B C1E00C                  shl   eax, 12              ; in bytes
0000187E BB0A000000              mov   ebx, 10
00001883 89D9                    mov   ecx, ebx      ; 10
00001885 BE[AD6C0000]            mov   esi, mem_total_b_str
0000188A E8B2000000              call  bintdstr
0000188F 58                      pop   eax
00001890 B107                    mov   cl, 7
00001892 BE[D16C0000]            mov   esi, mem_total_p_str
00001897 E8A5000000              call  bintdstr
                                 ; 14/08/2015
0000189C E8BD000000              call  calc_free_mem
                                 ; edx = calculated free pages
                                 ; ecx = 0
000018A1 A1[B0700000]            mov   eax, [free_pages]
000018A6 39D0                    cmp   eax, edx ; calculated free mem value
                                       ; and initial free mem value are same or not?
000018A8 751D                    jne   short pmim ; print mem info with '?' if not
000018AA 52                      push  edx ; free memory in pages
                                 ;mov  eax, edx
000018AB C1E00C                  shl   eax, 12 ; convert page count
                                           ; to byte count
000018AE B10A                    mov   cl, 10
000018B0 BE[F16C0000]            mov   esi, free_mem_b_str
000018B5 E887000000              call  bintdstr
000018BA 58                      pop   eax
000018BB B107                    mov   cl, 7
000018BD BE[156D0000]            mov   esi, free_mem_p_str
000018C2 E87A000000              call  bintdstr
                                pmim:
000018C7 BE[9B6C0000]            mov   esi, msg_memory_info
                                pmim_nb:
000018CC AC                      lodsb
000018CD 08C0                    or    al, al
000018CF 740D                    jz    short pmim_ok
000018D1 56                      push  esi
000018D2 31DB                    xor   ebx, ebx ; 0
                                           ; Video page 0 (bl=0)
000018D4 B407                    mov   ah, 07h ; Black background,
                                           ; light gray forecolor
000018D6 E842FCFFFF              call  write_tty
000018DB 5E                      pop   esi
000018DC EBEE                    jmp   short pmim_nb
                                pmim_ok:
000018DE C3                      retn

                                ; Convert binary number to hexadecimal string
                                ; 10/05/2015
                                ; dsectpm.s (28/02/2015)
                                ; Retro UNIX 386 v1 - Kernel v0.2.0.6
                                ; 01/12/2014
                                ; 25/11/2014
                                ;
                                bytetohex:
                                 ; INPUT ->
                                 ;     AL = byte (binary number)
                                 ; OUTPUT ->
                                 ;     AX = hexadecimal string
                                 ;
000018DF 53                      push  ebx
000018E0 31DB                    xor   ebx, ebx
000018E2 88C3                    mov   bl, al
000018E4 C0EB04                  shr   bl, 4
000018E7 8A9B[31190000]          mov   bl, [ebx+hexchrs]
000018ED 86D8                    xchg  bl, al
000018EF 80E30F                  and   bl, 0Fh
000018F2 8AA3[31190000]          mov   ah, [ebx+hexchrs]
000018F8 5B                      pop   ebx
000018F9 C3                      retn
```

```
                                 wordtohex:
                                  ; INPUT ->
                                  ;     AX = word (binary number)
                                  ; OUTPUT ->
                                  ;     EAX = hexadecimal string
                                  ;
000018FA 53                       push  ebx
000018FB 31DB                     xor   ebx, ebx
000018FD 86E0                     xchg  ah, al
000018FF 6650                     push  ax
00001901 88E3                     mov   bl, ah
00001903 C0EB04                   shr   bl, 4
00001906 8A83[31190000]           mov   al, [ebx+hexchrs]
0000190C 88E3                     mov   bl, ah
0000190E 80E30F                   and   bl, 0Fh
00001911 8AA3[31190000]           mov   ah, [ebx+hexchrs]
00001917 C1E010                   shl   eax, 16
0000191A 6658                     pop   ax
0000191C 5B                       pop   ebx
0000191D EBC0                     jmp   short bytetohex
                                  ;mov  bl, al
                                  ;shr  bl, 4
                                  ;mov  bl, [ebx+hexchrs]
                                  ;xchg bl, al
                                  ;and  bl, 0Fh
                                  ;mov  ah, [ebx+hexchrs]
                                  ;pop  ebx
                                  ;retn

                                 dwordtohex:
                                  ; INPUT ->
                                  ;     EAX = dword (binary number)
                                  ; OUTPUT ->
                                  ;     EDX:EAX = hexadecimal string
                                  ;
0000191F 50                       push  eax
00001920 C1E810                   shr   eax, 16
00001923 E8D2FFFFFF               call  wordtohex
00001928 89C2                     mov   edx, eax
0000192A 58                       pop   eax
0000192B E8CAFFFFFF               call  wordtohex
00001930 C3                       retn

                                 ; 10/05/2015
                                 hex_digits:
                                 hexchrs:
00001931 303132333435363738-      db '0123456789ABCDEF'
0000193A 39414243444546

                                 ; Convert binary number to decimal/numeric string
                                 ; 06/11/2014
                                 ; Temporary Code
                                 ;

                                 bintdstr:
                                  ; EAX = binary number
                                  ; ESI = decimal/numeric string address
                                  ; EBX = divisor (10)
                                  ; ECX = string length (<=10)
00001941 01CE                     add   esi, ecx
                                 btdstr0:
00001943 4E                       dec   esi
00001944 31D2                     xor   edx, edx
00001946 F7F3                     div   ebx
00001948 80C230                   add   dl, 30h
0000194B 8816                     mov   [esi], dl
0000194D FEC9                     dec   cl
0000194F 740C                     jz    btdstr2
00001951 09C0                     or    eax, eax
00001953 75EE                     jnz   short btdstr0
                                 btdstr1:
00001955 4E                       dec   esi
00001956 C60620                          mov     byte [esi], 20h ; blank space
00001959 FEC9                     dec   cl
0000195B 75F8                     jnz   short btdstr1
                                 btdstr2:
0000195D C3                       retn

                                 ; Calculate free memory pages on M.A.T.
                                 ; 06/11/2014
                                 ; Temporary Code
                                 ;

                                 calc_free_mem:
0000195E 31D2                     xor   edx, edx
                                  ;xor ecx, ecx
00001960 668B0D[C0700000]         mov   cx, [mat_size] ; in pages
00001967 C1E10A                   shl   ecx, 10     ; 1024 dwords per page
0000196A BE00001000               mov   esi, MEM_ALLOC_TBL
                                 cfm0:
0000196F AD                       lodsd
00001970 51                       push  ecx
```

```
00001971 B920000000                 mov   ecx, 32
                                   cfm1:
00001976 D1E8                       shr   eax, 1
00001978 7301                       jnc   short cfm2
0000197A 42                         inc   edx
                                   cfm2:
0000197B E2F9                       loop  cfm1
0000197D 59                         pop   ecx
0000197E E2EF                       loop  cfm0
00001980 C3                         retn

                                   %include 'diskio.inc'  ; 07/03/2015
                               <1> ; Retro UNIX 386 v1 Kernel - DISKIO.INC
                               <1> ; Last Modification: 22/08/2015
                               <1> ;      (Initialized Disk Parameters Data is in 'DISKDATA.INC')
                               <1> ;      (Uninitialized Disk Parameters Data is in 'DISKBSS.INC')
                               <1>
                               <1> ; DISK I/O SYSTEM - Erdogan Tan (Retro UNIX 386 v1 project)
                               <1>
                               <1> ; ///////// DISK I/O SYSTEM ///////////////
                               <1>
                               <1> ; 06/02/2015
                               <1> diskette_io:
00001981 9C                    <1>  pushfd
00001982 0E                    <1>  push  cs
00001983 E809000000            <1>  call  DISKETTE_IO_1
00001988 C3                    <1>  retn
                               <1>
                               <1> ;;;;;; DISKETTE I/O ;;;;;;;;;;;;;;;;;;;; 06/02/2015 ;;;
                               <1> ;//////////////////////////////////////////////////////
                               <1>
                               <1> ; DISKETTE I/O - Erdogan Tan (Retro UNIX 386 v1 project)
                               <1> ; 20/02/2015
                               <1> ; 06/02/2015 (unix386.s)
                               <1> ; 16/12/2014 - 02/01/2015 (dsectrm2.s)
                               <1> ;
                               <1> ; Code (DELAY) modifications - AWARD BIOS 1999 (ADISK.EQU, COMMON.MAC)
                               <1> ;
                               <1> ; ADISK.EQU
                               <1>
                               <1> ;----- Wait control constants
                               <1>
                               <1> ;amount of time to wait while RESET is active.
                               <1>
                               <1> WAITCPU_RESET_ON   EQU   21          ;Reset on must last at least 14us
                               <1>                               ;at 250 KBS xfer rate.
                               <1>                               ;see INTEL MCS, 1985, pg. 5-456
                               <1>
                               <1> WAITCPU_FOR_STATUS EQU   100         ;allow 30 microseconds for
                               <1>                               ;status register to become valid
                               <1>                               ;before re-reading.
                               <1>
                               <1> ;After sending a byte to NEC, status register may remain
                               <1> ;incorrectly set for 24 us.
                               <1>
                               <1> WAITCPU_RQM_LOW          EQU   24          ;number of loops to check for
                               <1>                               ;RQM low.
                               <1>
                               <1> ; COMMON.MAC
                               <1> ;
                               <1> ;Timing macros
                               <1> ;
                               <1>
                               <1> %macro       SIODELAY 0              ; SHORT IODELAY
                               <1>        jmp short $+2
                               <1> %endmacro
                               <1>
                               <1> %macro       IODELAY  0              ; NORMAL IODELAY
                               <1>        jmp short $+2
                               <1>        jmp short $+2
                               <1> %endmacro
                               <1>
                               <1> %macro       NEWIODELAY 0
                               <1>        out   0ebh,al
                               <1> %endmacro
                               <1>
                               <1> ; (According to) AWARD BIOS 1999 - ATORGS.ASM (dw -> equ, db -> equ)
                               <1> ;;; WAIT_FOR_MEM
                               <1> ;WAIT_FDU_INT_LO   equ   017798            ; 2.5 secs in 30 micro units.
                               <1> ;WAIT_FDU_INT_HI   equ   1
                               <1> WAIT_FDU_INT_LH          equ   83334       ; 27/02/2015 (2.5 seconds waiting)
                               <1> ;;; WAIT_FOR_PORT
                               <1> ;WAIT_FDU_SEND_LO  equ   16667       ; .5 secons in 30 us units.
                               <1> ;WAIT_FDU_SEND_HI  equ   0
                               <1> WAIT_FDU_SEND_LH   equ   16667       ; 27/02/2015
                               <1> ;Time to wait while waiting for each byte of NEC results = .5
                               <1> ;seconds.  .5 seconds = 500,000 micros.  500,000/30 = 16,667.
                               <1> ;WAIT_FDU_RESULTS_LO     equ   16667       ; .5 seconds in 30 micro units.
                               <1> ;WAIT_FDU_RESULTS_HI     equ   0
                               <1> WAIT_FDU_RESULTS_LH      equ   16667  ; 27/02/2015
                               <1> ;;; WAIT_REFRESH
                               <1> ;amount of time to wait for head settle, per unit in parameter
```

```
<1> ;table = 1 ms.
<1> WAIT_FDU_HEAD_SETTLE     equ   33          ; 1 ms in 30 micro units.
<1> 
<1> 
<1> ; /////////////// DISKETTE I/O ///////////////
<1> 
<1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - POSTEQU.INC)
<1> 
<1> ;----------------------------------------
<1> ; EQUATES USED BY POST AND BIOS :
<1> ;----------------------------------------
<1> 
<1> ;--------- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS ------------
<1> ;PORT_A      EQU   060H        ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
<1> ;PORT_B      EQU   061H        ; PORT B READ/WRITE DIAGNOSTIC REGISTER
<1> ;REFRESH_BIT EQU   00010000B   ; REFRESH TEST BIT
<1> 
<1> ;----------------------------------------
<1> ; CMOS EQUATES FOR THIS SYSTEM  :
<1> ;-------------------------------------------------------------------------------
<1> ;CMOS_PORT   EQU   070H        ; I/O ADDRESS OF CMOS ADDRESS PORT
<1> ;CMOS_DATA   EQU   071H        ; I/O ADDRESS OF CMOS DATA PORT
<1> ;NMI        EQU   10000000B   ; DISABLE NMI INTERRUPTS MASK -
<1>                          ;  HIGH BIT OF CMOS LOCATION ADDRESS
<1> 
<1> ;---------- CMOS TABLE LOCATION ADDRESS'S ## -----------------------------------
<1> CMOS_DISKETTEEQU   010H        ; DISKETTE DRIVE TYPE BYTE         ;
<1> ;      EQU   011H        ; - RESERVED                       ;C
<1> CMOS_DISK    EQU   012H        ; FIXED DISK TYPE BYTE             ;H
<1> ;      EQU   013H        ; - RESERVED                       ;E
<1> CMOS_EQUIP   EQU   014H        ; EQUIPMENT WORD LOW BYTE          ;C
<1> 
<1> ;---------- DISKETTE EQUATES --------------------------------------------------
<1> INT_FLAG     EQU   10000000B   ; INTERRUPT OCCURRENCE FLAG
<1> DSK_CHG      EQU   10000000B   ; DISKETTE CHANGE FLAG MASK BIT
<1> DETERMINED   EQU   00010000B   ; SET STATE DETERMINED IN STATE BITS
<1> HOME         EQU   00010000B   ; TRACK 0 MASK
<1> SENSE_DRV_ST EQU   00000100B   ; SENSE DRIVE STATUS COMMAND
<1> TRK_SLAP     EQU   030H        ; CRASH STOP (48 TPI DRIVES)
<1> QUIET_SEEK   EQU   00AH        ; SEEK TO TRACK 10
<1> ;MAX_DRV     EQU   2           ; MAX NUMBER OF DRIVES
<1> HD12_SETTLE  EQU   15          ; 1.2 M HEAD SETTLE TIME
<1> HD320_SETTLE EQU   20          ; 320 K HEAD SETTLE TIME
<1> MOTOR_WAIT   EQU   37          ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
<1> 
<1> ;---------- DISKETTE ERRORS ----------------------------------------------------
<1> ;TIME_OUT    EQU   080H        ; ATTACHMENT FAILED TO RESPOND
<1> ;BAD_SEEK    EQU   040H        ; SEEK OPERATION FAILED
<1> BAD_NEC      EQU   020H        ; DISKETTE CONTROLLER HAS FAILED
<1> BAD_CRC      EQU   010H        ; BAD CRC ON DISKETTE READ
<1> MED_NOT_FND  EQU   00CH        ; MEDIA TYPE NOT FOUND
<1> DMA_BOUNDARY EQU   009H        ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
<1> BAD_DMA      EQU   008H        ; DMA OVERRUN ON OPERATION
<1> MEDIA_CHANGE EQU   006H        ; MEDIA REMOVED ON DUAL ATTACH CARD
<1> RECORD_NOT_FND     EQU   004H        ; REQUESTED SECTOR NOT FOUND
<1> WRITE_PROTECTEQU   003H        ; WRITE ATTEMPTED ON WRITE PROTECT DISK
<1> BAD_ADDR_MARKEQU   002H        ; ADDRESS MARK NOT FOUND
<1> BAD_CMD      EQU   001H        ; BAD COMMAND PASSED TO DISKETTE I/O
<1> 
<1> ;---------- DISK CHANGE LINE EQUATES -------------------------------------------
<1> NOCHGLN      EQU   001H        ; NO DISK CHANGE LINE AVAILABLE
<1> CHGLN        EQU   002H        ; DISK CHANGE LINE AVAILABLE
<1> 
<1> ;---------- MEDIA/DRIVE STATE INDICATORS ---------------------------------------
<1> TRK_CAPA     EQU   00000001B   ; 80 TRACK CAPABILITY
<1> FMT_CAPA     EQU   00000010B   ; MULTIPLE FORMAT CAPABILITY (1.2M)
<1> DRV_DET      EQU   00000100B   ; DRIVE DETERMINED
<1> MED_DET      EQU   00010000B   ; MEDIA DETERMINED BIT
<1> DBL_STEP     EQU   00100000B   ; DOUBLE STEP BIT
<1> RATE_MSK     EQU   11000000B   ; MASK FOR CLEARING ALL BUT RATE
<1> RATE_500     EQU   00000000B   ; 500 KBS DATA RATE
<1> RATE_300     EQU   01000000B   ; 300 KBS DATA RATE
<1> RATE_250     EQU   10000000B   ; 250 KBS DATA RATE
<1> STRT_MSK     EQU   00001100B   ; OPERATION START RATE MASK
<1> SEND_MSK     EQU   11000000B   ; MASK FOR SEND RATE BITS
<1> 
<1> ;---------- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY -------------------------
<1> M3D3U        EQU   00000000B   ; 360 MEDIA/DRIVE NOT ESTABLISHED
<1> M3D1U        EQU   00000001B   ; 360 MEDIA,1.2DRIVE NOT ESTABLISHED
<1> M1D1U        EQU   00000010B   ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
<1> MED_UNK      EQU   00000111B   ; NONE OF THE ABOVE
<1> 
<1> ;---------- INTERRUPT EQUATES --------------------------------------------------
<1> ;EOI         EQU   020H        ; END OF INTERRUPT COMMAND TO 8259
<1> ;INTA00      EQU   020H        ; 8259 PORT
<1> INTA01       EQU   021H        ; 8259 PORT
<1> INTB00       EQU   0A0H        ; 2ND 8259
<1> INTB01       EQU   0A1H        ;
<1> 
<1> ;-------------------------------------------------------------------------------
<1> DMA08        EQU   008H        ; DMA STATUS REGISTER PORT ADDRESS
<1> DMA          EQU   000H        ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
```

```
<1> DMA18        EQU   0D0H        ; 2ND DMA STATUS PORT ADDRESS
<1> DMA1         EQU   0C0H        ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
<1> ;-------------------------------------------------------------------------------
<1> ;TIMER       EQU   040H        ; 8254 TIMER - BASE ADDRESS
<1>
<1> ;-------------------------------------------------------------------------------
<1> DMA_PAGE     EQU   081H        ; START OF DMA PAGE REGISTERS
<1>
<1> ; 06/02/2015 (unix386.s, protected mode modifications)
<1> ; (unix386.s <-- dsectrm2.s)
<1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - DSEG.INC)
<1>
<1> ; 10/12/2014
<1> ;
<1> ;int40h:
<1> ;pushf
<1> ;push  cs
<1> ;;cli
<1> ;call  DISKETTE_IO_1
<1> ;retn
<1>
<1> ; DSKETTE ----- 04/21/86 DISKETTE BIOS
<1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
<1> ;
<1>
<1> ;-- INT13H ----------------------------------------------------------------------
<1> ; DISKETTE I/O
<1> ;THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4 INCH 360 KB,
<1> ;1.2 MB, 720 KB AND 1.44 MB DISKETTE DRIVES.
<1> ; INPUT
<1> ;(AH) =  00H RESET DISKETTE SYSTEM
<1> ;      HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
<1> ;      ON ALL DRIVES
<1> ;-------------------------------------------------------------------------------
<1> ;(AH)= 01H  READ THE STATUS OF THE SYSTEM INTO (AH)
<1> ;      @DISKETTE_STATUS FROM LAST OPERATION IS USED
<1> ;-------------------------------------------------------------------------------
<1> ;REGISTERS FOR READ/WRITE/VERIFY/FORMAT
<1> ;(DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
<1> ;(DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
<1> ;(CH) - TRACK NUMBER (NOT VALUE CHECKED)
<1> ;      MEDIA DRIVE TRACK NUMBER
<1> ;      320/360     320/360         0-39
<1> ;      320/360     1.2M      0-39
<1> ;      1.2M  1.2M      0-79
<1> ;      720K  720K      0-79
<1> ;      1.44M 1.44M     0-79
<1> ;(CL) -      SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
<1> ;      MEDIA DRIVE SECTOR NUMBER
<1> ;      320/360     320/360          1-8/9
<1> ;      320/360     1.2M       1-8/9
<1> ;      1.2M  1.2M       1-15
<1> ;      720K  720K       1-9
<1> ;      1.44M 1.44M      1-18
<1> ;(AL)  NUMBER OF SECTORS (NOT VALUE CHECKED)
<1> ;      MEDIA DRIVE MAX NUMBER OF SECTORS
<1> ;      320/360     320/360            8/9
<1> ;      320/360     1.2M          8/9
<1> ;      1.2M  1.2M        15
<1> ;      720K  720K        9
<1> ;      1.44M 1.44M       18
<1> ;
<1> ;(ES:BX) - ADDRESS OF BUFFER (NOT REQUIRED FOR VERIFY)
<1> ;
<1> ;-------------------------------------------------------------------------------
<1> ;(AH)= 02H  READ THE DESIRED SECTORS INTO MEMORY
<1> ;-------------------------------------------------------------------------------
<1> ;(AH)= 03H  WRITE THE DESIRED SECTORS FROM MEMORY
<1> ;-------------------------------------------------------------------------------
<1> ;(AH)= 04H  VERIFY THE DESIRED SECTORS
<1> ;-------------------------------------------------------------------------------
<1> ;(AH)= 05H  FORMAT THE DESIRED TRACK
<1> ;      (ES,BX) MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
<1> ;      FOR THE     TRACK. EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N),
<1> ;      WHERE C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
<1> ;      N= NUMBER OF BYTES PER SECTOR (00=128,01=256,02=512,03=1024),
<1> ;      THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
<1> ;      THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
<1> ;      READ/WRITE ACCESS.
<1> ;      PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
<1> ;      ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
<1> ;      THEN "SET DASD TYPE" (INT 13H, AH = 17H) OR 'SET MEDIA TYPE'
<1> ;      (INT 13H, AH =  18H) MUST BE CALLED TO SET THE DISKETTE TYPE
<1> ;      THAT IS TO BE FORMATTED. IF "SET DASD TYPE" OR "SET MEDIA TYPE"
<1> ;      IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE
<1> ;      MEDIA FORMAT TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
<1> ;
<1> ;      THESE PARAMETERS OF DISK BASE MUST BE CHANGED IN ORDER TO
<1> ;      FORMAT THE FOLLOWING MEDIAS:
<1> ;      ---------------------------------------------
<1> ;      : MEDIA  :     DRIVE      : PARM 1 : PARM 2 :
<1> ;      ---------------------------------------------
```

```
<1> ;      : 320K       : 320K/360K/1.2M :  50H   :   8    :
<1> ;      : 360K       : 320K/360K/1.2M :  50H   :   9    :
<1> ;      : 1.2M       : 1.2M           :  54H   :  15    :
<1> ;      : 720K       : 720K/1.44M     :  50H   :   9    :
<1> ;      : 1.44M      : 1.44M          :  6CH   :  18    :
<1> ;      --------------------------------------------
<1> ;      NOTES: - PARM 1 = GAP LENGTH FOR FORMAT
<1> ;             - PARM 2 = EOT (LAST SECTOR ON TRACK)
<1> ;             - DISK BASE IS POINTED BY DISK POINTER LOCATED
<1> ;             AT ABSOLUTE ADDRESS 0:78.
<1> ;             - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
<1> ;             SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.

<1> ;-------------------------------------------------------------------------------
<1> ; (AH) = 08H READ DRIVE PARAMETERS
<1> ; REGISTERS
<1> ;  INPUT
<1> ;    (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
<1> ;  OUTPUT
<1> ;    (ES:DI) POINTS TO DRIVE PARAMETER TABLE
<1> ;    (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
<1> ;    (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
<1> ;           BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
<1> ;    (DH) - MAXIMUM HEAD NUMBER
<1> ;    (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
<1> ;    (BH) - 0
<1> ;    (BL) - BITS 7 THRU 4 - 0
<1> ;           BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
<1> ;    (AX) - 0
<1> ; UNDER THE FOLLOWING CIRCUMSTANCES:
<1> ;    (1) THE DRIVE NUMBER IS INVALID,
<1> ;    (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
<1> ;    (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
<1> ;    (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
<1> ;    THEN ES,AX,BX,CX,DH,DI=0 ; DL=NUMBER OF DRIVES.
<1> ;    IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.
<1> ;    @DISKETTE_STATUS = 0 AND CY IS RESET.
<1> ;-------------------------------------------------------------------------------
<1> ; (AH)= 15H  READ DASD TYPE
<1> ; OUTPUT REGISTERS
<1> ; (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
<1> ;      00 - DRIVE NOT PRESENT
<1> ;      01 - DISKETTE, NO CHANGE LINE AVAILABLE
<1> ;      02 - DISKETTE, CHANGE LINE AVAILABLE
<1> ;      03 - RESERVED (FIXED DISK)
<1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
<1> ;-------------------------------------------------------------------------------
<1> ; (AH)= 16H  DISK CHANGE LINE STATUS
<1> ; OUTPUT REGISTERS
<1> ; (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
<1> ;        06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
<1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
<1> ;-------------------------------------------------------------------------------
<1> ; (AH)= 17H  SET DASD TYPE FOR FORMAT
<1> ; INPUT REGISTERS
<1> ; (AL) -      00 - NOT USED
<1> ;      01 - DISKETTE 320/360K IN 360K DRIVE
<1> ;      02 - DISKETTE 360K IN 1.2M DRIVE
<1> ;      03 - DISKETTE 1.2M IN 1.2M DRIVE
<1> ;      04 - DISKETTE 720K IN 720K DRIVE
<1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED:
<1> ;        (DO NOT USE WHEN DISKETTE ATTACH CARD USED)
<1> ;-------------------------------------------------------------------------------
<1> ; (AH)= 18H  SET MEDIA TYPE FOR FORMAT
<1> ; INPUT REGISTERS
<1> ; (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM TRACKS
<1> ; (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
<1> ;        BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
<1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHACKED)
<1> ; OUTPUT REGISTERS:
<1> ; (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
<1> ;         UNCHANGED IF (AH) IS NON-ZERO
<1> ; (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
<1> ;      - 01H, CY = 1, FUNCTION IS NOT AVAILABLE
<1> ;      - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
<1> ;      - 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
<1> ;-------------------------------------------------------------------------------
<1> ; DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
<1> ; THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
<1> ; ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
<1> ;      ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
<1> ;      IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
<1> ;      CHANGE ERROR CODE
<1> ;      IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
<1> ;      TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
<1> ; IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
<1> ;
<1> ; DATA VARIABLE -- @DISK_POINTER
<1> ; DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
<1> ;-------------------------------------------------------------------------------
<1> ; OUTPUT FOR ALL FUNCTIONS
<1> ; AH = STATUS OF OPERATION
```

```
                             <1> ;       STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE_STATUS
                             <1> ;       VARIABLE IN THE DATA SEGMENT OF THIS MODULE
                             <1> ; CY = 0      SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
                             <1> ;       TYPE AH=(15)).
                             <1> ; CY = 1      FAILED OPERATION (AH HAS ERROR REASON)
                             <1> ; FOR READ/WRITE/VERIFY
                             <1> ;       DS,BX,DX,CX PRESERVED
                             <1> ; NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
                             <1> ;       ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
                             <1> ;       ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
                             <1> ;       THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
                             <1> ;       PROBLEM IS NOT DUE TO MOTOR START-UP.
                             <1> ;-----------------------------------------------------------------------------
                             <1> ;
                             <1> ; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
                             <1> ;
                             <1> ;   -----------------------------------------------------------------
                             <1> ;   |       |       |       |       |       |       |       |       |
                             <1> ;   |   7   |   6   |   5   |   4   |   3   |   2   |   1   |   0   |
                             <1> ;   |       |       |       |       |       |       |       |       |
                             <1> ;   -----------------------------------------------------------------
                             <1> ; |     |     |     |     |     |     |     |
                             <1> ; |     |     |     |     |     -----------------
                             <1> ; |     |     |     |     |           |
                             <1> ; |     |     |     |    RESERVED           |
                             <1> ; |     |     |     |            PRESENT STATE
                             <1> ; |     |     |     |     000: 360K IN 360K DRIVE UNESTABLISHED
                             <1> ; |     |     |     |     001: 360K IN 1.2M DRIVE UNESTABLISHED
                             <1> ; |     |     |     |     010: 1.2M IN 1.2M DRIVE UNESTABLISHED
                             <1> ; |     |     |     |     011: 360K IN 360K DRIVE ESTABLISHED
                             <1> ; |     |     |     |     100: 360K IN 1.2M DRIVE ESTABLISHED
                             <1> ; |     |     |     |     101: 1.2M IN 1.2M DRIVE ESTABLISHED
                             <1> ; |     |     |     |     110: RESERVED
                             <1> ; |     |     |     |     111: NONE OF THE ABOVE
                             <1> ; |     |     |     |
                             <1> ; |     |     |     ------>     MEDIA/DRIVE ESTABLISHED
                             <1> ; |     |     |
                             <1> ; |     |     -------------->   DOUBLE STEPPING REQUIRED (360K IN 1.2M
                             <1> ; |     |                 DRIVE)
                             <1> ; |     |
                             <1> ;------------------------------>    DATA TRANSFER RATE FOR THIS DRIVE:
                             <1> ;
                             <1> ;                              00: 500 KBS
                             <1> ;                              01: 300 KBS
                             <1> ;                              10: 250 KBS
                             <1> ;                              11: RESERVED
                             <1> ;
                             <1> ;
                             <1> ;-----------------------------------------------------------------------------
                             <1> ; STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)
                             <1> ;-----------------------------------------------------------------------------
                             <1> ; PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)
                             <1> ;-----------------------------------------------------------------------------
                             <1>
                             <1> struc MD
00000000 <res 00000001>      <1>  .SPEC1           resb  1     ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00000001 <res 00000001>      <1>  .SPEC2           resb  1     ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00000002 <res 00000001>      <1>  .OFF_TIM    resb  1     ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00000003 <res 00000001>      <1>  .BYT_SEC    resb  1     ; 512 BYTES/SECTOR
00000004 <res 00000001>      <1>  .SEC_TRK    resb  1     ; EOT (LAST SECTOR ON TRACK)
00000005 <res 00000001>      <1>  .GAP        resb  1     ; GAP LENGTH
00000006 <res 00000001>      <1>  .DTL        resb  1     ; DTL
00000007 <res 00000001>      <1>  .GAP3       resb  1     ; GAP LENGTH FOR FORMAT
00000008 <res 00000001>      <1>  .FIL_BYT    resb  1     ; FILL BYTE FOR FORMAT
00000009 <res 00000001>      <1>  .HD_TIM           resb  1     ; HEAD SETTLE TIME (MILLISECONDS)
0000000A <res 00000001>      <1>  .STR_TIM    resb  1     ; MOTOR START TIME (1/8 SECONDS)
0000000B <res 00000001>      <1>  .MAX_TRK    resb  1     ; MAX. TRACK NUMBER
0000000C <res 00000001>      <1>  .RATE       resb  1     ; DATA TRANSFER RATE
                             <1> endstruc
                             <1>
                             <1> BIT7OFFEQU   7FH
                             <1> BIT7ON EQU   80H
                             <1>
                             <1> ;;int13h: ; 16/02/2015
                             <1> ;; 16/02/2015 - 21/02/2015
                             <1> int40h:
00001989 9C                  <1>  pushfd
0000198A 0E                  <1>  push  cs
0000198B E801000000          <1>  call  DISKETTE_IO_1
00001990 C3                  <1>  retn
                             <1>
                             <1> DISKETTE_IO_1:
                             <1>
00001991 FB                  <1>  STI                     ; INTERRUPTS BACK ON
00001992 55                  <1>  PUSH  eBP               ; USER REGISTER
00001993 57                  <1>  PUSH  eDI               ; USER REGISTER
00001994 52                  <1>  PUSH  eDX               ; HEAD #, DRIVE # OR USER REGISTER
00001995 53                  <1>  PUSH  eBX               ; BUFFER OFFSET PARAMETER OR REGISTER
00001996 51                  <1>  PUSH  eCX               ; TRACK #-SECTOR # OR USER REGISTER
00001997 89E5                <1>  MOV   eBP,eSP                 ; BP     => PARAMETER LIST DEP. ON AH
                             <1>                          ; [BP]   = SECTOR #
                             <1>                          ; [BP+1] = TRACK #
                             <1>                          ; [BP+2] = BUFFER OFFSET
                             <1>                          ; FOR RETURN OF DRIVE PARAMETERS:
                             <1>                          ; CL/[BP] = BITS 7&6 HI BITS OF MAX CYL
                             <1>                          ;         BITS 0-5 MAX SECTORS/TRACK
                             <1>                          ; CH/[BP+1] = LOW 8 BITS OF MAX CYL.
                             <1>                          ; BL/[BP+2] = BITS 7-4 = 0
                             <1>                          ;           BITS 3-0 = VALID CMOS TYPE
                             <1>                          ; BH/[BP+3] = 0
                             <1>                          ; DL/[BP+4] = # DRIVES INSTALLED
                             <1>                          ; DH/[BP+5] = MAX HEAD #
                             <1>                          ; DI/[BP+6] = OFFSET TO DISK BASE
00001999 06                  <1>  push  es ; 06/02/2015
0000199A 1E                  <1>  PUSH  DS                ; BUFFER SEGMENT PARM OR USER REGISTER
0000199B 56                  <1>  PUSH  eSI               ; USER REGISTERS
                             <1>  ;CALL DDS               ; SEGMENT OF BIOS DATA AREA TO DS
                             <1>  ;mov  cx, cs
                             <1>  ;mov  ds, cx
0000199C 66B91000            <1>  mov   cx, KDATA
000019A0 8ED9                <1>         mov     ds, cx
000019A2 8EC1                <1>         mov     es, cx
                             <1>
                             <1>  ;CMP  AH,(FNC_TAE-FNC_TAB)/2  ; CHECK FOR > LARGEST FUNCTION
000019A4 80FC19              <1>  cmp   ah,(FNC_TAE-FNC_TAB)/4 ; 18/02/2015
000019A7 7202                <1>  JB    short OK_FUNC           ; FUNCTION OK
000019A9 B414                <1>  MOV   AH,14H                  ; REPLACE WITH KNOWN INVALID FUNCTION
                             <1> OK_FUNC:
000019AB 80FC01              <1>  CMP   AH,1              ; RESET OR STATUS ?
000019AE 760C                <1>  JBE   short OK_DRV              ; IF RESET OR STATUS DRIVE ALWAYS OK
000019B0 80FC08              <1>  CMP   AH,8              ; READ DRIVE PARMS ?
000019B3 7407                <1>  JZ    short OK_DRV              ; IF SO DRIVE CHECKED LATER
000019B5 80FA01              <1>  CMP   DL,1              ; DRIVES 0 AND 1 OK
000019B8 7602                <1>  JBE   short OK_DRV              ; IF 0 OR 1 THEN JUMP
000019BA B414                <1>  MOV   AH,14H                  ; REPLACE WITH KNOWN INVALID FUNCTION
                             <1> OK_DRV:
000019BC 31C9                <1>  xor   ecx, ecx
                             <1>  ;mov  esi, ecx ; 08/02/2015
000019BE 89CF                <1>  mov   edi, ecx ; 08/02/2015
000019C0 88E1                <1>  MOV   CL,AH             ; CL = FUNCTION
                             <1>  ;XOR  CH,CH             ; CX = FUNCTION
                             <1>  ;SHL  CL, 1             ; FUNCTION TIMES 2
000019C2 C0E102              <1>  SHL   CL, 2 ; 20/02/2015      ; FUNCTION TIMES 4 (for 32 bit offset)
000019C5 BB[FD190000]        <1>  MOV   eBX,FNC_TAB       ; LOAD START OF FUNCTION TABLE
000019CA 01CB                <1>  ADD   eBX,eCX                   ; ADD OFFSET INTO TABLE => ROUTINE
000019CC 88F4                <1>  MOV   AH,DH             ; AX = HEAD #,# OF SECTORS OR DASD TYPE
000019CE 30F6                <1>  XOR   DH,DH             ; DX = DRIVE #
000019D0 6689C6              <1>  MOV   SI,AX             ; SI = HEAD #,# OF SECTORS OR DASD TYPE
000019D3 6689D7              <1>  MOV     DI,DX                   ; DI = DRIVE #
                             <1>  ;
                             <1>  ; 11/12/2014
000019D6 8815[516B0000]      <1>         mov     [cfd], dl               ; current floppy drive (for 'GET_PARM')
                             <1>  ;
000019DC 8A25[2C710000]      <1>  MOV   AH, [DSKETTE_STATUS]    ; LOAD STATUS TO AH FOR STATUS FUNCTION
000019E2 C605[2C710000]00    <1>  MOV   byte [DSKETTE_STATUS],0 ; INITIALIZE FOR ALL OTHERS
                             <1>
                             <1> ;THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
                             <1> ;THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
                             <1> ;FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
                             <1> ;
                             <1> ;      DI    : DRIVE #
                             <1> ;      SI-HI : HEAD #
                             <1> ;      SI-LOW      : # OF SECTORS OR DASD TYPE FOR FORMAT
                             <1> ;      ES    : BUFFER SEGMENT
                             <1> ;      [BP]  : SECTOR #
                             <1> ;      [BP+1]      : TRACK #
                             <1> ;      [BP+2]      : BUFFER OFFSET
                             <1> ;
                             <1> ;ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
                             <1> ;SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
                             <1> ;CONDITION). IN MOST CASES, WHEN CY = 1, @DSKETTE_STATUS CONTAINS THE
                             <1> ;SPECIFIC ERROR CODE.
                             <1> ;
                             <1>                          ; (AH) = @DSKETTE_STATUS
000019E9 FF13                <1>  CALL  dWORD [eBX]       ; CALL THE REQUESTED FUNCTION
000019EB 5E                  <1>  POP   eSI               ; RESTORE ALL REGISTERS
000019EC 1F                  <1>  POP   DS
000019ED 07                  <1>  pop   es    ; 06/02/2015
000019EE 59                  <1>  POP   eCX
000019EF 5B                  <1>  POP   eBX
000019F0 5A                  <1>  POP   eDX
000019F1 5F                  <1>  POP   eDI
000019F2 89E5                <1>  MOV   eBP, eSP
000019F4 50                  <1>  PUSH  eAX
000019F5 9C                  <1>  PUSHFd
000019F6 58                  <1>  POP   eAX
                             <1>  ;MOV  [BP+6], AX
000019F7 89450C              <1>  mov   [ebp+12], eax  ; 18/02/2015, flags
000019FA 58                  <1>  POP   eAX
000019FB 5D                  <1>  POP   eBP
000019FC CF                  <1>  IRETd
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DW --> dd (06/02/2015)
```

```
000019FD [611A0000]          <1> FNC_TABdd    DSK_RESET         ; AH = 00H; RESET
00001A01 [DA1A0000]          <1>  dd    DSK_STATUS        ; AH = 01H; STATUS
00001A05 [EB1A0000]          <1>  dd    DSK_READ          ; AH = 02H; READ
00001A09 [FC1A0000]          <1>  dd    DSK_WRITE         ; AH = 03H; WRITE
00001A0D [0D1B0000]          <1>  dd    DSK_VERF          ; AH = 04H; VERIFY
00001A11 [1E1B0000]          <1>  dd    DSK_FORMAT        ; AH = 05H; FORMAT
00001A15 [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 06H; INVALID
00001A19 [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 07H; INVALID
00001A1D [B01B0000]          <1>  dd    DSK_PARMS         ; AH = 08H; READ DRIVE PARAMETERS
00001A21 [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 09H; INVALID
00001A25 [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 0AH; INVALID
00001A29 [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 0BH; INVALID
00001A2D [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 0CH; INVALID
00001A31 [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 0DH; INVALID
00001A35 [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 0EH; INVALID
00001A39 [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 0FH; INVALID
00001A3D [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 10H; INVALID
00001A41 [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 11H; INVALID
00001A45 [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 12H; INVALID
00001A49 [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 13H; INVALID
00001A4D [A31B0000]          <1>  dd    FNC_ERR                 ; AH = 14H; INVALID
00001A51 [711C0000]          <1>  dd    DSK_TYPE          ; AH = 15H; READ DASD TYPE
00001A55 [9C1C0000]          <1>  dd    DSK_CHANGE        ; AH = 16H; CHANGE STATUS
00001A59 [D61C0000]          <1>  dd    FORMAT_SET        ; AH = 17H; SET DASD TYPE
00001A5D [591D0000]          <1>  dd    SET_MEDIA         ; AH = 18H; SET MEDIA TYPE
                             <1> FNC_TAE EQU     $                        ; END
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DISK_RESET (AH = 00H)
                             <1> ;      RESET THE DISKETTE SYSTEM.
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> DSK_RESET:
00001A61 66BAF203            <1>  MOV   DX,03F2H          ; ADAPTER CONTROL PORT
00001A65 FA                  <1>  CLI                     ; NO INTERRUPTS
00001A66 A0[2A710000]        <1>  MOV   AL,[MOTOR_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION
00001A6B 243F                <1>  AND   AL,00111111B            ; KEEP SELECTED AND MOTOR ON BITS
00001A6D C0C004              <1>  ROL   AL,4              ; MOTOR VALUE TO HIGH NIBBLE
                             <1>                          ; DRIVE SELECT TO LOW NIBBLE
00001A70 0C08                <1>  OR    AL,00001000B            ; TURN ON INTERRUPT ENABLE
00001A72 EE                  <1>  OUT   DX,AL             ; RESET THE ADAPTER
00001A73 C605[29710000]00    <1>  MOV   byte [SEEK_STATUS],0    ; SET RECALIBRATE REQUIRED ON ALL DRIVES
                             <1>  ;JMP  $+2               ; WAIT FOR I/O
                             <1>  ;JMP  $+2               ; WAIT FOR I/O (TO INSURE MINIMUM
                             <1>                          ;      PULSE WIDTH)
                             <1>  ; 19/12/2014
                             <1>  NEWIODELAY
00001A7A E6EB                <2>  out 0ebh,al
                             <1>
                             <1>  ; 17/12/2014
                             <1>  ; AWARD BIOS 1999 - RESETDRIVES (ADISK.ASM)
00001A7C B915000000          <1>  mov   ecx, WAITCPU_RESET_ON   ; cx = 21 -- Min. 14 micro seconds !?
                             <1> wdw1:
                             <1>  NEWIODELAY   ; 27/02/2015
00001A81 E6EB                <2>  out 0ebh,al
00001A83 E2FC                <1>  loop  wdw1
                             <1>  ;
00001A85 0C04                <1>  OR    AL,00000100B            ; TURN OFF RESET BIT
00001A87 EE                  <1>  OUT   DX,AL             ; RESET THE ADAPTER
                             <1>  ; 16/12/2014
                             <1>  IODELAY
00001A88 EB00                <2>  jmp short $+2
00001A8A EB00                <2>  jmp short $+2
                             <1>  ;
                             <1>  ;STI                    ; ENABLE THE INTERRUPTS
00001A8C E8240C0000          <1>  CALL  WAIT_INT          ; WAIT FOR THE INTERRUPT
00001A91 723E                <1>  JC    short DR_ERR            ; IF ERROR, RETURN IT
00001A93 66B9C000            <1>  MOV   CX,11000000B            ; CL = EXPECTED @NEC_STATUS
                             <1> NXT_DRV:
00001A97 6651                <1>  PUSH  CX                ; SAVE FOR CALL
00001A99 B8[CF1A0000]        <1>  MOV   eAX, DR_POP_ERR   ; LOAD NEC_OUTPUT ERROR ADDRESS
00001A9E 50                  <1>  PUSH  eAX               ; "
00001A9F B408                <1>  MOV   AH,08H                  ; SENSE INTERRUPT STATUS COMMAND
00001AA1 E8020B0000          <1>  CALL  NEC_OUTPUT
00001AA6 58                  <1>  POP   eAX               ; THROW AWAY ERROR RETURN
00001AA7 E8390C0000          <1>  CALL  RESULTS                 ; READ IN THE RESULTS
00001AAC 6659                <1>  POP   CX                ; RESTORE AFTER CALL
00001AAE 7221                <1>  JC    short DR_ERR            ; ERROR RETURN
00001AB0 3A0D[2D710000]      <1>  CMP   CL, [NEC_STATUS]  ; TEST FOR DRIVE READY TRANSITION
00001AB6 7519                <1>  JNZ   short DR_ERR            ; EVERYTHING OK
00001AB8 FEC1                <1>  INC   CL                ; NEXT EXPECTED @NEC_STATUS
00001ABA 80F9C3              <1>  CMP   CL,11000011B            ; ALL POSSIBLE DRIVES CLEARED
00001ABD 76D8                <1>  JBE   short NXT_DRV           ; FALL THRU IF 11000100B OR >
                             <1>  ;
00001ABF E852030000          <1>  CALL  SEND_SPEC         ; SEND SPECIFY COMMAND TO NEC
                             <1> RESBAC:
00001AC4 E805090000          <1>  CALL  SETUP_END         ; VARIOUS CLEANUPS
00001AC9 6689F3              <1>  MOV   BX,SI             ; GET SAVED AL TO BL
00001ACC 88D8                <1>  MOV   AL,BL             ; PUT BACK FOR RETURN
00001ACE C3                  <1>  RETn
                             <1> DR_POP_ERR:
```

```
00001ACF 6659                  <1>  POP   CX                ; CLEAR STACK
                               <1> DR_ERR:
00001AD1 800D[2C710000]20      <1>  OR    byte [DSKETTE_STATUS],BAD_NEC ; SET ERROR CODE
00001AD8 EBEA                  <1>  JMP   SHORT RESBAC            ; RETURN FROM RESET
                               <1>
                               <1> ;--------------------------------------------------------------------------------
                               <1> ; DISK_STATUS(AH = 01H)
                               <1> ;DISKETTE STATUS.
                               <1> ;
                               <1> ; ON ENTRY:  AH : STATUS OF PREVIOUS OPERATION
                               <1> ;
                               <1> ; ON EXIT:   AH, @DSKETTE_STATUS, CY REFLECT STATUS OF PREVIOUS OPERATION.
                               <1> ;--------------------------------------------------------------------------------
                               <1> DSK_STATUS:
00001ADA 8825[2C710000]        <1>  MOV   [DSKETTE_STATUS],AH     ; PUT BACK FOR SETUP END
00001AE0 E8E9080000            <1>  CALL  SETUP_END         ; VARIOUS CLEANUPS
00001AE5 6689F3                <1>  MOV   BX,SI             ; GET SAVED AL TO BL
00001AE8 88D8                  <1>  MOV   AL,BL             ; PUT BACK FOR RETURN
00001AEA C3                    <1>  RETn
                               <1>
                               <1> ;--------------------------------------------------------------------------------
                               <1> ; DISK_READ  (AH = 02H)
                               <1> ;DISKETTE READ.
                               <1> ;
                               <1> ; ON ENTRY:  DI    : DRIVE #
                               <1> ;      SI-HI : HEAD #
                               <1> ;      SI-LOW      : # OF SECTORS
                               <1> ;      ES    : BUFFER SEGMENT
                               <1> ;      [BP]  : SECTOR #
                               <1> ;      [BP+1]      : TRACK #
                               <1> ;      [BP+2]      : BUFFER OFFSET
                               <1> ;
                               <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                               <1> ;--------------------------------------------------------------------------------
                               <1>
                               <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
                               <1>
                               <1> DSK_READ:
00001AEB 8025[2A710000]7F      <1>  AND   byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
00001AF2 66B846E6              <1>  MOV   AX,0E646H         ; AX = NEC COMMAND, DMA COMMAND
00001AF6 E825040000            <1>  CALL  RD_WR_VF          ; COMMON READ/WRITE/VERIFY
00001AFB C3                    <1>  RETn
                               <1>
                               <1> ;--------------------------------------------------------------------------------
                               <1> ; DISK_WRITE (AH = 03H)
                               <1> ;DISKETTE WRITE.
                               <1> ;
                               <1> ; ON ENTRY:  DI    : DRIVE #
                               <1> ;      SI-HI : HEAD #
                               <1> ;      SI-LOW      : # OF SECTORS
                               <1> ;      ES    : BUFFER SEGMENT
                               <1> ;      [BP]  : SECTOR #
                               <1> ;      [BP+1]      : TRACK #
                               <1> ;      [BP+2]      : BUFFER OFFSET
                               <1> ;
                               <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                               <1> ;--------------------------------------------------------------------------------
                               <1>
                               <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
                               <1>
                               <1> DSK_WRITE:
00001AFC 66B84AC5              <1>  MOV   AX,0C54AH         ; AX = NEC COMMAND, DMA COMMAND
00001B00 800D[2A710000]80      <1>         OR      byte [MOTOR_STATUS],10000000B ; INDICATE WRITE OPERATION
00001B07 E814040000            <1>  CALL  RD_WR_VF          ; COMMON READ/WRITE/VERIFY
00001B0C C3                    <1>  RETn
                               <1>
                               <1> ;--------------------------------------------------------------------------------
                               <1> ; DISK_VERF  (AH = 04H)
                               <1> ;DISKETTE VERIFY.
                               <1> ;
                               <1> ; ON ENTRY:  DI    : DRIVE #
                               <1> ;      SI-HI : HEAD #
                               <1> ;      SI-LOW      : # OF SECTORS
                               <1> ;      ES    : BUFFER SEGMENT
                               <1> ;      [BP]  : SECTOR #
                               <1> ;      [BP+1]      : TRACK #
                               <1> ;      [BP+2]      : BUFFER OFFSET
                               <1> ;
                               <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                               <1> ;--------------------------------------------------------------------------------
                               <1> DSK_VERF:
00001B0D 8025[2A710000]7F      <1>  AND   byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
00001B14 66B842E6              <1>  MOV   AX,0E642H         ; AX = NEC COMMAND, DMA COMMAND
00001B18 E803040000            <1>  CALL  RD_WR_VF          ; COMMON READ/WRITE/VERIFY
00001B1D C3                    <1>  RETn
                               <1>
                               <1> ;--------------------------------------------------------------------------------
                               <1> ; DISK_FORMAT(AH = 05H)
                               <1> ;DISKETTE FORMAT.
                               <1> ;
                               <1> ; ON ENTRY:  DI    : DRIVE #
                               <1> ;      SI-HI : HEAD #
```

```
                             <1> ;       SI-LOW      : # OF SECTORS
                             <1> ;       ES    : BUFFER SEGMENT
                             <1> ;       [BP]  : SECTOR #
                             <1> ;       [BP+1]      : TRACK #
                             <1> ;       [BP+2]      : BUFFER OFFSET
                             <1> ;       @DISK_POINTER POINTS TO THE PARAMETER TABLE OF THIS DRIVE
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> DSK_FORMAT:
00001B1E E83C030000          <1>  CALL  XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
00001B23 E838050000          <1>  CALL  FMT_INIT          ; ESTABLISH STATE IF UNESTABLISHED
00001B28 800D[2A710000]80    <1>        OR      byte [MOTOR_STATUS], 10000000B ; INDICATE WRITE OPERATION
00001B2F E880050000          <1>  CALL  MED_CHANGE        ; CHECK MEDIA CHANGE AND RESET IF SO
00001B34 725D                <1>        JC      short FM_DON            ; MEDIA CHANGED, SKIP
00001B36 E8DB020000          <1>  CALL  SEND_SPEC         ; SEND SPECIFY COMMAND TO NEC
00001B3B E8E6050000          <1>  CALL  CHK_LASTRATE            ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
00001B40 7405                <1>        JZ      short FM_WR             ; YES, SKIP SPECIFY COMMAND
00001B42 E8BD050000          <1>  CALL  SEND_RATE         ; SEND DATA RATE TO CONTROLLER
                             <1> FM_WR:
00001B47 E872060000          <1>  CALL  FMTDMA_SET        ; SET UP THE DMA FOR FORMAT
00001B4C 7245                <1>        JC      short FM_DON            ; RETURN WITH ERROR
00001B4E B44D                <1>  MOV   AH,04DH                 ; ESTABLISH THE FORMAT COMMAND
00001B50 E8CF060000          <1>  CALL  NEC_INIT          ; INITIALIZE THE NEC
00001B55 723C                <1>        JC      short FM_DON            ; ERROR - EXIT
00001B57 B8[931B0000]        <1>        MOV     eAX, FM_DON             ; LOAD ERROR ADDRESS
00001B5C 50                  <1>  PUSH  eAX               ; PUSH NEC_OUT ERROR RETURN
00001B5D B203                <1>  MOV   DL,3              ; BYTES/SECTOR VALUE TO NEC
00001B5F E83E090000          <1>  CALL  GET_PARM
00001B64 E83F0A0000          <1>  CALL  NEC_OUTPUT
00001B69 B204                <1>  MOV   DL,4              ; SECTORS/TRACK VALUE TO NEC
00001B6B E832090000          <1>  CALL  GET_PARM
00001B70 E8330A0000          <1>  CALL  NEC_OUTPUT
00001B75 B207                <1>  MOV   DL,7              ; GAP LENGTH VALUE TO NEC
00001B77 E826090000          <1>  CALL  GET_PARM
00001B7C E8270A0000          <1>  CALL  NEC_OUTPUT
00001B81 B208                <1>  MOV   DL,8              ; FILLER BYTE TO NEC
00001B83 E81A090000          <1>  CALL  GET_PARM
00001B88 E81B0A0000          <1>  CALL  NEC_OUTPUT
00001B8D 58                  <1>  POP   eAX               ; THROW AWAY ERROR
00001B8E E80F070000          <1>  CALL  NEC_TERM          ; TERMINATE, RECEIVE STATUS, ETC,
                             <1> FM_DON:
00001B93 E8F8020000          <1>  CALL  XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
00001B98 E831080000          <1>  CALL  SETUP_END         ; VARIOUS CLEANUPS
00001B9D 6689F3              <1>  MOV   BX,SI             ; GET SAVED AL TO BL
00001BA0 88D8                <1>  MOV   AL,BL             ; PUT BACK FOR RETURN
00001BA2 C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; FNC_ERR
                             <1> ;INVALID FUNCTION REQUESTED OR INVALID DRIVE:
                             <1> ;SET BAD COMMAND IN STATUS.
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> FNC_ERR:                       ; INVALID FUNCTION REQUEST
00001BA3 6689F0              <1>  MOV   AX,SI             ; RESTORE AL
00001BA6 B401                <1>  MOV   AH,BAD_CMD        ; SET BAD COMMAND ERROR
00001BA8 8825[2C710000]      <1>  MOV   [DSKETTE_STATUS],AH     ; STORE IN DATA AREA
00001BAE F9                  <1>  STC                     ; SET CARRY INDICATING ERROR
00001BAF C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DISK_PARMS (AH = 08H)
                             <1> ;READ DRIVE PARAMETERS.
                             <1> ;
                             <1> ; ON ENTRY:  DI : DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   CL/[BP]   = BITS 7 & 6 HI 2 BITS OF MAX CYLINDER
                             <1> ;                  BITS 0-5 MAX SECTORS/TRACK
                             <1> ;       CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER
                             <1> ;       BL/[BP+2] = BITS 7-4 = 0
                             <1> ;                  BITS 3-0 = VALID CMOS DRIVE TYPE
                             <1> ;       BH/[BP+3] = 0
                             <1> ;       DL/[BP+4] = # DRIVES INSTALLED (VALUE CHECKED)
                             <1> ;       DH/[BP+5] = MAX HEAD #
                             <1> ;       DI/[BP+6] = OFFSET TO DISK_BASE
                             <1> ;       ES        = SEGMENT OF DISK_BASE
                             <1> ;       AX        = 0
                             <1> ;
                             <1> ;       NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
                             <1> ;              THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM
                             <1> ;              INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
                             <1> ;              CALLER.
                             <1> ;-------------------------------------------------------------------------------
                             <1> DSK_PARMS:
00001BB0 E8AA020000          <1>  CALL  XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH,
                             <1>      ; MOV   WORD [BP+2],0          ; DRIVE TYPE = 0
00001BB5 29D2                <1>  sub     edx, edx ; 20/02/2015
00001BB7 895504              <1>        mov  [ebp+4], edx ; 20/02/2015
                             <1>      ;  MOV     AX, [EQUIP_FLAG]        ; LOAD EQUIPMENT FLAG FOR # DISKETTES
                             <1>      ;  AND     AL,11000001B            ; KEEP DISKETTE DRIVE BITS
```

```
                              <1>      ;  MOV     DL,2                  ; DISKETTE DRIVES = 2
                              <1>      ;  CMP     AL,01000001B          ; 2 DRIVES INSTALLED ?
                              <1>      ;  JZ      short STO_DL          ; IF YES JUMP
                              <1>      ;  DEC     DL                    ; DISKETTE DRIVES = 1
                              <1>      ;  CMP     AL,00000001B          ; 1 DRIVE INSTALLED ?
                              <1>      ;  JNZ     short NON_DRV         ; IF NO JUMP
                              <1>  ;sub  edx, edx
00001BBA 66A1[D86D0000]       <1>  mov     ax, [fd0_type]
00001BC0 6621C0               <1>  and     ax, ax
00001BC3 7474                 <1>  jz      short NON_DRV
00001BC5 FEC2                 <1>  inc     dl
00001BC7 20E4                 <1>  and     ah, ah
00001BC9 7402                 <1>  jz      short STO_DL
00001BCB FEC2                 <1>  inc     dl
                              <1> STO_DL:
                              <1>  ;MOV  [BP+4],DL        ; STORE NUMBER OF DRIVES
00001BCD 895508               <1>  mov   [ebp+8], edx ; 20/02/2015
00001BD0 6683FF01             <1>  CMP   DI,1             ; CHECK FOR VALID DRIVE
00001BD4 7766                 <1>  JA    short NON_DRV1         ; DRIVE INVALID
                              <1>  ;MOV  BYTE [BP+5],1          ; MAXIMUM HEAD NUMBER = 1
00001BD6 C6450901             <1>  mov   byte [ebp+9], 1  ; 20/02/2015
00001BDA E8BA080000           <1>  CALL  CMOS_TYPE        ; RETURN DRIVE TYPE IN AL
                              <1>  ;;20/02/2015
                              <1>  ;;JC  short CHK_EST          ; IF CMOS BAD CHECKSUM ESTABLISHED
                              <1>  ;;OR  AL,AL            ; TEST FOR NO DRIVE TYPE
00001BDF 7412                 <1>  JZ    short CHK_EST          ; JUMP IF SO
00001BE1 E805020000           <1>  CALL  DR_TYPE_CHECK          ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
00001BE6 720B                 <1>  JC    short CHK_EST          ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
                              <1>  ;MOV  [BP+2],AL        ; STORE VALID CMOS DRIVE TYPE
00001BE8 884504               <1>        mov  [ebp+4], al ; 06/02/2015
00001BEB 8A4B04               <1>  MOV     CL, [eBX+MD.SEC_TRK]     ; GET SECTOR/TRACK
00001BEE 8A6B0B               <1>        MOV     CH, [eBX+MD.MAX_TRK]     ; GET MAX. TRACK NUMBER
00001BF1 EB36                 <1>  JMP   SHORT STO_CX           ; CMOS GOOD, USE CMOS
                              <1> CHK_EST:
00001BF3 8AA7[39710000]       <1>  MOV   AH, [DSK_STATE+eDI]     ; LOAD STATE FOR THIS DRIVE
00001BF9 F6C410               <1>  TEST  AH,MED_DET        ; CHECK FOR ESTABLISHED STATE
00001BFC 743E                 <1>  JZ    short NON_DRV1         ; CMOS BAD/INVALID OR UNESTABLISHED
                              <1> USE_EST:
00001BFE 80E4C0               <1>  AND   AH,RATE_MSK       ; ISOLATE STATE
00001C01 80FC80               <1>  CMP   AH,RATE_250       ; RATE 250 ?
00001C04 7557                 <1>  JNE   short USE_EST2         ; NO, GO CHECK OTHER RATE
                              <1>
                              <1> ;----- DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST
                              <1>
00001C06 B001                 <1>  MOV   AL,01            ; DRIVE TYPE 1 (360KB)
00001C08 E8DE010000           <1>  CALL  DR_TYPE_CHECK          ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
00001C0D 8A4B04               <1>        MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
00001C10 8A6B0B               <1>        MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
00001C13 F687[39710000]01     <1>  TEST  byte [DSK_STATE+eDI],TRK_CAPA ; 80 TRACK ?
00001C1A 740D                 <1>  JZ    short STO_CX           ; MUST BE 360KB DRIVE
                              <1>
                              <1> ;----- IT IS 1.44 MB DRIVE
                              <1>
                              <1> PARM144:
00001C1C B004                 <1>  MOV   AL,04            ; DRIVE TYPE 4 (1.44MB)
00001C1E E8C8010000           <1>  CALL  DR_TYPE_CHECK          ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
00001C23 8A4B04               <1>        MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
00001C26 8A6B0B               <1>        MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
                              <1> STO_CX:
00001C29 894D00               <1>  MOV   [eBP],eCX        ; SAVE POINTER IN STACK FOR RETURN
                              <1> ES_DI:
                              <1>  ;MOV  [BP+6],BX        ; ADDRESS OF MEDIA/DRIVE PARM TABLE
00001C2C 895D0C               <1>  mov   [ebp+12], ebx ; 06/02/2015
                              <1>  ;MOV  AX,CS            ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
                              <1>  ;MOV  ES,AX            ; ES IS SEGMENT OF TABLE
                              <1> DP_OUT:
00001C2F E85C020000           <1>  CALL  XLAT_OLD         ; TRANSLATE STATE TO COMPATIBLE MODE
00001C34 6631C0               <1>  XOR   AX,AX            ; CLEAR
00001C37 F8                   <1>  CLC
00001C38 C3                   <1>  RETn
                              <1>
                              <1> ;----- NO DRIYE PRESENT HANDLER
                              <1>
                              <1> NON_DRV:
                              <1>  ;MOV  BYTE [BP+4],0          ; CLEAR NUMBER OF DRIVES
00001C39 895508               <1>  mov   [ebp+8], edx ; 0 ; 20/02/2015
                              <1> NON_DRV1:
00001C3C 6681FF8000           <1>  CMP   DI,80H                 ; CHECK FOR FIXED MEDIA TYPE REQUEST
00001C41 720C                 <1>  JB    short NON_DRV2         ; CONTINUE IF NOT REQUEST FALL THROUGH
                              <1>
                              <1> ;----- FIXED DISK REQUEST FALL THROUGH ERROR
                              <1>
00001C43 E848020000           <1>  CALL  XLAT_OLD         ; ELSE TRANSLATE TO COMPATIBLE MODE
00001C48 6689F0               <1>  MOV   AX,SI            ; RESTORE AL
00001C4B B401                 <1>  MOV   AH,BAD_CMD       ; SET BAD COMMAND ERROR
00001C4D F9                   <1>  STC
00001C4E C3                   <1>  RETn
                              <1>
                              <1> NON_DRV2:
                              <1>  ;XOR  AX,AX            ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
00001C4F 31C0                 <1>  xor   eax, eax
00001C51 66894500             <1>  MOV   [eBP],AX         ; TRACKS, SECTORS/TRACK = 0
```

```
                             <1>  ;MOV  [BP+5],AH        ; HEAD = 0
00001C55 886509              <1>  mov   [ebp+9], ah ; 06/02/2015
                             <1>  ;MOV  [BP+6],AX        ; OFFSET TO DISK_BASE = 0
00001C58 89450C              <1>  mov   [ebp+12], eax
                             <1>  ;MOV  ES,AX            ; ES IS SEGMENT OF TABLE
00001C5B EBD2                <1>  JMP   SHORT DP_OUT
                             <1>
                             <1> ;----- DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST
                             <1>
                             <1> USE_EST2:
00001C5D B002                <1>  MOV   AL,02            ; DRIVE TYPE 2 (1.2MB)
00001C5F E887010000          <1>  CALL  DR_TYPE_CHECK          ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
00001C64 8A4B04              <1>         MOV     CL, [eBX+MD.SEC_TRK]    ; GET SECTOR/TRACK
00001C67 8A6B0B              <1>         MOV     CH, [eBX+MD.MAX_TRK]    ; GET MAX. TRACK NUMBER
00001C6A 80FC40              <1>  CMP   AH,RATE_300       ; RATE 300 ?
00001C6D 74BA                <1>  JZ    short STO_CX           ; MUST BE 1.2MB DRIVE
00001C6F EBAB                <1>  JMP   SHORT PARM144          ; ELSE, IT IS 1.44MB DRIVE
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DISK_TYPE (AH = 15H)
                             <1> ;THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
                             <1> ;
                             <1> ;  ON ENTRY: DI = DRIVE #
                             <1> ;
                             <1> ;  ON EXIT:  AH = DRIVE TYPE, CY=0
                             <1> ;-------------------------------------------------------------------------------
                             <1> DSK_TYPE:
00001C71 E8E9010000          <1>  CALL  XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
00001C76 8A87[39710000]      <1>  MOV   AL, [DSK_STATE+eDI]     ; GET PRESENT STATE INFORMATION
00001C7C 08C0                <1>  OR    AL,AL             ; CHECK FOR NO DRIVE
00001C7E 7418                <1>  JZ    short NO_DRV
00001C80 B401                <1>  MOV   AH,NOCHGLN        ; NO CHANGE LINE FOR 40 TRACK DRIVE
00001C82 A801                <1>  TEST  AL,TRK_CAPA       ; IS THIS DRIVE AN 80 TRACK DRIVE?
00001C84 7402                <1>  JZ    short DT_BACK                ; IF NO JUMP
00001C86 B402                <1>  MOV   AH,CHGLN          ; CHANGE LINE FOR 80 TRACK DRIVE
                             <1> DT_BACK:
00001C88 6650                <1>  PUSH  AX                ; SAVE RETURN VALUE
00001C8A E801020000          <1>  CALL  XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
00001C8F 6658                <1>  POP   AX                ; RESTORE RETURN VALUE
00001C91 F8                  <1>  CLC                     ; NO ERROR
00001C92 6689F3              <1>  MOV   BX,SI             ; GET SAVED AL TO BL
00001C95 88D8                <1>  MOV   AL,BL             ; PUT BACK FOR RETURN
00001C97 C3                  <1>  RETn
                             <1> NO_DRV:
00001C98 30E4                <1>  XOR   AH,AH             ; NO DRIVE PRESENT OR UNKNOWN
00001C9A EBEC                <1>  JMP   SHORT DT_BACK
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DISK_CHANGE(AH = 16H)
                             <1> ;THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   AH = @DSKETTE_STATUS
                             <1> ;           00 - DISK CHANGE LINE INACTIVE, CY = 0
                             <1> ;           06 - DISK CHANGE LINE ACTIVE, CY = 1
                             <1> ;-------------------------------------------------------------------------------
                             <1> DSK_CHANGE:
00001C9C E8BE010000          <1>  CALL  XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
00001CA1 8A87[39710000]      <1>  MOV   AL, [DSK_STATE+eDI]     ; GET MEDIA STATE INFORMATION
00001CA7 08C0                <1>  OR    AL,AL             ; DRIVE PRESENT ?
00001CA9 7422                <1>  JZ    short DC_NON           ; JUMP IF NO DRIVE
00001CAB A801                <1>  TEST  AL,TRK_CAPA       ; 80 TRACK DRIVE ?
00001CAD 7407                <1>  JZ    short SETIT       ; IF SO , CHECK CHANGE LINE
                             <1> DC0:
00001CAF E88C0A0000          <1>         CALL    READ_DSKCHNG            ; GO CHECK STATE OF DISK CHANGE LINE
00001CB4 7407                <1>  JZ    short FINIS       ; CHANGE LINE NOT ACTIVE
                             <1>
00001CB6 C605[2C710000]06    <1> SETIT: MOV   byte [DSKETTE_STATUS], MEDIA_CHANGE ; INDICATE MEDIA REMOVED
                             <1>
00001CBD E8CE010000          <1> FINIS: CALL  XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
00001CC2 E807070000          <1>  CALL  SETUP_END         ; VARIOUS CLEANUPS
00001CC7 6689F3              <1>  MOV   BX,SI             ; GET SAVED AL TO BL
00001CCA 88D8                <1>  MOV   AL,BL             ; PUT BACK FOR RETURN
00001CCC C3                  <1>  RETn
                             <1> DC_NON:
00001CCD 800D[2C710000]80    <1>  OR    byte [DSKETTE_STATUS], TIME_OUT ; SET TIMEOUT, NO DRIVE
00001CD4 EBE7                <1>  JMP   SHORT FINIS
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; FORMAT_SET (AH = 17H)
                             <1> ;THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF MEDIA TO BE USED
                             <1> ;FOR THE FOLLOWING FORMAT OPERATION.
                             <1> ;
                             <1> ; ON ENTRY:  SI LOW = DASD TYPE FOR FORMAT
                             <1> ;       DI     = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS REFLECTS STATUS
                             <1> ;       AH = @DSKETTE_STATUS
                             <1> ;       CY = 1 IF ERROR
                             <1> ;-------------------------------------------------------------------------------
                             <1> FORMAT_SET:
```

```
00001CD6 E884010000            <1>  CALL  XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
00001CDB 6656                  <1>  PUSH  SI                ; SAVE DASD TYPE
00001CDD 6689F0                <1>  MOV   AX,SI             ; AH = ? , AL , DASD TYPE
00001CE0 30E4                  <1>  XOR   AH,AH             ; AH , 0 , AL , DASD TYPE
00001CE2 6689C6                <1>  MOV   SI,AX             ; SI = DASD TYPE
00001CE5 80A7[39710000]0F      <1>  AND   byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
00001CEC 664E                  <1>  DEC   SI                ; CHECK FOR 320/360K MEDIA & DRIVE
00001CEE 7509                  <1>  JNZ   short NOT_320           ; BYPASS IF NOT
00001CF0 808F[39710000]90      <1>  OR    byte [DSK_STATE+eDI], MED_DET+RATE_250 ; SET TO 320/360
00001CF7 EB48                  <1>  JMP   SHORT S0
                               <1>
                               <1> NOT_320:
00001CF9 E8B6030000            <1>  CALL  MED_CHANGE        ; CHECK FOR TIME_OUT
00001CFE 803D[2C710000]80      <1>  CMP   byte [DSKETTE_STATUS], TIME_OUT
00001D05 743A                  <1>  JZ    short S0          ; IF TIME OUT TELL CALLER
                               <1> S3:
00001D07 664E                  <1>  DEC   SI                ; CHECK FOR 320/360K IN 1.2M DRIVE
00001D09 7509                  <1>  JNZ   short NOT_320_12  ; BYPASS IF NOT
00001D0B 808F[39710000]70      <1>  OR    byte [DSK_STATE+eDI], MED_DET+DBL_STEP+RATE_300 ; SET STATE
00001D12 EB2D                  <1>  JMP   SHORT S0
                               <1>
                               <1> NOT_320_12:
00001D14 664E                  <1>  DEC   SI                ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
00001D16 7509                  <1>  JNZ   short NOT_12            ; BYPASS IF NOT
00001D18 808F[39710000]10      <1>  OR    byte [DSK_STATE+eDI], MED_DET+RATE_500 ; SET STATE VARIABLE
00001D1F EB20                  <1>  JMP   SHORT S0          ; RETURN TO CALLER
                               <1>
                               <1> NOT_12:
00001D21 664E                  <1>  DEC   SI                ; CHECK FOR SET DASD TYPE 04
00001D23 752B                  <1>  JNZ   short FS_ERR            ; BAD COMMAND EXIT IF NOT VALID TYPE
                               <1>
00001D25 F687[39710000]04      <1>  TEST  byte [DSK_STATE+eDI], DRV_DET ; DRIVE DETERMINED ?
00001D2C 740B                  <1>  JZ    short ASSUME            ; IF STILL NOT DETERMINED ASSUME
00001D2E B050                  <1>  MOV   AL,MED_DET+RATE_300
00001D30 F687[39710000]02      <1>         TEST    byte [DSK_STATE+eDI], FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
00001D37 7502                  <1>  JNZ   short OR_IT_IN          ; IF 1.2 M THEN DATA RATE 300
                               <1>
                               <1> ASSUME:
00001D39 B090                  <1>  MOV   AL,MED_DET+RATE_250     ; SET UP
                               <1>
                               <1> OR_IT_IN:
00001D3B 0887[39710000]        <1>  OR    [DSK_STATE+eDI], AL     ; OR IN THE CORRECT STATE
                               <1> S0:
00001D41 E84A010000            <1>  CALL  XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
00001D46 E883060000            <1>  CALL  SETUP_END         ; VARIOUS CLEANUPS
00001D4B 665B                  <1>  POP   BX                ; GET SAVED AL TO BL
00001D4D 88D8                  <1>  MOV   AL,BL             ; PUT BACK FOR RETURN
00001D4F C3                    <1>  RETn
                               <1>
                               <1> FS_ERR:
00001D50 C605[2C710000]01      <1>  MOV   byte [DSKETTE_STATUS], BAD_CMD ; UNKNOWN STATE,BAD COMMAND
00001D57 EBE8                  <1>  JMP   SHORT S0
                               <1>
                               <1> ;-------------------------------------------------------------------------------
                               <1> ; SET_MEDIA   (AH = 18H)
                               <1> ;THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE
                               <1> ;TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
                               <1> ;
                               <1> ; ON ENTRY:
                               <1> ;[BP]  = SECTOR PER TRACK
                               <1> ;[BP+1]      = TRACK #
                               <1> ;DI    = DRIVE #
                               <1> ;
                               <1> ; ON EXIT:
                               <1> ;@DSKETTE_STATUS REFLECTS STATUS
                               <1> ;IF NO ERROR:
                               <1> ;      AH = 0
                               <1> ;      CY = 0
                               <1> ;      ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
                               <1> ;      DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE
                               <1> ;IF ERROR:
                               <1> ;      AH = @DSKETTE_STATUS
                               <1> ;      CY = 1
                               <1> ;-------------------------------------------------------------------------------
                               <1> SET_MEDIA:
00001D59 E801010000            <1>  CALL  XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
00001D5E F687[39710000]01      <1>         TEST    byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR CHANGE LINE AVAILABLE
00001D65 7415                  <1>  JZ    short SM_CMOS           ; JUMP IF 40 TRACK DRIVE
00001D67 E848030000            <1>  CALL  MED_CHANGE        ; RESET CHANGE LINE
00001D6C 803D[2C710000]80      <1>  CMP   byte [DSKETTE_STATUS], TIME_OUT ; IF TIME OUT TELL CALLER
00001D73 746B                  <1>  JE    short SM_RTN
00001D75 C605[2C710000]00      <1>  MOV   byte [DSKETTE_STATUS], 0 ; CLEAR STATUS
                               <1> SM_CMOS:
00001D7C E818070000            <1>  CALL  CMOS_TYPE         ; RETURN DRIVE TYPE IN (AL)
                               <1>  ;;20/02/2015
                               <1>  ;;JC  short MD_NOT_FND  ; ERROR IN CMOS
                               <1>  ;;OR  AL,AL             ; TEST FOR NO DRIVE
00001D81 745D                  <1>  JZ    short SM_RTN            ; RETURN IF SO
00001D83 E863000000            <1>  CALL  DR_TYPE_CHECK           ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
00001D88 7231                  <1>  JC    short MD_NOT_FND  ; TYPE NOT IN TABLE (BAD CMOS)
00001D8A 57                    <1>  PUSH  eDI               ; SAVE REG.
00001D8B 31DB                  <1>  XOR   eBX,eBX                 ; BX = INDEX TO DR. TYPE TABLE
```

```
5963 00001D8D B906000000          <1>  MOV   eCX,DR_CNT       ; CX = LOOP COUNT
5964                              <1> DR_SEARCH:
5965 00001D92 8AA3[DC6A0000]      <1>  MOV   AH, [DR_TYPE+eBX] ; GET DRIVE TYPE
5966 00001D98 80E47F              <1>  AND   AH,BIT7OFF       ; MASK OUT MSB
5967 00001D9B 38E0                <1>  CMP   AL,AH            ; DRIVE TYPE MATCH ?
5968 00001D9D 7516                <1>  JNE   short NXT_MD           ; NO, CHECK NEXT DRIVE TYPE
5969                              <1> DR_FND:
5970 00001D9F 8BBB[DD6A0000]      <1>  MOV   eDI, [DR_TYPE+eBX+1]    ; DI = MEDIA/DRIVE PARAM TABLE
5971                              <1> MD_SEARCH:
5972 00001DA5 8A6704              <1>        MOV     AH, [eDI+MD.SEC_TRK]    ; GET SECTOR/TRACK
5973 00001DA8 386500              <1>  CMP   [eBP],AH         ; MATCH?
5974 00001DAB 7508                <1>  JNE   short NXT_MD           ; NO, CHECK NEXT MEDIA
5975 00001DAD 8A670B              <1>        MOV     AH, [eDI+MD.MAX_TRK]    ; GET MAX. TRACK #
5976 00001DB0 386501              <1>  CMP   [eBP+1],AH       ; MATCH?
5977 00001DB3 740F                <1>  JE    short MD_FND           ; YES, GO GET RATE
5978                              <1> NXT_MD:
5979                              <1>  ;ADD  BX,3             ; CHECK NEXT DRIVE TYPE
5980 00001DB5 83C305              <1>        add  ebx, 5 ; 18/02/2015
5981 00001DB8 E2D8                <1>  LOOP    DR_SEARCH
5982 00001DBA 5F                  <1>  POP   eDI              ; RESTORE REG.
5983                              <1> MD_NOT_FND:
5984 00001DBB C605[2C710000]0C    <1>  MOV   byte [DSKETTE_STATUS], MED_NOT_FND ; ERROR, MEDIA TYPE NOT FOUND
5985 00001DC2 EB1C                <1>  JMP   SHORT SM_RTN           ; RETURN
5986                              <1> MD_FND:
5987 00001DC4 8A470C              <1>        MOV     AL, [eDI+MD.RATE]       ; GET RATE
5988 00001DC7 3C40                <1>  CMP   AL,RATE_300      ; DOUBLE STEP REQUIRED FOR RATE 300
5989 00001DC9 7502                <1>  JNE   short MD_SET
5990 00001DCB 0C20                <1>  OR    AL,DBL_STEP
5991                              <1> MD_SET:
5992                              <1>  ;MOV  [BP+6],DI        ; SAVE TABLE POINTER IN STACK
5993 00001DCD 897D0C              <1>  mov   [ebp+12], edi ; 18/02/2015
5994 00001DD0 0C10                <1>  OR    AL,MED_DET       ; SET MEDIA ESTABLISHED
5995 00001DD2 5F                  <1>  POP   eDI
5996 00001DD3 80A7[39710000]0F    <1>  AND   byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
5997 00001DDA 0887[39710000]      <1>  OR    [DSK_STATE+eDI], AL
5998                              <1>  ;MOV  AX, CS                  ; SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
5999                              <1>  ;MOV  ES, AX                  ; ES IS SEGMENT OF TABLE
6000                              <1> SM_RTN:
6001 00001DE0 E8AB000000          <1>  CALL  XLAT_OLD         ; TRANSLATE STATE TO COMPATIBLE MODE
6002 00001DE5 E8E4050000          <1>  CALL  SETUP_END        ; VARIOUS CLEANUPS
6003 00001DEA C3                  <1>  RETn
6004                              <1>
6005                              <1> ;-------------------------------------------------------------
6006                              <1> ; DR_TYPE_CHECK                                               :
6007                              <1> ;CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL)              :
6008                              <1> ;IS SUPPORTED IN BIOS DRIVE TYPE TABLE                       :
6009                              <1> ; ON ENTRY:                                          :
6010                              <1> ;AL = DRIVE TYPE                                     :
6011                              <1> ; ON EXIT:                                           :
6012                              <1> ;CS = SEGMENT MEDIA/DRIVE PARAMETER TABLE (CODE)             :
6013                              <1> ;CY = 0      DRIVE TYPE SUPPORTED                            :
6014                              <1> ;      BX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE            :
6015                              <1> ;CY = 1      DRIVE TYPE NOT SUPPORTED                        :
6016                              <1> ; REGISTERS ALTERED: eBX                                     :
6017                              <1> ;-------------------------------------------------------------
6018                              <1> DR_TYPE_CHECK:
6019 00001DEB 6650                <1>  PUSH  AX
6020 00001DED 51                  <1>  PUSH  eCX
6021 00001DEE 31DB                <1>  XOR   eBX,eBX                 ; BX = INDEX TO DR_TYPE TABLE
6022 00001DF0 B906000000          <1>  MOV   eCX,DR_CNT       ; CX = LOOP COUNT
6023                              <1> TYPE_CHK:
6024 00001DF5 8AA3[DC6A0000]      <1>  MOV   AH,[DR_TYPE+eBX]  ; GET DRIVE TYPE
6025 00001DFB 38E0                <1>  CMP   AL,AH            ; DRIVE TYPE MATCH?
6026 00001DFD 740D                <1>  JE    short DR_TYPE_VALID     ; YES, RETURN WITH CARRY RESET
6027                              <1>  ;ADD  BX,3             ; CHECK NEXT DRIVE TYPE
6028 00001DFF 83C305              <1>        add  ebx, 5      ; 16/02/2015 (32 bit address modification)
6029 00001E02 E2F1                <1>  LOOP    TYPE_CHK
6030                              <1>  ;
6031 00001E04 BB[3B6B0000]        <1>  mov   ebx, MD_TBL6           ; 1.44MB fd parameter table
6032                              <1>                         ; Default for GET_PARM (11/12/2014)
6033                              <1>  ;
6034 00001E09 F9                  <1>  STC                    ; DRIVE TYPE NOT FOUND IN TABLE
6035 00001E0A EB06                <1>  JMP   SHORT TYPE_RTN
6036                              <1> DR_TYPE_VALID:
6037 00001E0C 8B9B[DD6A0000]      <1>  MOV   eBX,[DR_TYPE+eBX+1]     ; BX = MEDIA TABLE
6038                              <1> TYPE_RTN:
6039 00001E12 59                  <1>  POP   eCX
6040 00001E13 6658                <1>  POP   AX
6041 00001E15 C3                  <1>  RETn
6042                              <1>
6043                              <1> ;-------------------------------------------------------------
6044                              <1> ; SEND_SPEC                                          :
6045                              <1> ;SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM     :
6046                              <1> ;THE DRIVE PARAMETER TABLE POINTED BY @DISK_POINTER   :
6047                              <1> ; ON ENTRY:  @DISK_POINTER = DRIVE PARAMETER TABLE          :
6048                              <1> ; ON EXIT:   NONE                                    :
6049                              <1> ; REGISTERS ALTERED: CX, DX                                  :
6050                              <1> ;-------------------------------------------------------------
6051                              <1> SEND_SPEC:
6052 00001E16 50                  <1>  PUSH  eAX              ; SAVE AX
6053 00001E17 B8[3D1E0000]        <1>  MOV   eAX, SPECBAC           ; LOAD ERROR ADDRESS
6054 00001E1C 50                  <1>  PUSH  eAX              ; PUSH NEC_OUT ERROR RETURN
```

```
00001E1D B403                <1>  MOV   AH,03H                  ; SPECIFY COMMAND
00001E1F E884070000          <1>  CALL  NEC_OUTPUT        ; OUTPUT THE COMMAND
00001E24 28D2                <1>  SUB   DL,DL             ; FIRST SPECIFY BYTE
00001E26 E877060000          <1>  CALL  GET_PARM          ; GET PARAMETER TO AH
00001E2B E878070000          <1>  CALL  NEC_OUTPUT        ; OUTPUT THE COMMAND
00001E30 B201                <1>  MOV   DL,1              ; SECOND SPECIFY BYTE
00001E32 E86B060000          <1>  CALL  GET_PARM          ; GET PARAMETER TO AH
00001E37 E86C070000          <1>  CALL  NEC_OUTPUT        ; OUTPUT THE COMMAND
00001E3C 58                  <1>  POP   eAX               ; POP ERROR RETURN
                             <1> SPECBAC:
00001E3D 58                  <1>  POP   eAX               ; RESTORE ORIGINAL AX VALUE
00001E3E C3                  <1>  RETn
                             <1>
                             <1> ;-----------------------------------------------------------------
                             <1> ; SEND_SPEC_MD                                         :
                             <1> ;SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM     :
                             <1> ;THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX)   :
                             <1> ; ON ENTRY:  CS:BX = MEDIA/DRIVE PARAMETER TABLE      :
                             <1> ; ON EXIT:   NONE                                :
                             <1> ; REGISTERS ALTERED: AX                                :
                             <1> ;-----------------------------------------------------------------
                             <1> SEND_SPEC_MD:
00001E3F 50                  <1>  PUSH  eAX               ; SAVE RATE DATA
00001E40 B8[5D1E0000]        <1>  MOV   eAX, SPEC_ESBAC         ; LOAD ERROR ADDRESS
00001E45 50                  <1>  PUSH  eAX               ; PUSH NEC_OUT ERROR RETURN
00001E46 B403                <1>  MOV   AH,03H                  ; SPECIFY COMMAND
00001E48 E85B070000          <1>  CALL  NEC_OUTPUT        ; OUTPUT THE COMMAND
00001E4D 8A23                <1>         MOV     AH, [eBX+MD.SPEC1]      ; GET 1ST SPECIFY BYTE
00001E4F E854070000          <1>  CALL  NEC_OUTPUT        ; OUTPUT THE COMMAND
00001E54 8A6301              <1>         MOV     AH, [eBX+MD.SPEC2]      ; GET SECOND SPECIFY BYTE
00001E57 E84C070000          <1>  CALL  NEC_OUTPUT        ; OUTPUT THE COMMAND
00001E5C 58                  <1>  POP   eAX               ; POP ERROR RETURN
                             <1> SPEC_ESBAC:
00001E5D 58                  <1>  POP   eAX               ; RESTORE ORIGINAL AX VALUE
00001E5E C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; XLAT_NEW
                             <1> ;TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE
                             <1> ;MODE TO NEW ARCHITECTURE.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;-------------------------------------------------------------------------------
                             <1> XLAT_NEW:
00001E5F 83FF01              <1>  CMP   eDI,1                   ; VALID DRIVE
00001E62 7725                <1>  JA    short XN_OUT                  ; IF INVALID BACK
00001E64 80BF[39710000]00    <1>  CMP   byte [DSK_STATE+eDI], 0       ; NO DRIVE ?
00001E6B 741D                <1>  JZ    short DO_DET                  ; IF NO DRIVE ATTEMPT DETERMINE
00001E6D 6689F9              <1>  MOV   CX,DI                   ; CX = DRIVE NUMBER
00001E70 C0E102              <1>  SHL   CL,2                    ; CL = SHIFT COUNT, A=0, B=4
00001E73 A0[38710000]        <1>  MOV   AL, [HF_CNTRL]                ; DRIVE INFORMATION
00001E78 D2C8                <1>  ROR   AL,CL                   ; TO LOW NIBBLE
00001E7A 2407                <1>  AND   AL,DRV_DET+FMT_CAPA+TRK_CAPA  ; KEEP DRIVE BITS
00001E7C 80A7[39710000]F8    <1>         AND     byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA)
00001E83 0887[39710000]      <1>  OR    [DSK_STATE+eDI], AL           ; UPDATE DRIVE STATE
                             <1> XN_OUT:
00001E89 C3                  <1>  RETn
                             <1> DO_DET:
00001E8A E8BE080000          <1>  CALL  DRIVE_DET               ; TRY TO DETERMINE
00001E8F C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; XLAT_OLD
                             <1> ;TRANSLATES DISKETTE STATE LOCATIONS FROM NEW
                             <1> ;ARCHITECTURE TO COMPATIBLE MODE.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE
                             <1> ;-------------------------------------------------------------------------------
                             <1> XLAT_OLD:
00001E90 83FF01              <1>  CMP   eDI,1             ; VALID DRIVE ?
                             <1>         ;JA     short XO_OUT            ; IF INVALID BACK
00001E93 0F8786000000        <1>         ja      XO_OUT
00001E99 80BF[39710000]00    <1>         CMP  byte [DSK_STATE+eDI],0  ; NO DRIVE ?
00001EA0 747D                <1>  JZ    short XO_OUT            ; IF NO DRIVE TRANSLATE DONE
                             <1>
                             <1> ;----- TEST FOR SAVED DRIVE INFORMATION ALREADY SET
                             <1>
00001EA2 6689F9              <1>  MOV   CX,DI             ; CX = DRIVE NUMBER
00001EA5 C0E102              <1>  SHL   CL,2              ; CL = SHIFT COUNT, A=0, B=4
00001EA8 B402                <1>  MOV   AH,FMT_CAPA       ; LOAD MULTIPLE DATA RATE BIT MASK
00001EAA D2CC                <1>  ROR   AH,CL             ; ROTATE BY MASK
00001EAC 8425[38710000]      <1>  TEST  [HF_CNTRL], AH          ; MULTIPLE-DATA RATE DETERMINED ?
00001EB2 751C                <1>  JNZ   short SAVE_SET          ; IF SO, NO NEED TO RE-SAVE
                             <1>
                             <1> ;----- ERASE DRIVE BITS IN @HF_CNTRL FOR THIS DRIVE
                             <1>
00001EB4 B407                <1>  MOV   AH,DRV_DET+FMT_CAPA+TRK_CAPA ; MASK TO KEEP
00001EB6 D2CC                <1>  ROR   AH,CL             ; FIX MASK TO KEEP
00001EB8 F6D4                <1>  NOT   AH                ; TRANSLATE MASK
00001EBA 2025[38710000]      <1>  AND   [HF_CNTRL], AH          ; KEEP BITS FROM OTHER DRIVE INTACT
                             <1>
                             <1> ;----- ACCESS CURRENT DRIVE BITS AND STORE IN @HF_CNTRL
                             <1>
00001EC0 8A87[39710000]      <1>  MOV   AL, [DSK_STATE+eDI]     ; ACCESS STATE
00001EC6 2407                <1>  AND   AL,DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
00001EC8 D2C8                <1>  ROR   AL,CL             ; FIX FOR THIS DRIVE
00001ECA 0805[38710000]      <1>  OR    [HF_CNTRL], AL         ; UPDATE SAVED DRIVE STATE
                             <1>
                             <1> ;----- TRANSLATE TO COMPATIBILITY MODE
                             <1>
                             <1> SAVE_SET:
00001ED0 8AA7[39710000]      <1>  MOV   AH, [DSK_STATE+eDI]     ; ACCESS STATE
00001ED6 88E7                <1>  MOV   BH,AH             ; TO BH FOR LATER
00001ED8 80E4C0              <1>  AND   AH,RATE_MSK       ; KEEP ONLY RATE
00001EDB 80FC00              <1>  CMP   AH,RATE_500       ; RATE 500 ?
00001EDE 7410                <1>  JZ    short CHK_144          ; YES 1.2/1.2 OR 1.44/1.44
00001EE0 B001                <1>  MOV   AL,M3D1U          ; AL = 360 IN 1.2 UNESTABLISHED
00001EE2 80FC40              <1>  CMP   AH,RATE_300       ; RATE 300 ?
00001EE5 7518                <1>  JNZ   short CHK_250          ; NO, 360/360, 720/720 OR 720/1.44
00001EE7 F6C720              <1>  TEST  BH,DBL_STEP       ; CHECK FOR DOUBLE STEP
00001EEA 751F                <1>  JNZ   short TST_DET          ; MUST BE 360 IN 1.2
                             <1> UNKNO:
00001EEC B007                <1>  MOV   AL,MED_UNK        ; NONE OF THE ABOVE
00001EEE EB22                <1>  JMP   SHORT AL_SET           ; PROCESS COMPLETE
                             <1> CHK_144:
00001EF0 E8A4050000          <1>  CALL  CMOS_TYPE         ; RETURN DRIVE TYPE IN (AL)
                             <1>  ;;20/02/2015
                             <1>  ;;JC  short UNKNO       ; ERROR, SET 'NONE OF ABOVE'
00001EF5 74F5                <1>  jz    short UNKNO ;; 20/02/2015
00001EF7 3C02                <1>  CMP   AL,2              ; 1.2MB DRIVE ?
00001EF9 75F1                <1>  JNE   short UNKNO       ; NO, GO SET 'NONE OF ABOVE'
00001EFB B002                <1>  MOV   AL,M1D1U          ; AL = 1.2 IN 1.2 UNESTABLISHED
00001EFD EB0C                <1>  JMP   SHORT TST_DET
                             <1> CHK_250:
00001EFF B000                <1>  MOV   AL,M3D3U          ; AL = 360 IN 360 UNESTABLISHED
00001F01 80FC80              <1>  CMP   AH,RATE_250       ; RATE 250 ?
00001F04 75E6                <1>  JNZ   short UNKNO       ; IF SO FALL IHRU
00001F06 F6C701              <1>  TEST  BH,TRK_CAPA       ; 80 TRACK CAPABILITY ?
00001F09 75E1                <1>  JNZ   short UNKNO       ; IF SO JUMP, FALL THRU TEST DET
                             <1> TST_DET:
00001F0B F6C710              <1>  TEST  BH,MED_DET        ; DETERMINED ?
00001F0E 7402                <1>  JZ    short AL_SET           ; IF NOT THEN SET
00001F10 0403                <1>  ADD   AL,3              ; MAKE DETERMINED/ESTABLISHED
                             <1> AL_SET:
00001F12 80A7[39710000]F8    <1>  AND   byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA) ; CLEAR DRIVE
00001F19 0887[39710000]      <1>  OR    [DSK_STATE+eDI], AL     ; REPLACE WITH COMPATIBLE MODE
                             <1> XO_OUT:
00001F1F C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; RD_WR_VF
                             <1> ;COMMON READ, WRITE AND VERIFY:
                             <1> ;MAIN LOOP FOR STATE RETRIES.
                             <1> ;
                             <1> ; ON ENTRY:  AH = READ/WRITE/VERIFY NEC PARAMETER
                             <1> ;       AL = READ/WRITE/VERIFY DMA PARAMETER
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> RD_WR_VF:
00001F20 6650                <1>  PUSH  AX                ; SAVE DMA, NEC PARAMETERS
00001F22 E838FFFFFF          <1>  CALL  XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
00001F27 E8F3000000          <1>  CALL  SETUP_STATE       ; INITIALIZE START AND END RATE
00001F2C 6658                <1>  POP   AX                ; RESTORE READ/WRITE/VERIFY
                             <1> DO_AGAIN:
00001F2E 6650                <1>  PUSH  AX                ; SAVE READ/WRITE/VERIFY PARAMETER
00001F30 E87F010000          <1>  CALL  MED_CHANGE        ; MEDIA CHANGE AND RESET IF CHANGED
00001F35 6658                <1>  POP   AX                ; RESTORE READ/WRITE/VERIFY
00001F37 0F82C9000000        <1>        JC      RWV_END                 ; MEDIA CHANGE ERROR OR TIME-OUT
                             <1> RWV:
00001F3D 6650                <1>  PUSH  AX                ; SAVE READ/WRITE/VERIFY PARAMETER
00001F3F 8AB7[39710000]      <1>  MOV   DH, [DSK_STATE+eDI]     ; GET RATE STATE OF THIS DRIVE
00001F45 80E6C0              <1>  AND   DH,RATE_MSK       ; KEEP ONLY RATE
00001F48 E84C050000          <1>  CALL  CMOS_TYPE         ; RETURN DRIVE TYPE IN AL (AL)
                             <1>  ;;20/02/2015
                             <1>  ;;JC  short RWV_ASSUME  ; ERROR IN CMOS
00001F4D 7451                <1>  jz    short RWV_ASSUME ; 20/02/2015
00001F4F 3C01                <1>  CMP   AL,1              ; 40 TRACK DRIVE?
00001F51 750D                <1>  JNE   short RWV_1       ; NO, BYPASS CMOS VALIDITY CHECK
00001F53 F687[39710000]01    <1>  TEST  byte [DSK_STATE+eDI], TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
00001F5A 7413                <1>  JZ    short RWV_2       ; YES, CMOS IS CORRECT
00001F5C B002                <1>  MOV   AL,2              ; CHANGE TO 1.2M
00001F5E EB0F                <1>  JMP   SHORT RWV_2
                             <1> RWV_1:
00001F60 720D                <1>  JB    short RWV_2       ; NO DRIVE SPECIFIED, CONTINUE
00001F62 F687[39710000]01    <1>  TEST    byte [DSK_STATE+eDI], TRK_CAPA ; IS IT REALLY 40 TRACK?
00001F69 7504                <1>  JNZ   short RWV_2       ; NO, 80 TRACK
00001F6B B001                <1>  MOV   AL,1              ; IT IS 40 TRACK, FIX CMOS VALUE
00001F6D EB04                <1>  jmp   short rwv_3
                             <1> RWV_2:
00001F6F 08C0                <1>  OR    AL,AL             ; TEST FOR NO DRIVE
00001F71 742D                <1>  JZ    short RWV_ASSUME  ; ASSUME TYPE, USE MAX TRACK
                             <1> rwv_3:
00001F73 E873FEFFFF          <1>  CALL  DR_TYPE_CHECK          ; RTN CS:BX = MEDIA/DRIVE PARAM TBL.
```

```
00001F78 7226                <1>  JC    short RWV_ASSUME  ; TYPE NOT IN TABLE (BAD CMOS)
                             <1>
                             <1> ;----- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
                             <1>
00001F7A 57                  <1>  PUSH  eDI               ; SAVE DRIVE #
00001F7B 31DB                <1>  XOR   eBX,eBX                 ; BX = INDEX TO DR_TYPE TABLE
00001F7D B906000000          <1>  MOV   eCX,DR_CNT        ; CX = LOOP COUNT
                             <1> RWV_DR_SEARCH:
00001F82 8AA3[DC6A0000]      <1>  MOV   AH, [DR_TYPE+eBX] ; GET DRIVE TYPE
00001F88 80E47F              <1>  AND   AH,BIT7OFF        ; MASK OUT MSB
00001F8B 38E0                <1>  CMP   AL,AH             ; DRIVE TYPE MATCH?
00001F8D 750B                <1>  JNE   short RWV_NXT_MD  ; NO, CHECK NEXT DRIVE TYPE
                             <1> RWV_DR_FND:
00001F8F 8BBB[DD6A0000]      <1>  MOV   eDI, [DR_TYPE+eBX+1]    ; DI = MEDIA/DRIVE PARAMETER TABLE
                             <1> RWV_MD_SEARH:
00001F95 3A770C              <1>         CMP     DH, [eDI+MD.RATE]       ; MATCH?
00001F98 741B                <1>  JE    short RWV_MD_FND  ; YES, GO GET 1ST SPECIFY BYTE
                             <1> RWV_NXT_MD:
                             <1>  ;ADD  BX,3              ; CHECK NEXT DRIVE TYPE
00001F9A 83C305              <1>  add   eBX, 5
00001F9D E2E3                <1>  LOOP  RWV_DR_SEARCH
00001F9F 5F                  <1>  POP   eDI               ; RESTORE DRIVE #
                             <1>
                             <1> ;----- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
                             <1>
                             <1> RWV_ASSUME:
00001FA0 BB[FA6A0000]        <1>  MOV   eBX, MD_TBL1            ; POINT TO 40 TRACK 250 KBS
00001FA5 F687[39710000]01    <1>  TEST  byte [DSK_STATE+eDI], TRK_CAPA ; TEST FOR 80 TRACK
00001FAC 740A                <1>  JZ    short RWV_MD_FND1 ; MUST BE 40 TRACK
00001FAE BB[146B0000]        <1>  MOV   eBX, MD_TBL3            ; POINT TO 80 TRACK 500 KBS
00001FB3 EB03                <1>  JMP   short RWV_MD_FND1 ; GO SPECIFY PARAMTERS
                             <1>
                             <1> ;----- CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE
                             <1>
                             <1> RWV_MD_FND:
00001FB5 89FB                <1>  MOV   eBX,eDI                 ; BX = MEDIA/DRIVE PARAMETER TABLE
00001FB7 5F                  <1>  POP   eDI               ; RESTORE DRIVE #
                             <1>
                             <1> ;----- SEND THE SPECIFY COMMAND TO THE CONTROLLER
                             <1>
                             <1> RWV_MD_FND1:
00001FB8 E882FEFFFF          <1>  CALL  SEND_SPEC_MD
00001FBD E864010000          <1>  CALL  CHK_LASTRATE            ; ZF=1 ATTEMP RATE IS SAME AS LAST RATE
00001FC2 7405                <1>  JZ    short RWV_DBL           ; YES,SKIP SEND RATE COMMAND
00001FC4 E83B010000          <1>  CALL  SEND_RATE         ; SEND DATA RATE TO NEC
                             <1> RWV_DBL:
00001FC9 53                  <1>  PUSH  eBX               ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
00001FCA E821040000          <1>  CALL  SETUP_DBL         ; CHECK FOR DOUBLE STEP
00001FCF 5B                  <1>  POP   eBX               ; RESTORE ADDRESS
00001FD0 7226                <1>  JC    short CHK_RET           ; ERROR FROM READ ID, POSSIBLE RETRY
00001FD2 6658                <1>  POP   AX                ; RESTORE NEC, DMA COMMAND
00001FD4 6650                <1>  PUSH  AX                ; SAVE NEC COMMAND
00001FD6 53                  <1>  PUSH  eBX               ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
00001FD7 E861010000          <1>  CALL  DMA_SETUP         ; SET UP THE DMA
00001FDC 5B                  <1>  POP   eBX
00001FDD 6658                <1>  POP   AX                ; RESTORE NEC COMMAND
00001FDF 722F                <1>  JC    short RWV_BAC           ; CHECK FOR DMA BOUNDARY ERROR
00001FE1 6650                <1>  PUSH  AX                ; SAVE NEC COMMAND
00001FE3 53                  <1>  PUSH  eBX               ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
00001FE4 E83B020000          <1>  CALL  NEC_INIT          ; INITIALIZE NEC
00001FE9 5B                  <1>  POP   eBX               ; RESTORE ADDRESS
00001FEA 720C                <1>  JC    short CHK_RET           ; ERROR - EXIT
00001FEC E865020000          <1>  CALL  RWV_COM                 ; OP CODE COMMON TO READ/WRITE/VERIFY
00001FF1 7205                <1>  JC    short CHK_RET           ; ERROR - EXIT
00001FF3 E8AA020000          <1>  CALL  NEC_TERM          ; TERMINATE, GET STATUS, ETC.
                             <1> CHK_RET:
00001FF8 E849030000          <1>  CALL  RETRY             ; CHECK FOR, SETUP RETRY
00001FFD 6658                <1>  POP   AX                ; RESTORE READ/WRITE/VERIFY PARAMETER
00001FFF 7305                <1>  JNC   short RWV_END           ; CY = 0 NO RETRY
00002001 E928FFFFFF          <1>         JMP     DO_AGAIN                ; CY = 1 MEANS RETRY
                             <1> RWV_END:
00002006 E8F3020000          <1>  CALL  DSTATE                  ; ESTABLISH STATE IF SUCCESSFUL
0000200B E886030000          <1>  CALL  NUM_TRANS         ; AL = NUMBER TRANSFERRED
                             <1> RWV_BAC:                       ; BAD DMA ERROR ENTRY
00002010 6650                <1>  PUSH  AX                ; SAVE NUMBER TRANSFERRED
00002012 E879FEFFFF          <1>  CALL  XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
00002017 6658                <1>  POP   AX                ; RESTORE NUMBER TRANSFERRED
00002019 E8B0030000          <1>  CALL  SETUP_END         ; VARIOUS CLEANUPS
0000201E C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; SETUP_STATE:     INITIALIZES START AND END RATES.
                             <1> ;-------------------------------------------------------------------------------
                             <1> SETUP_STATE:
0000201F F687[39710000]10    <1>  TEST  byte [DSK_STATE+eDI], MED_DET ; MEDIA DETERMINED ?
00002026 7537                <1>  JNZ   short J1C         ; NO STATES IF DETERMINED
00002028 66B84000            <1>         MOV     AX,(RATE_500*256)+RATE_300  ; AH = START RATE, AL = END RATE
0000202C F687[39710000]04    <1>  TEST  byte [DSK_STATE+eDI],DRV_DET ; DRIVE ?
00002033 740D                <1>  JZ    short AX_SET            ; DO NOT KNOW DRIVE
00002035 F687[39710000]02    <1>  TEST  byte [DSK_STATE+eDI], FMT_CAPA ; MULTI-RATE?
0000203C 7504                <1>  JNZ   short AX_SET            ; JUMP IF YES
0000203E 66B88080            <1>         MOV     AX,RATE_250*257         ; START A END RATE 250 FOR 360 DRIVE
```

```
                             <1> AX_SET:
00002042 80A7[39710000]1F    <1>  AND   byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP) ; TURN OFF THE RATE
00002049 08A7[39710000]      <1>  OR    [DSK_STATE+eDI], AH     ; RATE FIRST TO TRY
0000204F 8025[34710000]F3    <1>  AND   byte [LASTRATE], ~STRT_MSK ; ERASE LAST TO TRY RATE BITS
00002056 C0C804              <1>  ROR   AL,4               ; TO OPERATION LAST RATE LOCATION
00002059 0805[34710000]      <1>  OR    [LASTRATE], AL          ; LAST RATE
                             <1> J1C:
0000205F C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ;  FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
                             <1> ;-------------------------------------------------------------------------------
                             <1> FMT_INIT:
00002060 F687[39710000]10    <1>  TEST  byte [DSK_STATE+eDI], MED_DET ; IS MEDIA ESTABLISHED
00002067 7546                <1>  JNZ   short F1_OUT            ; IF SO RETURN
00002069 E82B040000          <1>  CALL  CMOS_TYPE          ; RETURN DRIVE TYPE IN AL
                             <1>  ;; 20/02/2015
                             <1>  ;;JC  short CL_DRV            ; ERROR IN CMOS ASSUME NO DRIVE
0000206E 7440                <1>  jz    short CL_DRV ;; 20/02/2015
00002070 FEC8                <1>  DEC   AL                 ; MAKE ZERO ORIGIN
                             <1>  ;;JS  short CL_DRV            ; NO DRIVE IF AL 0
00002072 8AA7[39710000]      <1>  MOV   AH, [DSK_STATE+eDI]     ; AH = CURRENT STATE
00002078 80E40F              <1>  AND   AH, ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR
0000207B 08C0                <1>  OR    AL,AL              ; CHECK FOR 360
0000207D 7505                <1>  JNZ   short N_360        ; IF 360 WILL BE 0
0000207F 80CC90              <1>  OR    AH,MED_DET+RATE_250     ; ESTABLISH MEDIA
00002082 EB25                <1>  JMP   SHORT SKP_STATE         ; SKIP OTHER STATE PROCESSING
                             <1> N_360:
00002084 FEC8                <1>  DEC   AL                 ; 1.2 M DRIVE
00002086 7505                <1>  JNZ   short N_12         ; JUMP IF NOT
                             <1> F1_RATE:
00002088 80CC10              <1>  OR    AH,MED_DET+RATE_500     ; SET FORMAT RATE
0000208B EB1C                <1>  JMP   SHORT SKP_STATE         ; SKIP OTHER STATE PROCESSING
                             <1> N_12:
0000208D FEC8                <1>  DEC   AL                 ; CHECK FOR TYPE 3
0000208F 750F                <1>  JNZ   short N_720        ; JUMP IF NOT
00002091 F6C404              <1>  TEST  AH,DRV_DET         ; IS DRIVE DETERMINED
00002094 7410                <1>  JZ    short ISNT_12           ; TREAT AS NON 1.2 DRIVE
00002096 F6C402              <1>  TEST  AH,FMT_CAPA        ; IS 1.2M
00002099 740B                <1>  JZ    short ISNT_12           ; JUMP IF NOT
0000209B 80CC50              <1>  OR    AH,MED_DET+RATE_300     ; RATE 300
0000209E EB09                <1>  JMP   SHORT SKP_STATE         ; CONTINUE
                             <1> N_720:
000020A0 FEC8                <1>  DEC   AL                 ; CHECK FOR TYPE 4
000020A2 750C                <1>  JNZ   short CL_DRV            ; NO DRIVE, CMOS BAD
000020A4 EBE2                <1>  JMP   SHORT F1_RATE
                             <1> ISNT_12:
000020A6 80CC90              <1>  OR    AH,MED_DET+RATE_250     ; MUST BE RATE 250
                             <1>
                             <1> SKP_STATE:
000020A9 88A7[39710000]      <1>  MOV   [DSK_STATE+eDI], AH     ; STORE AWAY
                             <1> F1_OUT:
000020AF C3                  <1>  RETn
                             <1> CL_DRV:
000020B0 30E4                <1>  XOR   AH,AH              ; CLEAR STATE
000020B2 EBF5                <1>  JMP   SHORT SKP_STATE         ; SAVE IT
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; MED_CHANGE
                             <1> ;CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
                             <1> ;CHECKS MEDIA CHANGE AGAIN.
                             <1> ;
                             <1> ; ON EXIT:   CY = 1 MEANS MEDIA CHANGE OR TIMEOUT
                             <1> ;       @DSKETTE_STATUS = ERROR CODE
                             <1> ;-------------------------------------------------------------------------------
                             <1> MED_CHANGE:
000020B4 E887060000          <1>  CALL  READ_DSKCHNG            ; READ DISK CHANCE LINE STATE
000020B9 7447                <1>  JZ    short MC_OUT            ; BYPASS HANDLING DISK CHANGE LINE
000020BB 80A7[39710000]EF    <1>  AND   byte [DSK_STATE+eDI], ~MED_DET ; CLEAR STATE FOR THIS DRIVE
                             <1>
                             <1> ;THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT
                             <1> ;ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
                             <1> ;BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).
                             <1>
000020C2 6689F9              <1>  MOV   CX,DI              ; CL = DRIVE 0
000020C5 B001                <1>  MOV   AL,1               ; MOTOR ON BIT MASK
000020C7 D2E0                <1>  SHL   AL,CL              ; TO APPROPRIATE POSITION
000020C9 F6D0                <1>  NOT   AL                 ; KEEP ALL BUT MOTOR ON
000020CB FA                  <1>  CLI                      ; NO INTERRUPTS
000020CC 2005[2A710000]      <1>  AND   [MOTOR_STATUS], AL      ; TURN MOTOR OFF INDICATOR
000020D2 FB                  <1>  STI                      ; INTERRUPTS ENABLED
000020D3 E80F040000          <1>  CALL  MOTOR_ON           ; TURN MOTOR ON
                             <1>
                             <1> ;----- THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
                             <1>
000020D8 E884F9FFFF          <1>  CALL  DSK_RESET          ; RESET NEC
000020DD B501                <1>  MOV   CH,01H                  ; MOVE TO CYLINDER 1
000020DF E8FE040000          <1>  CALL  SEEK               ; ISSUE SEEK
000020E4 30ED                <1>  XOR   CH,CH              ; MOVE TO CYLINDER 0
000020E6 E8F7040000          <1>  CALL  SEEK               ; ISSUE SEEK
000020EB C605[2C710000]06    <1>  MOV   byte [DSKETTE_STATUS], MEDIA_CHANGE ; STORE IN STATUS
                             <1> OK1:
```

```
000020F2 E849060000            <1>  CALL  READ_DSKCHNG             ; CHECK MEDIA CHANGED AGAIN
000020F7 7407                  <1>  JZ    short OK2          ; IF ACTIVE, NO DISKETTE, TIMEOUT
                               <1> OK4:
000020F9 C605[2C710000]80      <1>  MOV   byte [DSKETTE_STATUS], TIME_OUT ; TIMEOUT IF DRIVE EMPTY
                               <1> OK2:
00002100 F9                    <1>  STC                      ; MEDIA CHANGED, SET CY
00002101 C3                    <1>  RETn
                               <1> MC_OUT:
00002102 F8                    <1>  CLC                      ; NO MEDIA CHANGED, CLEAR CY
00002103 C3                    <1>  RETn
                               <1>
                               <1> ;-----------------------------------------------------------------------------
                               <1> ; SEND_RATE
                               <1> ;SENDS DATA RATE COMMAND TO NEC
                               <1> ; ON ENTRY:  DI = DRIVE #
                               <1> ; ON EXIT:   NONE
                               <1> ; REGISTERS ALTERED: DX
                               <1> ;-----------------------------------------------------------------------------
                               <1> SEND_RATE:
00002104 6650                  <1>  PUSH  AX                 ; SAVE REG.
00002106 8025[34710000]3F      <1>  AND   byte [LASTRATE], ~SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
0000210D 8A87[39710000]        <1>  MOV   AL, [DSK_STATE+eDI]      ; GET RATE STATE OF THIS DRIVE
00002113 24C0                  <1>  AND   AL,SEND_MSK        ; KEEP ONLY RATE BITS
00002115 0805[34710000]        <1>  OR    [LASTRATE], AL           ; SAVE NEW RATE FOR NEXT CHECK
0000211B C0C002                <1>  ROL   AL,2               ; MOVE TO BIT OUTPUT POSITIONS
0000211E 66BAF703              <1>  MOV   DX,03F7H           ; OUTPUT NEW DATA RATE
00002122 EE                    <1>  OUT   DX,AL
00002123 6658                  <1>  POP   AX                 ; RESTORE REG.
00002125 C3                    <1>  RETn
                               <1>
                               <1> ;-----------------------------------------------------------------------------
                               <1> ; CHK_LASTRATE
                               <1> ;CHECK PREVIOUS DATE RATE SNT TO THE CONTROLLER.
                               <1> ; ON ENTRY:
                               <1> ;DI = DRIVE #
                               <1> ; ON EXIT:
                               <1> ;ZF =  1 DATA RATE IS THE SAME AS THE LAST RATE SENT TO NEC
                               <1> ;ZF =  0 DATA RATE IS DIFFERENT FROM LAST RATE
                               <1> ; REGISTERS ALTERED: DX
                               <1> ;-----------------------------------------------------------------------------
                               <1> CHK_LASTRATE:
00002126 6650                  <1>  PUSH  AX                 ; SAVE REG
00002128 2225[34710000]        <1>  AND   AH, [LASTRATE]           ; GET LAST DATA RATE SELECTED
0000212E 8A87[39710000]        <1>  MOV   AL, [DSK_STATE+eDI]      ; GET RATE STATE OF THIS DRIVE
00002134 6625C0C0              <1>         AND     AX, SEND_MSK*257        ; KEEP ONLY RATE BITS OF BOTH
00002138 38E0                  <1>  CMP   AL, AH                   ; COMPARE TO PREVIOUSLY TRIED
                               <1>                           ; ZF = 1 RATE IS THE SAME
0000213A 6658                  <1>  POP   AX                 ; RESTORE REG.
0000213C C3                    <1>  RETn
                               <1>
                               <1> ;-----------------------------------------------------------------------------
                               <1> ; DMA_SETUP
                               <1> ;THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS.
                               <1> ;
                               <1> ; ON ENTRY:  AL = DMA COMMAND
                               <1> ;
                               <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                               <1> ;-----------------------------------------------------------------------------
                               <1>
                               <1> ; SI = Head #, # of Sectors or DASD Type
                               <1>
                               <1> ; 22/08/2015
                               <1> ; 08/02/2015 - Protected Mode Modification
                               <1> ; 06/02/2015 - 07/02/2015
                               <1> ; NOTE: Buffer address must be in 1st 16MB of Physical Memory (24 bit limit).
                               <1> ; (DMA Addres = Physical Address)
                               <1> ; (Retro UNIX 386 v1 Kernel/System Mode Virtual Address = Physical Address)
                               <1> ;
                               <1>
                               <1> ; 20/02/2015 modification (source: AWARD BIOS 1999, DMA_SETUP)
                               <1> ; 16/12/2014 (IODELAY)
                               <1>
                               <1> DMA_SETUP:
                               <1> ;; 20/02/2015
0000213D 8B5504                <1>  mov   edx, [ebp+4]             ; Buffer address
00002140 F7C2000000FF          <1>  test  edx, 0FF000000h          ; 16 MB limit (22/08/2015, bugfix)
00002146 756D                  <1>  jnz   short dma_bnd_err_stc
                               <1>  ;
00002148 6650                  <1>  push  ax                 ; DMA command
0000214A 52                    <1>  push  edx                ; *
0000214B B203                  <1>  mov   dl, 3              ; GET BYTES/SECTOR PARAMETER
0000214D E850030000            <1>  call  GET_PARM           ;
00002152 88E1                  <1>  mov   cl, ah                   ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
00002154 6689F0                <1>  mov   ax, si                   ; Sector count
00002157 88C4                  <1>  mov   ah, al                   ; AH =  # OF SECTORS
00002159 28C0                  <1>  sub   al, al                   ; AL = 0, AX = # SECTORS * 256
0000215B 66D1E8                <1>  shr   ax, 1              ; AX = # SECTORS * 128
0000215E 66D3E0                <1>  shl   ax, cl                   ; SHIFT BY PARAMETER VALUE
00002161 6648                  <1>  dec   ax                 ; -1 FOR DMA VALUE
00002163 6689C1                <1>  mov   cx, ax
00002166 5A                    <1>  pop   edx                ; *
00002167 6658                  <1>  pop   ax
```

```
00002169 3C42                <1>  cmp   al, 42h
0000216B 7507                <1>         jne     short NOT_VERF
0000216D BA0000FF00          <1>  mov   edx, 0FF0000h
00002172 EB08                <1>  jmp   short J33
                             <1> NOT_VERF:
00002174 6601CA              <1>  add   dx, cx                  ; check for overflow
00002177 723D                <1>  jc    short dma_bnd_err
                             <1>  ;
00002179 6629CA              <1>  sub   dx, cx                  ; Restore start address
                             <1> J33:
0000217C FA                  <1>  CLI                    ; DISABLE INTERRUPTS DURING DMA SET-UP
0000217D E60C                <1>  OUT   DMA+12,AL        ; SET THE FIRST/LA5T F/F
                             <1>  IODELAY                      ; WAIT FOR I/O
0000217F EB00                <2>  jmp short $+2
00002181 EB00                <2>  jmp short $+2
00002183 E60B                <1>  OUT   DMA+11,AL        ; OUTPUT THE MODE BYTE
00002185 89D0                <1>  mov   eax, edx         ; Buffer address
00002187 E604                <1>  OUT   DMA+4,AL         ; OUTPUT LOW ADDRESS
                             <1>  IODELAY                      ; WAIT FOR I/O
00002189 EB00                <2>  jmp short $+2
0000218B EB00                <2>  jmp short $+2
0000218D 88E0                <1>  MOV   AL,AH
0000218F E604                <1>  OUT   DMA+4,AL         ; OUTPUT HIGH ADDRESS
00002191 C1E810              <1>  shr   eax, 16
                             <1>  IODELAY                      ; I/O WAIT STATE
00002194 EB00                <2>  jmp short $+2
00002196 EB00                <2>  jmp short $+2
00002198 E681                <1>  OUT   081H,AL                 ; OUTPUT highest BITS TO PAGE REGISTER
                             <1>  IODELAY
0000219A EB00                <2>  jmp short $+2
0000219C EB00                <2>  jmp short $+2
0000219E 6689C8              <1>  mov   ax, cx                  ; Byte count - 1
000021A1 E605                <1>  OUT   DMA+5,AL         ; LOW BYTE OF COUNT
                             <1>  IODELAY                      ; WAIT FOR I/O
000021A3 EB00                <2>  jmp short $+2
000021A5 EB00                <2>  jmp short $+2
000021A7 88E0                <1>  MOV   AL, AH
000021A9 E605                <1>  OUT   DMA+5,AL         ; HIGH BYTE OF COUNT
                             <1>  IODELAY
000021AB EB00                <2>  jmp short $+2
000021AD EB00                <2>  jmp short $+2
000021AF FB                  <1>  STI                    ; RE-ENABLE INTERRUPTS
000021B0 B002                <1>  MOV   AL, 2            ; MODE FOR 8237
000021B2 E60A                <1>  OUT   DMA+10, AL       ; INITIALIZE THE DISKETTE CHANNEL
000021B4 C3                  <1>  retn
                             <1> dma_bnd_err_stc:
000021B5 F9                  <1>  stc
                             <1> dma_bnd_err:
000021B6 C605[2C710000]09    <1>  MOV   byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
000021BD C3                  <1>  RETn                   ; CY SET BY ABOVE IF ERROR
                             <1>
                             <1> ;; 16/12/2014
                             <1> ;;     CLI                     ; DISABLE INTERRUPTS DURING DMA SET-UP
                             <1> ;;     OUT   DMA+12,AL         ; SET THE FIRST/LA5T F/F
                             <1> ;;     ;JMP  $+2               ; WAIT FOR I/O
                             <1> ;;     IODELAY
                             <1> ;;     OUT   DMA+11,AL         ; OUTPUT THE MODE BYTE
                             <1> ;;     ;SIODELAY
                             <1> ;;      ;CMP AL, 42H                 ; DMA VERIFY COMMAND
                             <1> ;;      ;JNE short NOT_VERF          ; NO
                             <1> ;;      ;XOR AX, AX                  ; START ADDRESS
                             <1> ;;      ;JMP SHORT J33
                             <1> ;;;NOT_VERF:
                             <1> ;;     ;MOV  AX,ES             ; GET THE ES VALUE
                             <1> ;;     ;ROL  AX,4              ; ROTATE LEFT
                             <1> ;;     ;MOV  CH,AL             ; GET HIGHEST NIBBLE OF ES TO CH
                             <1> ;;     ;AND  AL,11110000B            ; ZERO THE LOW NIBBLE FROM SEGMENT
                             <1> ;;     ;ADD  AX,[BP+2]         ; TEST FOR CARRY FROM ADDITION
                             <1> ;;     mov   eax, [ebp+4] ; 06/02/2015
                             <1> ;;     ;JNC  short J33
                             <1> ;;     ;INC  CH                ; CARRY MEANS HIGH 4 BITS MUST BE INC
                             <1> ;;;J33:
                             <1> ;;     PUSH  eAX               ; SAVE START ADDRESS
                             <1> ;;     OUT   DMA+4,AL          ; OUTPUT LOW ADDRESS
                             <1> ;;     ;JMP  $+2               ; WAIT FOR I/O
                             <1> ;;     IODELAY
                             <1> ;;     MOV   AL,AH
                             <1> ;;     OUT   DMA+4,AL           ; OUTPUT HIGH ADDRESS
                             <1> ;;     shr   eax, 16          ; 07/02/2015
                             <1> ;;     ;MOV  AL,CH             ; GET HIGH 4 BITS
                             <1> ;;     ;JMP  $+2               ; I/O WAIT STATE
                             <1> ;;     IODELAY
                             <1> ;;     ;AND  AL,00001111B
                             <1> ;;     OUT   081H,AL                 ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
                             <1> ;;     ;SIODELAY
                             <1> ;;
                             <1> ;;;----- DETERMINE COUNT
                             <1> ;;     sub   eax, eax ; 08/02/2015
                             <1> ;;     MOV   AX, SI                  ; AL =  # OF SECTORS
                             <1> ;;     XCHG  AL, AH                  ; AH =  # OF SECTORS
                             <1> ;;     SUB   AL, AL                  ; AL = 0, AX = # SECTORS * 256
                             <1> ;;     SHR   AX, 1             ; AX = # SECTORS * 128
                             <1> ;;     PUSH  AX                ; SAVE # OF SECTORS * 128
                             <1> ;;     MOV   DL, 3             ; GET BYTES/SECTOR PARAMETER
                             <1> ;;     CALL  GET_PARM          ; "
                             <1> ;;     MOV   CL,AH             ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
                             <1> ;;     POP   AX                ; AX = # SECTORS * 128
                             <1> ;;     SHL   AX,CL             ; SHIFT BY PARAMETER VALUE
                             <1> ;;     DEC   AX                ; -1 FOR DMA VALUE
                             <1> ;;     PUSH  eAX  ; 08/02/2015 ; SAVE COUNT VALUE
                             <1> ;;     OUT   DMA+5,AL          ; LOW BYTE OF COUNT
                             <1> ;;     ;JMP  $+2               ; WAIT FOR I/O
                             <1> ;;     IODELAY
                             <1> ;;     MOV   AL, AH
                             <1> ;;     OUT   DMA+5,AL          ; HIGH BYTE OF COUNT
                             <1> ;;     ;IODELAY
                             <1> ;;     STI                     ; RE-ENABLE INTERRUPTS
                             <1> ;;     POP   eCX  ; 08/02/2015       ; RECOVER COUNT VALUE
                             <1> ;;     POP   eAX  ; 08/02/2015 ; RECOVER ADDRESS VALUE
                             <1> ;;     ;ADD  AX, CX                  ; ADD, TEST FOR 64K OVERFLOW
                             <1> ;;     add   ecx, eax ; 08/02/2015
                             <1> ;;     MOV   AL, 2             ; MODE FOR 8237
                             <1> ;;     ;JMP  $+2               ; WAIT FOR I/O
                             <1> ;;     SIODELAY
                             <1> ;;     OUT   DMA+10, AL        ; INITIALIZE THE DISKETTE CHANNEL
                             <1> ;;     ;JNC  short NO_BAD            ; CHECK FOR ERROR
                             <1> ;;     jc    short dma_bnd_err ; 08/02/2015
                             <1> ;;     and   ecx, 0FFF00000h ; 16 MB limit
                             <1> ;;     jz    short NO_BAD
                             <1> ;;dma_bnd_err:
                             <1> ;;     MOV   byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
                             <1> ;;NO_BAD:
                             <1> ;;     RETn                    ; CY SET BY ABOVE IF ERROR
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; FMTDMA_SET
                             <1> ;THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT OPERATION.
                             <1> ;
                             <1> ; ON ENTRY:  NOTHING REQUIRED
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1>
                             <1> FMTDMA_SET:
                             <1> ;; 20/02/2015 modification
000021BE 8B5504              <1>  mov   edx, [ebp+4]            ; Buffer address
000021C1 F7C20000F0FF        <1>  test  edx, 0FFF00000h         ; 16 MB limit
000021C7 75EC                <1>  jnz   short dma_bnd_err_stc
                             <1>  ;
000021C9 6652                <1>  push  dx                ; *
000021CB B204                <1>  mov   DL, 4             ; SECTORS/TRACK VALUE IN PARM TABLE
000021CD E8D0020000          <1>  call  GET_PARM          ; "
000021D2 88E0                <1>  mov   al, ah                  ; AL = SECTORS/TRACK VALUE
000021D4 28E4                <1>  sub   ah, ah                  ; AX = SECTORS/TRACK VALUE
000021D6 66C1E002            <1>  shl   ax, 2             ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
000021DA 6648                <1>  dec   ax                ; -1 FOR DMA VALUE
000021DC 6689C1              <1>  mov   cx, ax
000021DF 665A                <1>  pop   dx                ; *
000021E1 6601CA              <1>  add   dx, cx                  ; check for overflow
000021E4 72D0                <1>  jc    short dma_bnd_err
                             <1>  ;
000021E6 6629CA              <1>  sub   dx, cx                  ; Restore start address
                             <1>  ;
000021E9 B04A                <1>  MOV   AL, 04AH          ; WILL WRITE TO THE DISKETTE
000021EB FA                  <1>  CLI                     ; DISABLE INTERRUPTS DURING DMA SET-UP
000021EC E60C                <1>  OUT   DMA+12,AL         ; SET THE FIRST/LA5T F/F
                             <1>  IODELAY                       ; WAIT FOR I/O
000021EE EB00                <2> jmp short $+2
000021F0 EB00                <2> jmp short $+2
000021F2 E60B                <1>  OUT   DMA+11,AL         ; OUTPUT THE MODE BYTE
000021F4 89D0                <1>  mov   eax, edx          ; Buffer address
000021F6 E604                <1>  OUT   DMA+4,AL          ; OUTPUT LOW ADDRESS
                             <1>  IODELAY                       ; WAIT FOR I/O
000021F8 EB00                <2> jmp short $+2
000021FA EB00                <2> jmp short $+2
000021FC 88E0                <1>  MOV   AL,AH
000021FE E604                <1>  OUT   DMA+4,AL          ; OUTPUT HIGH ADDRESS
00002200 C1E810              <1>  shr   eax, 16
                             <1>  IODELAY                       ; I/O WAIT STATE
00002203 EB00                <2> jmp short $+2
00002205 EB00                <2> jmp short $+2
00002207 E681                <1>  OUT   081H,AL                 ; OUTPUT highest BITS TO PAGE REGISTER
                             <1>  IODELAY
00002209 EB00                <2> jmp short $+2
0000220B EB00                <2> jmp short $+2
0000220D 6689C8              <1>  mov   ax, cx                  ; Byte count - 1
00002210 E605                <1>  OUT   DMA+5,AL          ; LOW BYTE OF COUNT
                             <1>  IODELAY                       ; WAIT FOR I/O
00002212 EB00                <2> jmp short $+2
00002214 EB00                <2> jmp short $+2
00002216 88E0                <1>  MOV   AL, AH
00002218 E605                <1>  OUT   DMA+5,AL          ; HIGH BYTE OF COUNT
                             <1>  IODELAY
0000221A EB00                <2> jmp short $+2
```

```
0000221C EB00                <2>  jmp short $+2
0000221E FB                  <1>  STI                     ; RE-ENABLE INTERRUPTS
0000221F B002                <1>  MOV   AL, 2             ; MODE FOR 8237
00002221 E60A                <1>  OUT   DMA+10, AL        ; INITIALIZE THE DISKETTE CHANNEL
00002223 C3                  <1>  retn
                             <1>
                             <1> ;; 08/02/2015 - Protected Mode Modification
                             <1> ;;     MOV   AL, 04AH          ; WILL WRITE TO THE DISKETTE
                             <1> ;;     CLI                     ; DISABLE INTERRUPTS DURING DMA SET-UP
                             <1> ;;     OUT   DMA+12,AL         ; SET THE FIRST/LA5T F/F
                             <1> ;;     ;JMP  $+2               ; WAIT FOR I/O
                             <1> ;;     IODELAY
                             <1> ;;     OUT   DMA+11,AL         ; OUTPUT THE MODE BYTE
                             <1> ;;     ;MOV  AX,ES             ; GET THE ES VALUE
                             <1> ;;     ;ROL  AX,4              ; ROTATE LEFT
                             <1> ;;     ;MOV  CH,AL             ; GET HIGHEST NIBBLE OF ES TO CH
                             <1> ;;     ;AND  AL,11110000B            ; ZERO THE LOW NIBBLE FROM SEGMENT
                             <1> ;;     ;ADD  AX,[BP+2]         ; TEST FOR CARRY FROM ADDITION
                             <1> ;;     ;JNC  short J33A
                             <1> ;;     ;INC  CH                ; CARRY MEANS HIGH 4 BITS MUST BE INC
                             <1> ;;     mov   eax, [ebp+4] ; 08/02/2015
                             <1> ;;;J33A:
                             <1> ;;     PUSH  eAX ; 08/02/2015  ; SAVE START ADDRESS
                             <1> ;;     OUT   DMA+4,AL          ; OUTPUT LOW ADDRESS
                             <1> ;;     ;JMP  $+2               ; WAIT FOR I/O
                             <1> ;;     IODELAY
                             <1> ;;     MOV   AL,AH
                             <1> ;;     OUT   DMA+4,AL          ; OUTPUT HIGH ADDRESS
                             <1> ;;     shr   eax, 16 ; 08/02/2015
                             <1> ;;     ;MOV  AL,CH             ; GET HIGH 4 BITS
                             <1> ;;     ;JMP  $+2               ; I/O WAIT STATE
                             <1> ;;     IODELAY
                             <1> ;;     ;AND  AL,00001111B
                             <1> ;;     OUT   081H,AL                 ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
                             <1> ;;
                             <1> ;;;----- DETERMINE COUNT
                             <1> ;;     sub   eax, eax ; 08/02/2015
                             <1> ;;     MOV   DL, 4             ; SECTORS/TRACK VALUE IN PARM TABLE
                             <1> ;;     CALL  GET_PARM          ; "
                             <1> ;;     XCHG  AL, AH                  ; AL = SECTORS/TRACK VALUE
                             <1> ;;     SUB   AH, AH                  ; AX = SECTORS/TRACK VALUE
                             <1> ;;     SHL   AX, 2             ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
                             <1> ;;     DEC   AX                ; -1 FOR DMA VALUE
                             <1> ;;     PUSH  eAX   ; 08/02/2015      ; SAVE # OF BYTES TO BE TRANSFERED
                             <1> ;;     OUT   DMA+5,AL          ; LOW BYTE OF COUNT
                             <1> ;;     ;JMP  $+2               ; WAIT FOR I/O
                             <1> ;;     IODELAY
                             <1> ;;     MOV   AL, AH
                             <1> ;;     OUT   DMA+5,AL          ; HIGH BYTE OF COUNT
                             <1> ;;     STI                     ; RE-ENABLE INTERRUPTS
                             <1> ;;     POP   eCX   ; 08/02/2015      ; RECOVER COUNT VALUE
                             <1> ;;     POP   eAX   ; 08/02/2015      ; RECOVER ADDRESS VALUE
                             <1> ;;     ;ADD  AX, CX                  ; ADD, TEST FOR 64K OVERFLOW
                             <1> ;;     add   ecx, eax ; 08/02/2015
                             <1> ;;     MOV   AL, 2             ; MODE FOR 8237
                             <1> ;;     ;JMP  $+2               ; WAIT FOR I/O
                             <1> ;;     SIODELAY
                             <1> ;;     OUT   DMA+10, AL        ; INITIALIZE THE DISKETTE CHANNEL
                             <1> ;;     ;JNC  short FMTDMA_OK         ; CHECK FOR ERROR
                             <1> ;;     jc    short fmtdma_bnd_err ; 08/02/2015
                             <1> ;;     and   ecx, 0FFF00000h  ; 16 MB limit
                             <1> ;;     jz    short FMTDMA_OK
                             <1> ;;     stc   ; 20/02/2015
                             <1> ;;fmtdma_bnd_err:
                             <1> ;;     MOV   byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
                             <1> ;;FMTDMA_OK:
                             <1> ;;     RETn                    ; CY SET BY ABOVE IF ERROR
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; NEC_INIT
                             <1> ;THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND INITIALIZES
                             <1> ;THE NEC FOR THE READ/WRITE/VERIFY/FORMAT OPERATION.
                             <1> ;
                             <1> ; ON ENTRY:  AH = NEC COMMAND TO BE PERFORMED
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;--------------------------------------------------------------------------------
                             <1> NEC_INIT:
00002224 6650                <1>  PUSH  AX                ; SAVE NEC COMMAND
00002226 E8BC020000          <1>  CALL  MOTOR_ON          ; TURN MOTOR ON FOR SPECIFIC DRIVE
                             <1>
                             <1> ;----- DO THE SEEK OPERATION
                             <1>
0000222B 8A6D01              <1>  MOV   CH,[eBP+1]        ; CH = TRACK #
0000222E E8AF030000          <1>  CALL  SEEK              ; MOVE TO CORRECT TRACK
00002233 6658                <1>  POP   AX                ; RECOVER COMMAND
00002235 721E                <1>  JC    short ER_1        ; ERROR ON SEEK
00002237 BB[55220000]        <1>  MOV   eBX, ER_1         ; LOAD ERROR ADDRESS
0000223C 53                  <1>  PUSH  eBX               ; PUSH NEC_OUT ERROR RETURN
                             <1>
                             <1> ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
                             <1>
```

```
0000223D E866030000          <1>  CALL  NEC_OUTPUT        ; OUTPUT THE OPERATION COMMAND
00002242 6689F0              <1>  MOV   AX,SI             ; AH = HEAD #
00002245 89FB                <1>  MOV   eBX,eDI                 ; BL = DRIVE #
00002247 C0E402              <1>  SAL   AH,2              ; MOVE IT TO BIT 2
0000224A 80E404              <1>  AND   AH,00000100B            ; ISOLATE THAT BIT
0000224D 08DC                <1>  OR    AH,BL             ; OR IN THE DRIVE NUMBER
0000224F E854030000          <1>  CALL  NEC_OUTPUT        ; FALL THRU CY SET IF ERROR
00002254 5B                  <1>  POP   eBX               ; THROW AWAY ERROR RETURN
                             <1> ER_1:
00002255 C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; RWV_COM
                             <1> ;THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC TO THE
                             <1> ;READ/WRITE/VERIFY OPERATIONS.
                             <1> ;
                             <1> ; ON ENTRY:  CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> RWV_COM:
00002256 B8[A1220000]        <1>  MOV   eAX, ER_2         ; LOAD ERROR ADDRESS
0000225B 50                  <1>  PUSH  eAX               ; PUSH NEC_OUT ERROR RETURN
0000225C 8A6501              <1>  MOV   AH,[eBP+1]        ; OUTPUT TRACK #
0000225F E844030000          <1>  CALL  NEC_OUTPUT
00002264 6689F0              <1>  MOV   AX,SI             ; OUTPUT HEAD #
00002267 E83C030000          <1>  CALL  NEC_OUTPUT
0000226C 8A6500              <1>         MOV     AH,[eBP]                ; OUTPUT SECTOR #
0000226F E834030000          <1>  CALL  NEC_OUTPUT
00002274 B203                <1>  MOV   DL,3              ; BYTES/SECTOR PARAMETER FROM BLOCK
00002276 E827020000          <1>  CALL  GET_PARM          ; ... TO THE NEC
0000227B E828030000          <1>  CALL  NEC_OUTPUT        ; OUTPUT TO CONTROLLER
00002280 B204                <1>  MOV   DL,4              ; EOT PARAMETER FROM BLOCK
00002282 E81B020000          <1>  CALL  GET_PARM          ; ... TO THE NEC
00002287 E81C030000          <1>  CALL  NEC_OUTPUT        ; OUTPUT TO CONTROLLER
0000228C 8A6305              <1>         MOV     AH, [eBX+MD.GAP]        ; GET GAP LENGTH
                             <1> _R15:
0000228F E814030000          <1>  CALL  NEC_OUTPUT
00002294 B206                <1>  MOV   DL,6              ; DTL PARAMETER PROM BLOCK
00002296 E807020000          <1>  CALL  GET_PARM          ;  TO THE NEC
0000229B E808030000          <1>  CALL  NEC_OUTPUT        ; OUTPUT TO CONTROLLER
000022A0 58                  <1>  POP   eAX               ; THROW AWAY ERROR EXIT
                             <1> ER_2:
000022A1 C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; NEC_TERM
                             <1> ;THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS THE STATUS
                             <1> ;FROM THE NEC FOR THE READ/WRITE/VERIFY/FORWAT OPERATION.
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;-------------------------------------------------------------------------------
                             <1> NEC_TERM:
                             <1>
                             <1> ;----- LET THE OPERATION HAPPEN
                             <1>
000022A2 56                  <1>  PUSH  eSI               ; SAVE HEAD #, # OF SECTORS
000022A3 E80D040000          <1>  CALL  WAIT_INT          ; WAIT FOR THE INTERRUPT
000022A8 9C                  <1>  PUSHF
000022A9 E837040000          <1>  CALL  RESULTS                 ; GET THE NEC STATUS
000022AE 724B                <1>  JC    short SET_END_POP
000022B0 9D                  <1>  POPF
000022B1 723E                <1>  JC    short SET_END           ; LOOK FOR ERROR
                             <1>
                             <1> ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER
                             <1>
000022B3 FC                  <1>  CLD                     ; SET THE CORRECT DIRECTION
000022B4 BE[2D710000]        <1>  MOV   eSI, NEC_STATUS         ; POINT TO STATUS FIELD
000022B9 AC                  <1>  lodsb                   ; GET ST0
000022BA 24C0                <1>  AND   AL,11000000B            ; TEST FOR NORMAL TERMINATION
000022BC 7433                <1>  JZ    short SET_END
000022BE 3C40                <1>  CMP   AL,01000000B            ; TEST FOR ABNORMAL TERMINATION
000022C0 7527                <1>  JNZ   short J18         ; NOT ABNORMAL, BAD NEC
                             <1>
                             <1> ;----- ABNORMAL TERMINATION, FIND OUT WHY
                             <1>
000022C2 AC                  <1>  lodsb                   ; GET ST1
000022C3 D0E0                <1>  SAL   AL,1              ; TEST FOR EDT FOUND
000022C5 B404                <1>  MOV   AH,RECORD_NOT_FND
000022C7 7222                <1>  JC    short J19
000022C9 C0E002              <1>  SAL   AL,2
000022CC B410                <1>  MOV   AH,BAD_CRC
000022CE 721B                <1>  JC    short J19
000022D0 D0E0                <1>  SAL   AL,1              ; TEST FOR DMA OVERRUN
000022D2 B408                <1>  MOV   AH,BAD_DMA
000022D4 7215                <1>  JC    short J19
000022D6 C0E002              <1>  SAL   AL,2              ; TEST FOR RECORD NOT FOUND
000022D9 B404                <1>  MOV   AH,RECORD_NOT_FND
000022DB 720E                <1>  JC    short J19
000022DD D0E0                <1>  SAL   AL,1
000022DF B403                <1>  MOV   AH,WRITE_PROTECT  ; TEST FOR WRITE_PROTECT
000022E1 7208                <1>  JC    short J19
000022E3 D0E0                <1>  SAL   AL,1              ; TEST MISSING ADDRESS MARK
```

```
000022E5 B402                 <1>  MOV   AH,BAD_ADDR_MARK
000022E7 7202                 <1>  JC    short J19
                              <1>
                              <1> ;----- NEC MUST HAVE FAILED
                              <1> J18:
000022E9 B420                 <1>  MOV   AH,BAD_NEC
                              <1> J19:
000022EB 0825[2C710000]       <1>  OR    [DSKETTE_STATUS], AH
                              <1> SET_END:
000022F1 803D[2C710000]01     <1>  CMP   byte [DSKETTE_STATUS], 1 ; SET ERROR CONDITION
000022F8 F5                   <1>  CMC
000022F9 5E                   <1>  POP   eSI
000022FA C3                   <1>  RETn                    ; RESTORE HEAD #, # OF SECTORS
                              <1>
                              <1> SET_END_POP:
000022FB 9D                   <1>  POPF
000022FC EBF3                 <1>  JMP   SHORT SET_END
                              <1>
                              <1> ;--------------------------------------------------------------------------------
                              <1> ; DSTATE:    ESTABLISH STATE UPON SUCCESSFUL OPERATION.
                              <1> ;--------------------------------------------------------------------------------
                              <1> DSTATE:
000022FE 803D[2C710000]00     <1>  CMP   byte [DSKETTE_STATUS],0 ; CHECK FOR ERROR
00002305 753E                 <1>  JNZ   short SETBAC                ; IF ERROR JUMP
00002307 808F[39710000]10     <1>  OR    byte [DSK_STATE+eDI],MED_DET ; NO ERROR, MARK MEDIA AS DETERMINED
0000230E F687[39710000]04     <1>  TEST  byte [DSK_STATE+eDI],DRV_DET ; DRIVE DETERMINED ?
00002315 752E                 <1>  JNZ   short SETBAC            ; IF DETERMINED NO TRY TO DETERMINE
00002317 8A87[39710000]       <1>  MOV   AL,[DSK_STATE+eDI]      ; LOAD STATE
0000231D 24C0                 <1>  AND   AL,RATE_MSK       ; KEEP ONLY RATE
0000231F 3C80                 <1>  CMP   AL,RATE_250       ; RATE 250 ?
00002321 751B                 <1>  JNE   short M_12        ; NO, MUST BE 1.2M OR 1.44M DRIVE
                              <1>
                              <1> ;----- CHECK IF IT IS 1.44M
                              <1>
00002323 E871010000           <1>  CALL  CMOS_TYPE         ; RETURN DRIVE TYPE IN (AL)
                              <1>  ;;20/02/2015
                              <1>  ;;JC  short M_12        ; CMOS BAD
00002328 7414                 <1>  jz    short M_12 ;; 20/02/2015
0000232A 3C04                 <1>  CMP   AL, 4             ; 1.44MB DRIVE ?
0000232C 7410                 <1>  JE    short M_12        ; YES
                              <1> M_720:
0000232E 80A7[39710000]FD     <1>  AND   byte [DSK_STATE+eDI], ~FMT_CAPA ; TURN OFF FORMAT CAPABILITY
00002335 808F[39710000]04     <1>  OR    byte [DSK_STATE+eDI],DRV_DET  ; MARK DRIVE DETERMINED
0000233C EB07                 <1>  JMP   SHORT SETBAC            ; BACK
                              <1> M_12:
0000233E 808F[39710000]06     <1>  OR    byte [DSK_STATE+eDI],DRV_DET+FMT_CAPA
                              <1>                          ; TURN ON DETERMINED & FMT CAPA
                              <1> SETBAC:
00002345 C3                   <1>  RETn
                              <1>
                              <1> ;--------------------------------------------------------------------------------
                              <1> ; RETRY
                              <1> ;DETERMINES WHETHER A RETRY IS NECESSARY.
                              <1> ;IF RETRY IS REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
                              <1> ;
                              <1> ; ON EXIT:   CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
                              <1> ;--------------------------------------------------------------------------------
                              <1> RETRY:
00002346 803D[2C710000]00     <1>  CMP   byte [DSKETTE_STATUS],0 ; GET STATUS OF OPERATION
0000234D 7445                 <1>  JZ    short NO_RETRY          ; SUCCESSFUL OPERATION
0000234F 803D[2C710000]80     <1>  CMP   byte [DSKETTE_STATUS],TIME_OUT ; IF TIME OUT NO RETRY
00002356 743C                 <1>  JZ    short NO_RETRY
00002358 8AA7[39710000]       <1>  MOV   AH,[DSK_STATE+eDI]      ; GET MEDIA STATE OF DRIVE
0000235E F6C410               <1>  TEST  AH,MED_DET        ; ESTABLISHED/DETERMINED ?
00002361 7531                 <1>  JNZ   short NO_RETRY          ; IF ESTABLISHED STATE THEN TRUE ERROR
00002363 80E4C0               <1>  AND   AH,RATE_MSK       ; ISOLATE RATE
00002366 8A2D[34710000]       <1>  MOV   CH,[LASTRATE]           ; GET START OPERATION STATE
0000236C C0C504               <1>  ROL   CH,4              ; TO CORRESPONDING BITS
0000236F 80E5C0               <1>  AND   CH,RATE_MSK       ; ISOLATE RATE BITS
00002372 38E5                 <1>  CMP   CH,AH             ; ALL RATES TRIED
00002374 741E                 <1>  JE    short NO_RETRY          ; IF YES, THEN TRUE ERROR
                              <1>
                              <1> ;SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
                              <1> ;  00000000B (500) -> 10000000B (250)
                              <1> ;  10000000B (250) -> 01000000B (300)
                              <1> ;  01000000B (300) -> 00000000B (500)
                              <1>
00002376 80FC01               <1>  CMP   AH,RATE_500+1           ; SET CY FOR RATE 500
00002379 D0DC                 <1>  RCR   AH,1              ; TO NEXT STATE
0000237B 80E4C0               <1>  AND   AH,RATE_MSK       ; KEEP ONLY RATE BITS
0000237E 80A7[39710000]1F     <1>  AND   byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP)
                              <1>                          ; RATE, DBL STEP OFF
00002385 08A7[39710000]       <1>  OR    [DSK_STATE+eDI],AH      ; TURN ON NEW RATE
0000238B C605[2C710000]00     <1>  MOV   byte [DSKETTE_STATUS],0 ; RESET STATUS FOR RETRY
00002392 F9                   <1>  STC                     ; SET CARRY FOR RETRY
00002393 C3                   <1>  RETn                    ; RETRY RETURN
                              <1>
                              <1> NO_RETRY:
00002394 F8                   <1>  CLC                     ; CLEAR CARRY NO RETRY
00002395 C3                   <1>  RETn                    ; NO RETRY RETURN
                              <1>
                              <1> ;--------------------------------------------------------------------------------
```

```
                             <1> ; NUM_TRANS
                             <1> ;THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT WERE
                             <1> ;ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
                             <1> ;
                             <1> ; ON ENTRY:  [BP+1] = TRACK
                             <1> ;       SI-HI  = HEAD
                             <1> ;       [BP]   = START SECTOR
                             <1> ;
                             <1> ; ON EXIT:   AL = NUMBER ACTUALLY TRANSFERRED
                             <1> ;--------------------------------------------------------------------------------
                             <1> NUM_TRANS:
00002396 30C0                <1>  XOR   AL,AL             ; CLEAR FOR ERROR
00002398 803D[2C710000]00    <1>  CMP   byte [DSKETTE_STATUS],0 ; CHECK FOR ERROR
0000239F 752C                <1>  JNZ   NT_OUT                  ; IF ERROR 0 TRANSFERRED
000023A1 B204                <1>  MOV   DL,4              ; SECTORS/TRACK OFFSET TO DL
000023A3 E8FA000000          <1>  CALL  GET_PARM          ; AH = SECTORS/TRACK
000023A8 8A1D[32710000]      <1>  MOV   BL, [NEC_STATUS+5]      ; GET ENDING SECTOR
000023AE 6689F1              <1>  MOV   CX,SI             ; CH = HEAD # STARTED
000023B1 3A2D[31710000]      <1>  CMP   CH, [NEC_STATUS+4]      ; GET HEAD ENDED UP ON
000023B7 750D                <1>  JNZ   DIF_HD                  ; IF ON SAME HEAD, THEN NO ADJUST
000023B9 8A2D[30710000]      <1>  MOV   CH, [NEC_STATUS+3]      ; GET TRACK ENDED UP ON
000023BF 3A6D01              <1>  CMP   CH,[eBP+1]        ; IS IT ASKED FOR TRACK
000023C2 7404                <1>  JZ    short SAME_TRK          ; IF SAME TRACK NO INCREASE
000023C4 00E3                <1>  ADD   BL,AH             ; ADD SECTORS/TRACK
                             <1> DIF_HD:
000023C6 00E3                <1>  ADD   BL,AH             ; ADD SECTORS/TRACK
                             <1> SAME_TRK:
000023C8 2A5D00              <1>  SUB   BL,[eBP]          ; SUBTRACT START FROM END
000023CB 88D8                <1>  MOV   AL,BL             ; TO AL
                             <1> NT_OUT:
000023CD C3                  <1>  RETn
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; SETUP_END
                             <1> ;RESTORES @MOTOR_COUNT TO PARAMETER PROVIDED IN TABLE
                             <1> ;AND LOADS @DSKETTE_STATUS TO AH, AND SETS CY.
                             <1> ;
                             <1> ; ON EXIT:
                             <1> ;AH, @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                             <1> ;--------------------------------------------------------------------------------
                             <1> SETUP_END:
000023CE B202                <1>  MOV   DL,2              ; GET THE MOTOR WAIT PARAMETER
000023D0 6650                <1>  PUSH  AX                ; SAVE NUMBER TRANSFERRED
000023D2 E8CB000000          <1>  CALL  GET_PARM
000023D7 8825[2B710000]      <1>  MOV   [MOTOR_COUNT],AH  ; STORE UPON RETURN
000023DD 6658                <1>  POP   AX                ; RESTORE NUMBER TRANSFERRED
000023DF 8A25[2C710000]      <1>  MOV   AH, [DSKETTE_STATUS]    ; GET STATUS OF OPERATION
000023E5 08E4                <1>  OR    AH,AH             ; CHECK FOR ERROR
000023E7 7402                <1>  JZ    short NUN_ERR                 ; NO ERROR
000023E9 30C0                <1>  XOR   AL,AL             ; CLEAR NUMBER RETURNED
                             <1> NUN_ERR:
000023EB 80FC01              <1>  CMP   AH,1              ; SET THE CARRY FLAG TO INDICATE
000023EE F5                  <1>  CMC                     ; SUCCESS OR FAILURE
000023EF C3                  <1>  RETn
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; SETUP_DBL
                             <1> ;CHECK DOUBLE STEP.
                             <1> ;
                             <1> ; ON ENTRY : DI = DRIVE
                             <1> ;
                             <1> ; ON EXIT :  CY = 1 MEANS ERROR
                             <1> ;--------------------------------------------------------------------------------
                             <1> SETUP_DBL:
000023F0 8AA7[39710000]      <1>  MOV   AH, [DSK_STATE+eDI]     ; ACCESS STATE
000023F6 F6C410              <1>  TEST  AH,MED_DET        ; ESTABLISHED STATE ?
000023F9 757E                <1>  JNZ   short NO_DBL                  ; IF ESTABLISHED THEN DOUBLE DONE
                             <1>
                             <1> ;----- CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE
                             <1>
000023FB C605[29710000]00    <1>  MOV   byte [SEEK_STATUS],0    ; SET RECALIBRATE REQUIRED ON ALL DRIVES
00002402 E8E0000000          <1>  CALL  MOTOR_ON          ; ENSURE MOTOR STAY ON
00002407 B500                <1>  MOV   CH,0              ; LOAD TRACK 0
00002409 E8D4010000          <1>  CALL  SEEK              ; SEEK TO TRACK 0
0000240E E868000000          <1>  CALL  READ_ID                 ; READ ID FUNCTION
00002413 7249                <1>  JC    short SD_ERR            ; IF ERROR NO TRACK 0
                             <1>
                             <1> ;----- INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
                             <1>
00002415 66B95004            <1>  MOV   CX,0450H          ; START, MAX TRACKS
00002419 F687[39710000]01    <1>  TEST  byte [DSK_STATE+eDI],TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
00002420 7402                <1>  JZ    short CNT_OK            ; IF NOT COUNT IS SETUP
00002422 B1A0                <1>  MOV   CL,0A0H                 ; MAXIMUM TRACK 1.2 MB
                             <1>
                             <1> ;ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
                             <1> ;MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
                             <1> ;THEN SET DOUBLE STEP ON.
                             <1>
                             <1> CNT_OK:
00002424 C605[2B710000]FF    <1>         MOV     byte [MOTOR_COUNT], 0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION
0000242B 6651                <1>  PUSH  CX                ; SAVE TRACK, COUNT
0000242D C605[2C710000]00    <1>  MOV   byte [DSKETTE_STATUS],0 ; CLEAR STATUS, EXPECT ERRORS
```

```
00002434 6631C0               <1>  XOR   AX,AX             ; CLEAR AX
00002437 D0ED                 <1>  SHR   CH,1              ; HALVE TRACK, CY = HEAD
00002439 C0D003               <1>  RCL   AL,3              ; AX = HEAD IN CORRECT BIT
0000243C 6650                 <1>  PUSH  AX                ; SAVE HEAD
0000243E E89F010000           <1>  CALL  SEEK              ; SEEK TO TRACK
00002443 6658                 <1>  POP   AX                ; RESTORE HEAD
00002445 6609C7               <1>  OR    DI,AX             ; DI = HEAD OR'ED DRIVE
00002448 E82E000000           <1>  CALL  READ_ID                 ; READ ID HEAD 0
0000244D 9C                   <1>  PUSHF                   ; SAVE RETURN FROM READ_ID
0000244E 6681E7FB00           <1>  AND   DI,11111011B            ; TURN OFF HEAD 1 BIT
00002453 9D                   <1>  POPF                    ; RESTORE ERROR RETURN
00002454 6659                 <1>  POP   CX                ; RESTORE COUNT
00002456 7308                 <1>  JNC   short DO_CHK            ; IF OK, ASKED = RETURNED TRACK ?
00002458 FEC5                 <1>  INC   CH                ; INC FOR NEXT TRACK
0000245A 38CD                 <1>  CMP   CH,CL             ; REACHED MAXIMUM YET
0000245C 75C6                 <1>  JNZ   short CNT_OK            ; CONTINUE TILL ALL TRIED
                              <1>
                              <1> ;----- FALL THRU, READ ID FAILED FOR ALL TRACKS
                              <1>
                              <1> SD_ERR:
0000245E F9                   <1>  STC                     ; SET CARRY FOR ERROR
0000245F C3                   <1>  RETn                    ; SETUP_DBL ERROR EXIT
                              <1>
                              <1> DO_CHK:
00002460 8A0D[30710000]       <1>  MOV   CL, [NEC_STATUS+3]      ; LOAD RETURNED TRACK
00002466 888F[3D710000]       <1>  MOV   [DSK_TRK+eDI], CL ; STORE TRACK NUMBER
0000246C D0ED                 <1>  SHR   CH,1              ; HALVE TRACK
0000246E 38CD                 <1>  CMP   CH,CL             ; IS IT THE SAME AS ASKED FOR TRACK
00002470 7407                 <1>  JZ    short NO_DBL            ; IF SAME THEN NO DOUBLE STEP
00002472 808F[39710000]20     <1>  OR    byte [DSK_STATE+eDI],DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
                              <1> NO_DBL:
00002479 F8                   <1>  CLC                     ; CLEAR ERROR FLAG
0000247A C3                   <1>  RETn
                              <1>
                              <1> ;-------------------------------------------------------------------------------
                              <1> ; READ_ID
                              <1> ;READ ID FUNCTION.
                              <1> ;
                              <1> ; ON ENTRY:  DI : BIT 2 = HEAD; BITS 1,0 = DRIVE
                              <1> ;
                              <1> ; ON EXIT:   DI : BIT 2 IS RESET, BITS 1,0 = DRIVE
                              <1> ;      @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
                              <1> ;-------------------------------------------------------------------------------
                              <1> READ_ID:
0000247B B8[98240000]         <1>  MOV   eAX, ER_3         ; MOVE NEC OUTPUT ERROR ADDRESS
00002480 50                   <1>  PUSH  eAX
00002481 B44A                 <1>  MOV   AH,4AH                  ; READ ID COMMAND
00002483 E820010000           <1>  CALL  NEC_OUTPUT        ; TO CONTROLLER
00002488 6689F8               <1>  MOV   AX,DI             ; DRIVE # TO AH, HEAD 0
0000248B 88C4                 <1>  MOV   AH,AL
0000248D E816010000           <1>  CALL  NEC_OUTPUT        ; TO CONTROLLER
00002492 E80BFEFFFF           <1>  CALL  NEC_TERM          ; WAIT FOR OPERATION, GET STATUS
00002497 58                   <1>  POP   eAX               ; THROW AWAY ERROR ADDRESS
                              <1> ER_3:
00002498 C3                   <1>  RETn
                              <1>
                              <1> ;-------------------------------------------------------------------------------
                              <1> ; CMOS_TYPE
                              <1> ;RETURNS DISKETTE TYPE FROM CMOS
                              <1> ;
                              <1> ; ON ENTRY:  DI = DRIVE #
                              <1> ;
                              <1> ; ON EXIT:   AL = TYPE; CY REFLECTS STATUS
                              <1> ;-------------------------------------------------------------------------------
                              <1>
                              <1> CMOS_TYPE: ; 11/12/2014
00002499 8A87[D86D0000]       <1> mov    al, [eDI+fd0_type]
0000249F 20C0                 <1> and    al, al ; 18/12/2014
000024A1 C3                   <1> retn
                              <1>
                              <1> ;CMOS_TYPE:
                              <1> ;MOV   AL, CMOS_DIAG           ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
                              <1> ;CALL  CMOS_READ         ; GET CMOS STATUS
                              <1> ;TEST  AL,BAD_BAT+BAD_CKSUM    ; BATTERY GOOD AND CHECKSUM VALID
                              <1> ;STC                     ; SET CY = 1 INDICATING ERROR FOR RETURN
                              <1> ;JNZ   short BAD_CM            ; ERROR IF EITHER BIT ON
                              <1> ;MOV   AL,CMOS_DISKETTE  ; ADDRESS OF DISKETTE BYTE IN CMOS
                              <1> ;CALL  CMOS_READ         ; GET DISKETTE BYTE
                              <1> ;OR    DI,DI             ; SEE WHICH DRIVE IN QUESTION
                              <1> ;JNZ   short TB          ; IF DRIVE 1, DATA IN LOW NIBBLE
                              <1> ;ROR   AL,4              ; EXCHANGE NIBBLES IF SECOND DRIVE
                              <1> ;TB:
                              <1> ;AND   AL,0FH                  ; KEEP ONLY DRIVE DATA, RESET CY, 0
                              <1> ;BAD_CM:
                              <1> ;RETn                    ; CY, STATUS OF READ
                              <1>
                              <1> ;-------------------------------------------------------------------------------
                              <1> ; GET_PARM
                              <1> ;THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK_BASE
                              <1> ;BLOCK POINTED TO BY THE DATA VARIABLE @DISK_POINTER. A BYTE FROM
                              <1> ;THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING
                              <1> ;THE PARAMETER IN DL.
```

```
                             <1> ;
                             <1> ; ON ENTRY:  DL = INDEX OF BYTE TO BE FETCHED
                             <1> ;
                             <1> ; ON EXIT:   AH = THAT BYTE FROM BLOCK
                             <1> ;      AL,DH DESTROYED
                             <1> ;-----------------------------------------------------------------------------
                             <1> GET_PARM:
                             <1>  ;PUSH DS
000024A2 56                  <1>  PUSH  eSI
                             <1>       ;SUB  AX,AX            ; DS = 0, BIOS DATA AREA
                             <1>       ;MOV  DS,AX
                             <1>  ;;mov ax, cs
                             <1>  ;;mov ds, ax
                             <1>  ; 08/02/2015 (protected mode modifications, bx -> ebx)
000024A3 87D3                <1>  XCHG  eDX,eBX                ; BL = INDEX
                             <1>  ;SUB  BH,BH            ; BX = INDEX
000024A5 81E3FF000000        <1>  and   ebx, 0FFh
                             <1>       ;LDS  SI, [DISK_POINTER]     ; POINT TO BLOCK
                             <1>  ;
                             <1>  ; 17/12/2014
000024AB 66A1[516B0000]      <1>  mov   ax, [cfd] ; current (AL) and previous fd (AH)
000024B1 38E0                <1>  cmp   al, ah
000024B3 7425                <1>  je    short gpndc
000024B5 A2[526B0000]        <1>  mov   [pfd], al ; current drive -> previous drive
000024BA 53                  <1>  push  ebx ; 08/02/2015
000024BB 88C3                <1>  mov   bl, al
                             <1>  ; 11/12/2014
000024BD 8A83[D86D0000]      <1>  mov   al, [eBX+fd0_type]     ; Drive type (0,1,2,3,4)
                             <1>  ; 18/12/2014
000024C3 20C0                <1>  and   al, al
000024C5 7507                <1>  jnz   short gpdtc
000024C7 BB[3B6B0000]        <1>  mov   ebx, MD_TBL6           ; 1.44 MB param. tbl. (default)
000024CC EB05                <1>        jmp     short gpdpu
                             <1> gpdtc:
000024CE E818F9FFFF          <1>  call  DR_TYPE_CHECK
                             <1>  ; cf = 1 -> eBX points to 1.44MB fd parameter table (default)
                             <1> gpdpu:
000024D3 891D[D86A0000]      <1>  mov   [DISK_POINTER], ebx
000024D9 5B                  <1>  pop   ebx
                             <1> gpndc:
000024DA 8B35[D86A0000]      <1>  mov   esi, [DISK_POINTER] ; 08/02/2015, si -> esi
000024E0 8A241E              <1>  MOV   AH, [eSI+eBX]          ; GET THE WORD
000024E3 87D3                <1>  XCHG  eDX,eBX                ; RESTORE BX
000024E5 5E                  <1>  POP   eSI
                             <1>  ;POP  DS
000024E6 C3                  <1>  RETn
                             <1>
                             <1> ;-----------------------------------------------------------------------------
                             <1> ; MOTOR_ON
                             <1> ;TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR_COUNT
                             <1> ;IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE
                             <1> ;THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
                             <1> ;MOTOR NEEDED TO BE TURNED ON, THE MULTI-TASKING HOOK FUNCTION
                             <1> ;(AX=90FDH, INT 15) IS CALLED TELLING THE OPERATING SYSTEM
                             <1> ;THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS
                             <1> ;FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT
                             <1> ;HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE
                             <1> ;THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
                             <1> ;NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE
                             <1> ;PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
                             <1> ;IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
                             <1> ;WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ; ON EXIT:   AX,CX,DX DESTROYED
                             <1> ;-----------------------------------------------------------------------------
                             <1> MOTOR_ON:
000024E7 53                  <1>  PUSH  eBX               ; SAVE REG.
000024E8 E82A000000          <1>  CALL  TURN_ON                 ; TURN ON MOTOR
000024ED 7226                <1>  JC    short MOT_IS_ON         ; IF CY=1 NO WAIT
000024EF E89CF9FFFF          <1>  CALL  XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
000024F4 E866F9FFFF          <1>  CALL  XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH,
                             <1>  ;CALL TURN_ON           ; CHECK AGAIN IF MOTOR ON
                             <1>  ;JC   MOT_IS_ON         ; IF NO WAIT MEANS IT IS ON
                             <1> M_WAIT:
000024F9 B20A                <1>  MOV   DL,10             ; GET THE MOTOR WAIT PARAMETER
000024FB E8A2FFFFFF          <1>  CALL  GET_PARM
                             <1>  ;MOV  AL,AH             ; AL = MOTOR WAIT PARAMETER
                             <1>  ;XOR  AH,AH             ; AX = MOTOR WAIT PARAMETER
                             <1>  ;CMP  AL,8              ; SEE IF AT LEAST A SECOND IS SPECIFIED
00002500 80FC08              <1>  cmp   ah, 8
                             <1>  ;JAE  short GP2         ; IF YES, CONTINUE
00002503 7702                <1>  ja    short J13
                             <1>  ;MOV  AL,8              ; ONE SECOND WAIT FOR MOTOR START UP
00002505 B408                <1>  mov   ah, 8
                             <1>
                             <1> ;----- AS CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
                             <1> GP2:
                             <1> ;----- FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
                             <1> J13:                             ; WAIT FOR 1/8 SECOND PER (AL)
00002507 B95E200000          <1>  MOV   eCX,8286          ; COUNT FOR 1/8 SECOND AT 15.085737 US
0000250C E81BF1FFFF          <1>  CALL  WAITF             ; GO TO FIXED WAIT ROUTINE
```

```
                             <1>  ;DEC  AL                ; DECREMENT TIME VALUE
00002511 FECC                <1>  dec   ah
00002513 75F2                <1>  JNZ   short J13         ; ARE WE DONE YET
                             <1> MOT_IS_ON:
00002515 5B                  <1>  POP   eBX               ; RESTORE REG.
00002516 C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; TURN_ON
                             <1> ;TURN MOTOR ON AND RETURN WAIT STATE.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   CY = 0 MEANS WAIT REQUIRED
                             <1> ;      CY = 1 MEANS NO WAIT REQUIRED
                             <1> ;      AX,BX,CX,DX DESTROYED
                             <1> ;-------------------------------------------------------------------------------
                             <1> TURN_ON:
00002517 89FB                <1>  MOV   eBX,eDI                 ; BX = DRIVE #
00002519 88D9                <1>  MOV   CL,BL             ; CL = DRIVE #
0000251B C0C304              <1>  ROL   BL,4              ; BL = DRIVE SELECT
0000251E FA                  <1>  CLI                     ; NO INTERRUPTS WHILE DETERMINING STATUS
0000251F C605[2B710000]FF    <1>  MOV   byte [MOTOR_COUNT],0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION
00002526 A0[2A710000]        <1>  MOV   AL, [MOTOR_STATUS]      ; GET DIGITAL OUTPUT REGISTER REFLECTION
0000252B 2430                <1>  AND   AL,00110000B            ; KEEP ONLY DRIVE SELECT BITS
0000252D B401                <1>  MOV   AH,1              ; MASK FOR DETERMINING MOTOR BIT
0000252F D2E4                <1>  SHL   AH,CL             ; AH = MOTOR ON, A=00000001, B=00000010
                             <1>
                             <1> ;  AL = DRIVE SELECT FROM @MOTOR_STATUS
                             <1> ;  BL = DRIVE SELECT DESIRED
                             <1> ;  AH = MOTOR ON MASK DESIRED
                             <1>
00002531 38D8                <1>  CMP   AL,BL             ; REQUESTED DRIVE ALREADY SELECTED ?
00002533 7508                <1>  JNZ   short TURN_IT_ON  ; IF NOT SELECTED JUMP
00002535 8425[2A710000]      <1>  TEST  AH, [MOTOR_STATUS]      ; TEST MOTOR ON BIT
0000253B 7535                <1>  JNZ   short NO_MOT_WAIT ; JUMP IF MOTOR ON AND SELECTED
                             <1>
                             <1> TURN_IT_ON:
0000253D 08DC                <1>  OR    AH,BL             ; AH = DRIVE SELECT AND MOTOR ON
0000253F 8A3D[2A710000]      <1>  MOV   BH,[MOTOR_STATUS] ; SAVE COPY OF @MOTOR_STATUS BEFORE
00002545 80E70F              <1>  AND   BH,00001111B            ; KEEP ONLY MOTOR BITS
00002548 8025[2A710000]CF    <1>  AND   byte [MOTOR_STATUS],11001111B ; CLEAR OUT DRIVE SELECT
0000254F 0825[2A710000]      <1>  OR    [MOTOR_STATUS],AH ; OR IN DRIVE SELECTED AND MOTOR ON
00002555 A0[2A710000]        <1>  MOV   AL,[MOTOR_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION
0000255A 88C3                <1>  MOV   BL,AL             ; BL=@MOTOR_STATUS AFTER, BH=BEFORE
0000255C 80E30F              <1>  AND   BL,00001111B            ; KEEP ONLY MOTOR BITS
0000255F FB                  <1>  STI                     ; ENABLE INTERRUPTS AGAIN
00002560 243F                <1>  AND   AL,00111111B            ; STRIP AWAY UNWANTED BITS
00002562 C0C004              <1>  ROL   AL,4              ; PUT BITS IN DESIRED POSITIONS
00002565 0C0C                <1>  OR    AL,00001100B            ; NO RESET, ENABLE DMA/INTERRUPT
00002567 66BAF203            <1>  MOV   DX,03F2H          ; SELECT DRIVE AND TURN ON MOTOR
0000256B EE                  <1>  OUT   DX,AL
0000256C 38FB                <1>  CMP   BL,BH             ; NEW MOTOR TURNED ON ?
                             <1>  ;JZ   short NO_MOT_WAIT ; NO WAIT REQUIRED IF JUST SELECT
0000256E 7403                <1>  je    short no_mot_w1 ; 27/02/2015
00002570 F8                  <1>  CLC                     ; (re)SET CARRY MEANING WAIT
00002571 C3                  <1>  RETn
                             <1>
                             <1> NO_MOT_WAIT:
00002572 FB                  <1>  sti
                             <1> no_mot_w1: ; 27/02/2015
00002573 F9                  <1>  STC                     ; SET NO WAIT REQUIRED
                             <1>  ;STI                    ; INTERRUPTS BACK ON
00002574 C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; HD_WAIT
                             <1> ;WAIT FOR HEAD SETTLE TIME.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   AX,BX,CX,DX DESTROYED
                             <1> ;-------------------------------------------------------------------------------
                             <1> HD_WAIT:
00002575 B209                <1>  MOV   DL,9              ; GET HEAD SETTLE PARAMETER
00002577 E826FFFFFF          <1>  CALL  GET_PARM
0000257C 08E4                <1>  or    ah, ah      ; 17/12/2014
0000257E 7519                <1>  jnz   short DO_WAT
00002580 F605[2A710000]80    <1>         TEST    byte [MOTOR_STATUS],10000000B ; SEE IF A WRITE OPERATION
                             <1>  ;JZ   short ISNT_WRITE   ; IF NOT, DO NOT ENFORCE ANY VALUES
                             <1>  ;OR   AH,AH             ; CHECK FOR ANY WAIT?
                             <1>  ;JNZ  short DO_WAT            ; IF THERE DO NOT ENFORCE
00002587 741E                <1>  jz    short HW_DONE
00002589 B40F                <1>  MOV   AH,HD12_SETTLE          ; LOAD 1.2M HEAD SETTLE MINIMUM
0000258B 8A87[39710000]      <1>  MOV   AL,[DSK_STATE+eDI]      ; LOAD STATE
00002591 24C0                <1>  AND   AL,RATE_MSK       ; KEEP ONLY RATE
00002593 3C80                <1>  CMP   AL,RATE_250       ; 1.2 M DRIVE ?
00002595 7502                <1>  JNZ   short DO_WAT            ; DEFAULT HEAD SETTLE LOADED
                             <1> ;GP3:
00002597 B414                <1>  MOV   AH,HD320_SETTLE         ; USE 320/360 HEAD SETTLE
                             <1> ;JMP   SHORT DO_WAT
                             <1>
                             <1> ;ISNT_WRITE:
                             <1> ;OR    AH,AH             ; CHECK FOR NO WAIT
                             <1> ;JZ    short HW_DONE           ; IF NOT WRITE AND 0 ITS OK
                             <1>
                             <1> ;----- AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
                             <1> DO_WAT:
                             <1> ;MOV   AL,AH             ; AL = # MILLISECONDS
                             <1> ;;XOR  AH,AH             ; AX = # MILLISECONDS
                             <1> J29:                           ;     1 MILLISECOND LOOP
                             <1>  ;mov  cx, WAIT_FDU_HEAD_SETTLE ; 33 ; 1 ms in 30 micro units.
00002599 B942000000          <1>  MOV   eCX,66                  ; COUNT AT 15.085737 US PER COUNT
0000259E E889F0FFFF          <1>  CALL  WAITF             ; DELAY FOR 1 MILLISECOND
                             <1>  ;DEC  AL                ; DECREMENT THE COUNT
000025A3 FECC                <1>  dec   ah
000025A5 75F2                <1>  JNZ   short J29         ; DO AL MILLISECOND # OF TIMES
                             <1> HW_DONE:
000025A7 C3                  <1>  RETn
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; NEC_OUTPUT
                             <1> ;THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
                             <1> ;FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
                             <1> ;TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE AMOUNT
                             <1> ;OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
                             <1> ;
                             <1> ; ON ENTRY:  AH = BYTE TO BE OUTPUT
                             <1> ;
                             <1> ; ON EXIT:   CY = 0  SUCCESS
                             <1> ;       CY = 1  FAILURE -- DISKETTE STATUS UPDATED
                             <1> ;               IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
                             <1> ;               HIGHER THAN THE CALLER OF NEC OUTPUT. THIS REMOVES THE
                             <1> ;               REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC_OUTPUT.
                             <1> ;       AX,CX,DX DESTROYED
                             <1> ;-------------------------------------------------------------------------------
                             <1>
                             <1> ; 09/12/2014 [Erdogan Tan]
                             <1> ;(from 'PS2 Hardware Interface Tech. Ref. May 88', Page 09-05.)
                             <1> ; Diskette Drive Controller Status Register (3F4h)
                             <1> ;This read only register facilitates the transfer of data between
                             <1> ;the system microprocessor and the controller.
                             <1> ; Bit 7 - When set to 1, the Data register is ready to transfer data
                             <1> ;  with the system micrprocessor.
                             <1> ; Bit 6 - The direction of data transfer. If this bit is set to 0,
                             <1> ;  the transfer is to the controller.
                             <1> ; Bit 5 - When this bit is set to 1, the controller is in the non-DMA mode.
                             <1> ; Bit 4 - When this bit is set to 1, a Read or Write command is being executed.
                             <1> ; Bit 3 - Reserved.
                             <1> ; Bit 2 - Reserved.
                             <1> ; Bit 1 - When this bit is set to 1, dskette drive 1 is in the seek mode.
                             <1> ; Bit 0 - When this bit is set to 1, dskette drive 1 is in the seek mode.
                             <1>
                             <1> ; Data Register (3F5h)
                             <1> ; This read/write register passes data, commands and parameters, and provides
                             <1> ; diskette status information.
                             <1>
                             <1> NEC_OUTPUT:
                             <1>  ;PUSH BX                ; SAVE REG.
000025A8 66BAF403            <1>  MOV   DX,03F4H          ; STATUS PORT
                             <1>  ;MOV  BL,2              ; HIGH ORDER COUNTER
                             <1>  ;XOR  CX,CX             ; COUNT FOR TIME OUT
                             <1>  ; 16/12/2014
                             <1>  ; waiting for (max.) 0.5 seconds
                             <1>         ;;mov     byte [wait_count], 0 ;; 27/02/2015
                             <1>  ;
                             <1>  ; 17/12/2014
                             <1>  ; Modified from AWARD BIOS 1999 - ADISK.ASM - SEND_COMMAND
                             <1>  ;
                             <1>  ;WAIT_FOR_PORT:   Waits for a bit at a port pointed to by DX to
                             <1>  ;           go on.
                             <1>  ;INPUT:
                             <1>  ;     AH=Mask for isolation bits.
                             <1>  ;     AL=pattern to look for.
                             <1>  ;     DX=Port to test for
                             <1>  ;     BH:CX=Number of memory refresh periods to delay.
                             <1>  ;          (normally 30 microseconds per period.)
                             <1>  ;
                             <1>  ;WFP_SHORT:
                             <1>  ;     Wait for port if refresh cycle is short (15-80 Us range).
                             <1>  ;
                             <1>
                             <1> ;mov   bl, WAIT_FDU_SEND_HI+1  ; 0+1
                             <1> ;mov   cx, WAIT_FDU_SEND_LO    ; 16667
000025AC B91B410000          <1>  mov   ecx, WAIT_FDU_SEND_LH   ; 16667 (27/02/2015)
                             <1> ;
                             <1> ;WFPS_OUTER_LP:
                             <1> ;;
                             <1> ;WFPS_CHECK_PORT:
                             <1> J23:
000025B1 EC                  <1>  IN    AL,DX             ; GET STATUS
000025B2 24C0                <1>  AND   AL,11000000B            ; KEEP STATUS AND DIRECTION
000025B4 3C80                <1>  CMP   AL,10000000B            ; STATUS 1 AND DIRECTION 0 ?
000025B6 7418                <1>  JZ    short J27         ; STATUS AND DIRECTION OK
```

```
                             <1> WFPS_HI:
000025B8 E461                <1>  IN    AL, PORT_B ;061h ; SYS1      ; wait for hi to lo
000025BA A810                <1>  TEST  AL,010H                     ; transition on memory
000025BC 75FA                <1>  JNZ   SHORT WFPS_HI             ; refresh.
                             <1> WFPS_LO:
000025BE E461                <1>  IN    AL, PORT_B        ; SYS1
000025C0 A810                <1>  TEST  AL,010H
000025C2 74FA                <1>  JZ    SHORT WFPS_LO
                             <1>  ;LOOP SHORT WFPS_CHECK_PORT
000025C4 E2EB                <1>  loop  J23   ; 27/02/2015
                             <1> ;;
                             <1> ;dec   bl
                             <1> ;jnz   short WFPS_OUTER_LP
                             <1> ;jmp   short WFPS_TIMEOUT     ; fail
                             <1> ;J23:
                             <1> ;IN    AL,DX             ; GET STATUS
                             <1> ;AND   AL,11000000B           ; KEEP STATUS AND DIRECTION
                             <1> ;CMP   AL,10000000B           ; STATUS 1 AND DIRECTION 0 ?
                             <1> ;JZ    short J27         ; STATUS AND DIRECTION OK
                             <1>  ;LOOP J23               ; CONTINUE TILL CX EXHAUSTED
                             <1>  ;DEC  BL                ; DECREMENT COUNTER
                             <1>  ;JNZ  short J23         ; REPEAT TILL DELAY FINISHED, CX = 0
                             <1>
                             <1>  ;;27/02/2015
                             <1>  ;16/12/2014
                             <1>         ;;cmp     byte [wait_count], 10   ; (10/18.2 seconds)
                             <1>  ;;jb  short J23
                             <1>
                             <1> ;WFPS_TIMEOUT:
                             <1>
                             <1> ;----- FALL THRU TO ERROR RETURN
                             <1>
000025C6 800D[2C710000]80    <1>  OR    byte [DSKETTE_STATUS],TIME_OUT
                             <1>  ;POP BX                 ; RESTORE REG.
000025CD 58                  <1>  POP   eAX ; 08/02/2015  ; DISCARD THE RETURN ADDRESS
000025CE F9                  <1>  STC                     ; INDICATE ERROR TO CALLER
000025CF C3                  <1>  RETn
                             <1>
                             <1> ;----- DIRECTION AND STATUS OK; OUTPUT BYTE
                             <1>
                             <1> J27:
000025D0 88E0                <1>  MOV   AL,AH             ; GET BYTE TO OUTPUT
000025D2 6642                <1>  INC   DX                ; DATA PORT = STATUS PORT + 1
000025D4 EE                  <1>  OUT   DX,AL             ; OUTPUT THE BYTE
                             <1>  ;;NEWIODELAY  ;; 27/02/2015
                             <1>  ; 27/02/2015
000025D5 9C                  <1>  PUSHF                   ; SAVE FLAGS
000025D6 B903000000          <1>  MOV   eCX, 3                  ; 30 TO 45 MICROSECONDS WAIT FOR
000025DB E84CF0FFFF          <1>  CALL  WAITF             ; NEC FLAGS UPDATE CYCLE
000025E0 9D                  <1>  POPF                    ; RESTORE FLAGS FOR EXIT
                             <1>  ;POP  BX                ; RESTORE REG
000025E1 C3                  <1>  RETn                    ; CY = 0 FROM TEST INSTRUCTION
                             <1>
                             <1> ;--------------------------------------------------------------------------------
                             <1> ; SEEK
                             <1> ;THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE TO THE NAMED
                             <1> ;TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED SINCE THE DRIVE
                             <1> ;RESET COMMAND WAS ISSUED, THE DRIVE WILL BE RECALIBRATED.
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;      CH = TRACK #
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
                             <1> ;      AX,BX,CX DX DESTROYED
                             <1> ;--------------------------------------------------------------------------------
                             <1> SEEK:
000025E2 89FB                <1>  MOV   eBX,eDI                 ; BX = DRIVE #
000025E4 B001                <1>  MOV   AL,1              ; ESTABLISH MASK FOR RECALIBRATE TEST
000025E6 86CB                <1>  XCHG  CL,BL             ; SET DRIVE VALULE INTO CL
000025E8 D2C0                <1>  ROL   AL,CL             ; SHIFT MASK BY THE DRIVE VALUE
000025EA 86CB                <1>  XCHG  CL,BL             ; RECOVER TRACK VALUE
000025EC 8405[29710000]      <1>  TEST  AL,[SEEK_STATUS]  ; TEST FOR RECALIBRATE REQUIRED
000025F2 7526                <1>  JNZ   short J28A        ; JUMP IF RECALIBRATE NOT REQUIRED
                             <1>
000025F4 0805[29710000]      <1>  OR    [SEEK_STATUS],AL  ; TURN ON THE NO RECALIBRATE BIT IN FLAG
000025FA E862000000          <1>  CALL  RECAL             ; RECALIBRATE DRIVE
000025FF 730E                <1>  JNC   short AFT_RECAL         ; RECALIBRATE DONE
                             <1>
                             <1> ;----- ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
                             <1>
00002601 C605[2C710000]00    <1>  MOV   byte [DSKETTE_STATUS],0 ; CLEAR OUT INVALID STATUS
00002608 E854000000          <1>  CALL  RECAL             ; RECALIBRATE DRIVE
0000260D 7251                <1>  JC    short RB          ; IF RECALIBRATE FAILS TWICE THEN ERROR
                             <1>
                             <1> AFT_RECAL:
0000260F C687[3D710000]00    <1>         MOV     byte [DSK_TRK+eDI],0    ; SAVE NEW CYLINDER AS PRESENT POSITION
00002616 08ED                <1>  OR    CH,CH             ; CHECK FOR SEEK TO TRACK 0
00002618 743F                <1>  JZ    short DO_WAIT           ; HEAD SETTLE, CY = 0 IF JUMP
                             <1>
                             <1> ;----- DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
                             <1>
0000261A F687[39710000]20    <1> J28A:  TEST  byte [DSK_STATE+eDI],DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
```

```
00002621 7402                 <1>  JZ    short _R7         ; SINGLE STEP REQUIRED BYPASS DOUBLE
00002623 D0E5                 <1>  SHL   CH,1              ; DOUBLE NUMBER OF STEP TO TAKE
                              <1>
00002625 3AAF[3D710000]       <1> _R7:   CMP   CH, [DSK_TRK+eDI] ; SEE IF ALREADY AT THE DESIRED TRACK
0000262B 7433                 <1>  JE    short RB          ; IF YES, DO NOT NEED TO SEEK
                              <1>
0000262D BA[60260000]         <1>  MOV   eDX, NEC_ERR           ; LOAD RETURN ADDRESS
00002632 52                   <1>  PUSH  eDX ; (*)         ; ON STACK FOR NEC OUTPUT ERROR
00002633 88AF[3D710000]       <1>  MOV   [DSK_TRK+eDI],CH  ; SAVE NEW CYLINDER AS PRESENT POSITION
00002639 B40F                 <1>  MOV   AH,0FH                 ; SEEK COMMAND TO NEC
0000263B E868FFFFFF           <1>  CALL  NEC_OUTPUT
00002640 89FB                 <1>  MOV   eBX,eDI                ; BX = DRIVE #
00002642 88DC                 <1>  MOV   AH,BL             ; OUTPUT DRIVE NUMBER
00002644 E85FFFFFFF           <1>  CALL  NEC_OUTPUT
00002649 8AA7[3D710000]       <1>  MOV   AH, [DSK_TRK+eDI] ; GET CYLINDER NUMBER
0000264F E854FFFFFF           <1>  CALL  NEC_OUTPUT
00002654 E829000000           <1>  CALL  CHK_STAT_2        ; ENDING INTERRUPT AND SENSE STATUS
                              <1>
                              <1> ;----- WAIT FOR HEAD SETTLE
                              <1>
                              <1> DO_WAIT:
00002659 9C                   <1>  PUSHF                   ; SAVE STATUS
0000265A E816FFFFFF           <1>  CALL  HD_WAIT                ; WAIT FOR HEAD SETTLE TIME
0000265F 9D                   <1>  POPF                    ; RESTORE STATUS
                              <1> RB:
                              <1> NEC_ERR:
                              <1>  ; 08/02/2015 (code trick here from original IBM PC/AT DISKETTE.ASM)
                              <1>  ; (*) nec_err -> retn (push edx -> pop edx) -> nec_err -> retn
00002660 C3                   <1>  RETn                    ; RETURN TO CALLER
                              <1>
                              <1> ;-------------------------------------------------------------------------------
                              <1> ; RECAL
                              <1> ;RECALIBRATE DRIVE
                              <1> ;
                              <1> ; ON ENTRY:  DI = DRIVE #
                              <1> ;
                              <1> ; ON EXIT:   CY REFLECTS STATUS OF OPERATION.
                              <1> ;-------------------------------------------------------------------------------
                              <1> RECAL:
00002661 6651                 <1>  PUSH  CX
00002663 B8[7F260000]         <1>  MOV   eAX, RC_BACK           ; LOAD NEC_OUTPUT ERROR
00002668 50                   <1>  PUSH  eAX
00002669 B407                 <1>  MOV   AH,07H                 ; RECALIBRATE COMMAND
0000266B E838FFFFFF           <1>  CALL  NEC_OUTPUT
00002670 89FB                 <1>  MOV   eBX,eDI                ; BX = DRIVE #
00002672 88DC                 <1>  MOV   AH,BL
00002674 E82FFFFFFF           <1>  CALL  NEC_OUTPUT        ; OUTPUT THE DRIVE NUMBER
00002679 E804000000           <1>  CALL  CHK_STAT_2        ; GET THE INTERRUPT AND SENSE INT STATUS
0000267E 58                   <1>  POP   eAX               ; THROW AWAY ERROR
                              <1> RC_BACK:
0000267F 6659                 <1>  POP   CX
00002681 C3                   <1>  RETn
                              <1>
                              <1> ;-------------------------------------------------------------------------------
                              <1> ; CHK_STAT_2
                              <1> ;THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER RECALIBRATE,
                              <1> ;OR SEEK TO THE ADAPTER. THE INTERRUPT IS WAITED FOR, THE
                              <1> ;INTERRUPT STATUS SENSED, AND THE RESULT RETURNED TO THE CALLER.
                              <1> ;
                              <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
                              <1> ;-------------------------------------------------------------------------------
                              <1> CHK_STAT_2:
00002682 B8[AA260000]         <1>         MOV     eAX, CS_BACK            ; LOAD NEC_OUTPUT ERROR ADDRESS
00002687 50                   <1>  PUSH  eAX
00002688 E828000000           <1>  CALL  WAIT_INT          ; WAIT FOR THE INTERRUPT
0000268D 721A                 <1>  JC    short J34         ; IF ERROR, RETURN IT
0000268F B408                 <1>  MOV   AH,08H                 ; SENSE INTERRUPT STATUS COMMAND
00002691 E812FFFFFF           <1>  CALL  NEC_OUTPUT
00002696 E84A000000           <1>  CALL  RESULTS                ; READ IN THE RESULTS
0000269B 720C                 <1>  JC    short J34
0000269D A0[2D710000]         <1>  MOV   AL,[NEC_STATUS]        ; GET THE FIRST STATUS BYTE
000026A2 2460                 <1>  AND   AL,01100000B           ; ISOLATE THE BITS
000026A4 3C60                 <1>  CMP   AL,01100000B           ; TEST FOR CORRECT VALUE
000026A6 7403                 <1>  JZ    short J35         ; IF ERROR, GO MARK IT
000026A8 F8                   <1>  CLC                     ; GOOD RETURN
                              <1> J34:
000026A9 58                   <1>  POP   eAX               ; THROW AWAY ERROR RETURN
                              <1> CS_BACK:
000026AA C3                   <1>  RETn
                              <1> J35:
000026AB 800D[2C710000]40     <1>  OR    byte [DSKETTE_STATUS], BAD_SEEK
000026B2 F9                   <1>  STC                     ; ERROR RETURN CODE
000026B3 EBF4                 <1>  JMP   SHORT J34
                              <1>
                              <1> ;-------------------------------------------------------------------------------
                              <1> ; WAIT_INT
                              <1> ;THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT ROUTINE
                              <1> ;TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE RETURNED
                              <1> ;IF THE DRIVE IS NOT READY.
                              <1> ;
                              <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
                              <1> ;-------------------------------------------------------------------------------
```

```
                             <1>
                             <1> ; 17/12/2014
                             <1> ; 2.5 seconds waiting !
                             <1> ;(AWARD BIOS - 1999, WAIT_FDU_INT_LOW, WAIT_FDU_INT_HI)
                             <1> ; amount of time to wait for completion interrupt from NEC.
                             <1>
                             <1>
                             <1> WAIT_INT:
000026B5 FB                  <1>  STI                     ; TURN ON INTERRUPTS, JUST IN CASE
000026B6 F8                  <1>  CLC                     ; CLEAR TIMEOUT INDICATOR
                             <1>         ;MOV  BL,10            ; CLEAR THE COUNTERS
                             <1>         ;XOR  CX,CX            ; FOR 2 SECOND WAIT
                             <1>
                             <1>  ; Modification from AWARD BIOS - 1999 (ATORGS.ASM, WAIT
                             <1>  ;
                             <1>  ;WAIT_FOR_MEM:
                             <1>  ;     Waits for a bit at a specified memory location pointed
                             <1>  ;     to by ES:[DI] to become set.
                             <1>  ;INPUT:
                             <1>  ;     AH=Mask to test with.
                             <1>  ;     ES:[DI] = memory location to watch.
                             <1>  ;     BH:CX=Number of memory refresh periods to delay.
                             <1>  ;          (normally 30 microseconds per period.)
                             <1>
                             <1>  ; waiting for (max.) 2.5 secs in 30 micro units.
                             <1> ;mov   cx, WAIT_FDU_INT_LO          ; 017798
                             <1> ;;      mov   bl, WAIT_FDU_INT_HI
                             <1> ;mov   bl, WAIT_FDU_INT_HI + 1
                             <1>  ; 27/02/2015
000026B7 B986450100          <1>  mov   ecx, WAIT_FDU_INT_LH    ; 83334 (2.5 seconds)
                             <1> WFMS_CHECK_MEM:
000026BC F605[29710000]80    <1>  test  byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
000026C3 7516                <1>         jnz     short J37
                             <1> WFMS_HI:
000026C5 E461                <1>  IN    AL,PORT_B  ; 061h ; SYS1, wait for lo to hi
000026C7 A810                <1>  TEST  AL,010H                 ; transition on memory
000026C9 75FA                <1>  JNZ   SHORT WFMS_HI           ; refresh.
                             <1> WFMS_LO:
000026CB E461                <1>  IN    AL,PORT_B         ;SYS1
000026CD A810                <1>  TEST  AL,010H
000026CF 74FA                <1>  JZ    SHORT WFMS_LO
000026D1 E2E9                <1>         LOOP    WFMS_CHECK_MEM
                             <1> ;WFMS_OUTER_LP:
                             <1> ;;      or    bl, bl                  ; check outer counter
                             <1> ;;      jz    short J36A        ; WFMS_TIMEOUT
                             <1> ;dec   bl
                             <1> ;jz    short J36A
                             <1> ;jmp   short WFMS_CHECK_MEM
                             <1>
                             <1>  ;17/12/2014
                             <1>  ;16/12/2014
                             <1> ;        mov     byte [wait_count], 0    ; Reset (INT 08H) counter
                             <1> ;J36:
                             <1> ;TEST  byte [SEEK_STATUS],INT_FLAG ; TEST FOR INTERRUPT OCCURRING
                             <1> ;JNZ   short J37
                             <1>  ;16/12/2014
                             <1>  ;LOOP J36               ; COUNT DOWN WHILE WAITING
                             <1>  ;DEC  BL                ; SECOND LEVEL COUNTER
                             <1>  ;JNZ  short J36
                             <1> ;       cmp     byte [wait_count], 46   ; (46/18.2 seconds)
                             <1> ;jb    short J36
                             <1>
                             <1> ;WFMS_TIMEOUT:
                             <1> ;J36A:
000026D3 800D[2C710000]80    <1>  OR    byte [DSKETTE_STATUS], TIME_OUT ; NOTHING HAPPENED
000026DA F9                  <1>  STC                     ; ERROR RETURN
                             <1> J37:
000026DB 9C                  <1>  PUSHF                   ; SAVE CURRENT CARRY
000026DC 8025[29710000]7F    <1>  AND   byte [SEEK_STATUS], ~INT_FLAG ; TURN OFF INTERRUPT FLAG
000026E3 9D                  <1>  POPF                    ; RECOVER CARRY
000026E4 C3                  <1>  RETn                    ; GOOD RETURN CODE
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; RESULTS
                             <1> ;THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER RETURNS
                             <1> ;FOLLOWING AN INTERRUPT.
                             <1> ;
                             <1> ; ON EXIT:   @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
                             <1> ;      AX,BX,CX,DX DESTROYED
                             <1> ;-------------------------------------------------------------------------------
                             <1> RESULTS:
000026E5 57                  <1>  PUSH  eDI
000026E6 BF[2D710000]        <1>  MOV   eDI, NEC_STATUS          ; POINTER TO DATA AREA
000026EB B307                <1>  MOV   BL,7               ; MAX STATUS BYTES
000026ED 66BAF403            <1>  MOV   DX,03F4H           ; STATUS PORT
                             <1>
                             <1> ;----- WAIT FOR REQUEST FOR MASTER
                             <1>
                             <1> _R10:
                             <1>  ; 16/12/2014
                             <1>  ; wait for (max) 0.5 seconds
                             <1>  ;MOV  BH,2              ; HIGH ORDER COUNTER
```

```
                              <1>  ;XOR  CX,CX             ; COUNTER
                              <1>
                              <1>  ;Time to wait while waiting for each byte of NEC results = .5
                              <1>  ;seconds.  .5 seconds = 500,000 micros.  500,000/30 = 16,667.
                              <1>  ; 27/02/2015
000026F1 B91B410000           <1>  mov   ecx, WAIT_FDU_RESULTS_LH ; 16667
                              <1>  ;mov  cx, WAIT_FDU_RESULTS_LO  ; 16667
                              <1>  ;mov  bh, WAIT_FDU_RESULTS_HI+1 ; 0+1
                              <1>
                              <1> WFPSR_OUTER_LP:
                              <1>  ;
                              <1> WFPSR_CHECK_PORT:
                              <1> J39:                           ; WAIT FOR MASTER
000026F6 EC                   <1>  IN    AL,DX             ; GET STATUS
000026F7 24C0                 <1>  AND   AL,11000000B            ; KEEP ONLY STATUS AND DIRECTION
000026F9 3CC0                 <1>  CMP   AL,11000000B            ; STATUS 1 AND DIRECTION 1 ?
000026FB 7418                 <1>  JZ    short J42         ; STATUS AND DIRECTION OK
                              <1> WFPSR_HI:
000026FD E461                 <1>  IN    AL, PORT_B  ;061h ; SYS1      ; wait for hi to lo
000026FF A810                 <1>  TEST  AL,010H                 ; transition on memory
00002701 75FA                 <1>  JNZ   SHORT WFPSR_HI          ; refresh.
                              <1> WFPSR_LO:
00002703 E461                 <1>  IN    AL, PORT_B        ; SYS1
00002705 A810                 <1>  TEST  AL,010H
00002707 74FA                 <1>  JZ    SHORT WFPSR_LO
00002709 E2EB                 <1>         LOOP    WFPSR_CHECK_PORT
                              <1>  ;; 27/02/2015
                              <1>  ;;dec bh
                              <1>  ;;jnz short WFPSR_OUTER_LP
                              <1>  ;jmp  short WFPSR_TIMEOUT     ; fail
                              <1>
                              <1>  ;;mov byte [wait_count], 0
                              <1> ;J39:                          ; WAIT FOR MASTER
                              <1> ;IN    AL,DX             ; GET STATUS
                              <1> ;AND   AL,11000000B            ; KEEP ONLY STATUS AND DIRECTION
                              <1> ;CMP   AL,11000000B            ; STATUS 1 AND DIRECTION 1 ?
                              <1> ;JZ    short J42         ; STATUS AND DIRECTION OK
                              <1>  ;LOOP J39               ; LOOP TILL TIMEOUT
                              <1>  ;DEC  BH                ; DECREMENT HIGH ORDER COUNTER
                              <1>  ;JNZ  short J39         ; REPEAT TILL DELAY DONE
                              <1>  ;
                              <1>  ;;cmp byte [wait_count], 10  ; (10/18.2 seconds)
                              <1>  ;;jb  short J39
                              <1>
                              <1> ;WFPSR_TIMEOUT:
0000270B 800D[2C710000]80     <1>  OR    byte [DSKETTE_STATUS],TIME_OUT
00002712 F9                   <1>  STC                     ; SET ERROR RETURN
00002713 EB29                 <1>  JMP   SHORT POPRES            ; POP REGISTERS AND RETURN
                              <1>
                              <1> ;----- READ IN THE STATUS
                              <1>
                              <1> J42:
00002715 EB00                 <1>  JMP   $+2               ; I/O DELAY
00002717 6642                 <1>  INC   DX                ; POINT AT DATA PORT
00002719 EC                   <1>  IN    AL,DX             ; GET THE DATA
                              <1>  ; 16/12/2014
                              <1>  NEWIODELAY
0000271A E6EB                 <2>  out 0ebh,al
0000271C 8807                 <1>         MOV     [eDI],AL                ; STORE THE BYTE
0000271E 47                   <1>  INC   eDI               ; INCREMENT THE POINTER
                              <1>  ; 16/12/2014
                              <1> ;push  cx
                              <1> ;mov   cx, 30
                              <1> ;wdw2:
                              <1> ;NEWIODELAY
                              <1> ;loop  wdw2
                              <1> ;pop   cx
                              <1>
0000271F B903000000           <1>  MOV   eCX,3             ; MINIMUM 24 MICROSECONDS FOR NEC
00002724 E803EFFFFF           <1>  CALL  WAITF             ; WAIT 30 TO 45 MICROSECONDS
00002729 664A                 <1>  DEC   DX                ; POINT AT STATUS PORT
0000272B EC                   <1>  IN    AL,DX             ; GET STATUS
                              <1>  ; 16/12/2014
                              <1>  NEWIODELAY
0000272C E6EB                 <2>  out 0ebh,al
                              <1>  ;
0000272E A810                 <1>  TEST  AL,00010000B            ; TEST FOR NEC STILL BUSY
00002730 740C                 <1>  JZ    short POPRES            ; RESULTS DONE ?
                              <1>
00002732 FECB                 <1>  DEC   BL                ; DECREMENT THE STATUS COUNTER
00002734 75BB                 <1>         JNZ     short _R10              ; GO BACK FOR MORE
00002736 800D[2C710000]20     <1>  OR    byte [DSKETTE_STATUS],BAD_NEC ; TOO MANY STATUS BYTES
0000273D F9                   <1>  STC                     ; SET ERROR FLAG
                              <1>
                              <1> ;----- RESULT OPERATION IS DONE
                              <1> POPRES:
0000273E 5F                   <1>  POP   eDI
0000273F C3                   <1>  RETn                    ; RETURN WITH CARRY SET
                              <1>
                              <1> ;-------------------------------------------------------------------------------
                              <1> ; READ_DSKCHNG
                              <1> ;READS THE STATE OF THE DISK CHANGE LINE.
```

```
                             <1> ;
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;
                             <1> ; ON EXIT:   DI = DRIVE #
                             <1> ;       ZF = 0 : DISK CHANGE LINE INACTIVE
                             <1> ;       ZF = 1 : DISK CHANGE LINE ACTIVE
                             <1> ;       AX,CX,DX DESTROYED
                             <1> ;-------------------------------------------------------------------------------
                             <1> READ_DSKCHNG:
00002740 E8A2FDFFFF          <1>  CALL  MOTOR_ON          ; TURN ON THE MOTOR IF OFF
00002745 66BAF703            <1>  MOV   DX,03F7H          ; ADDRESS DIGITAL INPUT REGISTER
00002749 EC                  <1>  IN    AL,DX             ; INPUT DIGITAL INPUT REGISTER
0000274A A880                <1>  TEST  AL,DSK_CHG        ; CHECK FOR DISK CHANGE LINE ACTIVE
0000274C C3                  <1>  RETn                    ; RETURN TO CALLER WITH ZERO FLAG SET
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DRIVE_DET
                             <1> ;DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND
                             <1> ;UPDATES STATE INFORMATION ACCORDINGLY.
                             <1> ; ON ENTRY:  DI = DRIVE #
                             <1> ;-------------------------------------------------------------------------------
                             <1> DRIVE_DET:
0000274D E895FDFFFF          <1>  CALL  MOTOR_ON          ; TURN ON MOTOR IF NOT ALREADY ON
00002752 E80AFFFFFF          <1>  CALL  RECAL             ; RECALIBRATE DRIVE
00002757 7251                <1>  JC    short DD_BAC            ; ASSUME NO DRIVE PRESENT
00002759 B530                <1>  MOV   CH,TRK_SLAP       ; SEEK TO TRACK 48
0000275B E882FEFFFF          <1>  CALL  SEEK
00002760 7248                <1>  JC    short DD_BAC            ; ERROR NO DRIVE
00002762 B50B                <1>  MOV   CH,QUIET_SEEK+1         ; SEEK TO TRACK 10
                             <1> SK_GIN:
00002764 FECD                <1>  DEC   CH                ; DECREMENT TO NEXT TRACK
00002766 6651                <1>  PUSH  CX                ; SAVE TRACK
00002768 E875FEFFFF          <1>  CALL  SEEK
0000276D 723C                <1>  JC    short POP_BAC           ; POP AND RETURN
0000276F B8[AB270000]        <1>  MOV   eAX, POP_BAC            ; LOAD NEC OUTPUT ERROR ADDRESS
00002774 50                  <1>  PUSH  eAX
00002775 B404                <1>  MOV   AH,SENSE_DRV_ST         ; SENSE DRIVE STATUS COMMAND BYTE
00002777 E82CFEFFFF          <1>  CALL  NEC_OUTPUT        ; OUTPUT TO NEC
0000277C 6689F8              <1>  MOV   AX,DI             ; AL = DRIVE
0000277F 88C4                <1>  MOV   AH,AL             ; AH = DRIVE
00002781 E822FEFFFF          <1>  CALL  NEC_OUTPUT        ; OUTPUT TO NEC
00002786 E85AFFFFFF          <1>  CALL  RESULTS                 ; GO GET STATUS
0000278B 58                  <1>  POP   eAX               ; THROW AWAY ERROR ADDRESS
0000278C 6659                <1>  POP   CX                ; RESTORE TRACK
0000278E F605[2D710000]10    <1>  TEST  byte [NEC_STATUS], HOME ; TRACK 0 ?
00002795 74CD                <1>  JZ    short SK_GIN            ; GO TILL TRACK 0
00002797 08ED                <1>  OR    CH,CH             ; IS HOME AT TRACK 0
00002799 7408                <1>  JZ    short IS_80       ; MUST BE 80 TRACK DRIVE
                             <1>
                             <1> ;DRIVE IS A 360; SET DRIVE TO DETERMINED;
                             <1> ;SET MEDIA TO DETERMINED AT RATE 250.
                             <1>
0000279B 808F[39710000]94    <1>  OR    byte [DSK_STATE+eDI], DRV_DET+MED_DET+RATE_250
000027A2 C3                  <1>  RETn                    ; ALL INFORMATION SET
                             <1> IS_80:
000027A3 808F[39710000]01    <1>  OR    byte [DSK_STATE+eDI], TRK_CAPA ; SETUP 80 TRACK CAPABILITY
                             <1> DD_BAC:
000027AA C3                  <1>  RETn
                             <1> POP_BAC:
000027AB 6659                <1>  POP   CX                ; THROW AWAY
000027AD C3                  <1>  RETn
                             <1>
                             <1> fdc_int:
                             <1>    ; 30/07/2015
                             <1>    ; 16/02/2015
                             <1> ;int_0Eh: ; 11/12/2014
                             <1>
                             <1> ;--- HARDWARE INT 0EH -- ( IRQ LEVEL  6 ) -------------------------------------
                             <1> ; DISK_INT
                             <1> ;THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
                             <1> ;
                             <1> ; ON EXIT:   THE INTERRUPT FLAG IS SET IN @SEEK_STATUS.
                             <1> ;-------------------------------------------------------------------------------
                             <1> DISK_INT_1:
                             <1>
000027AE 6650                <1>  PUSH  AX                ; SAVE WORK REGISTER
000027B0 1E                  <1>  push  ds
000027B1 66B81000            <1>  mov   ax, KDATA
000027B5 8ED8                <1>  mov   ds, ax
000027B7 800D[29710000]80    <1>         OR      byte [SEEK_STATUS], INT_FLAG ; TURN ON INTERRUPT OCCURRED
000027BE B020                <1>  MOV     AL,EOI                   ; END OF INTERRUPT MARKER
000027C0 E620                <1>  OUT   INTA00,AL         ; INTERRUPT CONTROL PORT
000027C2 1F                  <1>  pop   ds
000027C3 6658                <1>  POP   AX                ; RECOVER REGISTER
000027C5 CF                  <1>  IRET                    ; RETURN FROM INTERRUPT
                             <1>
                             <1> ;-------------------------------------------------------------------------------
                             <1> ; DSKETTE_SETUP
                             <1> ;THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE OF
                             <1> ;DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.
                             <1> ;-------------------------------------------------------------------------------
                             <1> DSKETTE_SETUP:
```

```
                             <1>  ;PUSH AX                ; SAVE REGISTERS
                             <1>  ;PUSH BX
                             <1>  ;PUSH CX
000027C6 52                  <1>  PUSH  eDX
                             <1>  ;PUSH DI
                             <1>  ;;PUSH      DS
                             <1>  ; 14/12/2014
                             <1>  ;mov  word [DISK_POINTER], MD_TBL6
                             <1>  ;mov  [DISK_POINTER+2], cs
                             <1>  ;
                             <1>  ;OR   byte [RTC_WAIT_FLAG], 1 ; NO RTC WAIT, FORCE USE OF LOOP
000027C7 31FF                <1>  XOR   eDI,eDI                 ; INITIALIZE DRIVE POINTER
000027C9 66C705[39710000]00- <1>  MOV   WORD [DSK_STATE],0      ; INITIALIZE STATES
000027D1 00                  <1>
000027D2 8025[34710000]33    <1>  AND   byte [LASTRATE],~(STRT_MSK+SEND_MSK) ; CLEAR START & SEND
000027D9 800D[34710000]C0    <1>  OR    byte [LASTRATE],SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
000027E0 C605[29710000]00    <1>  MOV   byte [SEEK_STATUS],0    ; INDICATE RECALIBRATE NEEDED
000027E7 C605[2B710000]00    <1>  MOV   byte [MOTOR_COUNT],0    ; INITIALIZE MOTOR COUNT
000027EE C605[2A710000]00    <1>  MOV   byte [MOTOR_STATUS],0   ; INITIALIZE DRIVES TO OFF STATE
000027F5 C605[2C710000]00    <1>  MOV   byte [DSKETTE_STATUS],0 ; NO ERRORS
                             <1>  ;
                             <1>  ; 28/02/2015
                             <1>  ;mov  word [cfd], 100h
000027FC E860F2FFFF          <1>  call  DSK_RESET
00002801 5A                  <1>  pop   edx
00002802 C3                  <1>  retn
                             <1>
                             <1> ;SUP0:
                             <1> ;CALL  DRIVE_DET         ; DETERMINE DRIVE
                             <1> ;CALL  XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
                             <1> ;; 02/01/2015
                             <1> ;;INC  DI                ; POINT TO NEXT DRIVE
                             <1> ;;CMP  DI,MAX_DRV        ; SEE IF DONE
                             <1> ;;JNZ  short SUP0        ; REPEAT FOR EACH ORIVE
                             <1> ;       cmp     byte [fd1_type], 0
                             <1> ;jna   short sup1
                             <1> ;or    di, di
                             <1> ;jnz   short sup1
                             <1> ;inc   di
                             <1> ;       jmp     short SUP0
                             <1> ;sup1:
                             <1> ;MOV   byte [SEEK_STATUS],0    ; FORCE RECALIBRATE
                             <1> ;;AND  byte [RTC_WAIT_FLAG],0FEH ; ALLOW FOR RTC WAIT
                             <1> ;CALL  SETUP_END         ; VARIOUS CLEANUPS
                             <1> ;;;POP DS                ; RESTORE CALLERS REGISTERS
                             <1> ;;POP  DI
                             <1> ;POP   eDX
                             <1> ;;POP  CX
                             <1> ;;POP  BX
                             <1> ;;POP  AX
                             <1> ;RETn
                             <1>
                             <1> ;/////////////////////////////////////////////////////
                             <1> ;; END OF DISKETTE I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             <1> ;
                             <1>
                             <1> int13h: ; 21/02/2015
00002803 9C                  <1>  pushfd
00002804 0E                  <1>  push  cs
00002805 E859000000          <1>  call  DISK_IO
0000280A C3                  <1>  retn
                             <1>
                             <1> ;;;;;; DISK I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; 21/02/2015 ;;;
                             <1> ;////////////////////////////////////////////////////////////////////
                             <1>
                             <1> ; DISK I/O - Erdogan Tan (Retro UNIX 386 v1 project)
                             <1> ; 23/02/2015
                             <1> ; 21/02/2015 (unix386.s)
                             <1> ; 22/12/2014 - 14/02/2015 (dsectrm2.s)
                             <1> ;
                             <1> ; Original Source Code:
                             <1> ; DISK ----- 09/25/85 FIXED DISK BIOS
                             <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
                             <1> ;
                             <1> ; Modifications: by reference of AWARD BIOS 1999 (D1A0622)
                             <1> ;       Source Code - ATORGS.ASM, AHDSK.ASM
                             <1> ;
                             <1>
                             <1>
                             <1> ;The wait for controller to be not busy is 10 seconds.
                             <1> ;10,000,000 / 30 = 333,333.  333,333 decimal = 051615h
                             <1> ;;WAIT_HDU_CTLR_BUSY_LO  equ   1615h
                             <1> ;;WAIT_HDU_CTLR_BUSY_HI  equ     05h
                             <1> WAIT_HDU_CTRL_BUSY_LH    equ   51615h       ;21/02/2015
                             <1>
                             <1> ;The wait for controller to issue completion interrupt is 10 seconds.
                             <1> ;10,000,000 / 30 = 333,333.  333,333 decimal = 051615h
                             <1> ;;WAIT_HDU_INT_LO  equ   1615h
                             <1> ;;WAIT_HDU_INT_HI  equ     05h
                             <1> WAIT_HDU_INT_LH          equ   51615h      ; 21/02/2015
                             <1>
                             <1> ;The wait for Data request on read and write longs is
```

```
<1> ;2000 us. (?)
<1> ;;WAIT_HDU_DRQ_LO  equ   1000  ; 03E8h
<1> ;;WAIT_HDU_DRQ_HI  equ   0
<1> WAIT_HDU_DRQ_LH          equ   1000  ; 21/02/2015
<1>
<1> ; Port 61h (PORT_B)
<1> SYS1         equ   61h   ; PORT_B  (diskette.inc)
<1>
<1> ; 23/12/2014
<1> %define CMD_BLOCK       eBP-8  ; 21/02/2015
<1>
<1>
<1> ;--- INT 13H ------------------------------------------------------------------
<1> ;                                                      :
<1> ; FIXED DISK I/O INTERFACE                                         :
<1> ;                                                      :
<1> ;THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH           :
<1> ;THE IBM FIXED DISK CONTROLLER.                                    :
<1> ;                                                      :
<1> ;THE  BIOS  ROUTINES  ARE  MEANT  TO  BE  ACCESSED  THROUGH        :
<1> ;SOFTWARE  INTERRUPTS  ONLY.    ANY  ADDRESSES  PRESENT     IN           :
<1> ;THESE  LISTINGS  ARE  INCLUDED      ONLY FOR  COMPLETENESS,             :
<1> ;NOT  FOR  REFERENCE.  APPLICATIONS   WHICH  REFERENCE  ANY        :
<1> ;ABSOLUTE  ADDRESSES  WITHIN  THE  CODE   SEGMENTS  OF  BIOS             :
<1> ;VIOLATE  THE  STRUCTURE  AND  DESIGN  OF  BIOS.                   :
<1> ;                                                      :
<1> ;-----------------------------------------------------------------------------:
<1> ;                                                      :
<1> ; INPUT  (AH)= HEX COMMAND VALUE                                   :
<1> ;                                                      :
<1> ;(AH)= 00H  RESET DISK (DL = 80H,81H) / DISKETTE                   :
<1> ;(AH)= 01H  READ THE STATUS OF THE LAST DISK OPERATION INTO (AL)        :
<1> ;          NOTE: DL < 80H - DISKETTE                         :
<1> ;              DL > 80H - DISK                         :
<1> ;(AH)= 02H  READ THE DESIRED SECTORS INTO MEMORY                   :
<1> ;(AH)= 03H  WRITE THE DESIRED SECTORS FROM MEMORY                  :
<1> ;(AH)= 04H  VERIFY THE DESIRED SECTORS                             :
<1> ;(AH)= 05H  FORMAT THE DESIRED TRACK                         :
<1> ;(AH)= 06H  UNUSED                                     :
<1> ;(AH)= 07H  UNUSED                                     :
<1> ;(AH)= 08H  RETURN THE CURRENT DRIVE PARAMETERS                    :
<1> ;(AH)= 09H  INITIALIZE DRIVE PAIR CHARACTERISTICS                  :
<1> ;          INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0           :
<1> ;          INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1           :
<1> ;(AH)= 0AH  READ LONG                                        :
<1> ;(AH)= 0BH  WRITE LONG  (READ & WRITE LONG ENCOMPASS 512 + 4 BYTES ECC) :
<1> ;(AH)= 0CH  SEEK                                       :
<1> ;(AH)= 0DH  ALTERNATE DISK RESET (SEE DL)                    :
<1> ;(AH)= 0EH  UNUSED                                     :
<1> ;(AH)= 0FH  UNUSED                                     :
<1> ;(AH)= 10H  TEST DRIVE READY                                 :
<1> ;(AH)= 11H  RECALIBRATE                                      :
<1> ;(AH)= 12H  UNUSED                                     :
<1> ;(AH)= 13H  UNUSED                                     :
<1> ;(AH)= 14H  CONTROLLER INTERNAL DIAGNOSTIC                   :
<1> ;(AH)= 15H  READ DASD TYPE                                   :
<1> ;                                                      :
<1> ;------------------------------------------------------------------------------
<1> ;                                                      :
<1> ;REGISTERS USED FOR FIXED DISK OPERATIONS                    :
<1> ;                                                      :
<1> ;      (DL)  -  DRIVE NUMBER     (80H-81H FOR DISK. VALUE CHECKED)  :
<1> ;      (DH)  -  HEAD NUMBER        (0-15 ALLOWED, NOT VALUE CHECKED)  :
<1> ;      (CH)  -  CYLINDER NUMBER  (0-1023, NOT VALUE CHECKED)(SEE CL):
<1> ;      (CL)  -  SECTOR NUMBER    (1-17, NOT VALUE CHECKED)         :
<1> ;                                                      :
<1> ;               NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED     :
<1> ;                   IN THE HIGH 2 BITS OF THE CL REGISTER          :
<1> ;                   (10 BITS TOTAL)                    :
<1> ;                                                      :
<1> ;      (AL)  -  NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H,    :
<1> ;                              FOR READ/WRITE LONG 1-79H)       :
<1> ;                                                      :
<1> ;      (ES:BX) -  ADDRESS OF BUFFER FOR READS AND WRITES,          :
<1> ;               (NOT REQUIRED FOR VERIFY)                    :
<1> ;                                                      :
<1> ;      FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST     :
<1> ;               2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.:
<1> ;               F = 00H FOR A GOOD SECTOR                    :
<1> ;                   80H FOR A BAD SECTOR                     :
<1> ;               N = SECTOR NUMBER                            :
<1> ;               FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK        :
<1> ;               THE TABLE SHOULD BE:                         :
<1> ;                                                      :
<1> ;         DB    00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH     :
<1> ;         DB    00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH     :
<1> ;         DB    00H,07H,00H,10H,00H,08H,00H,11H,00H,09H            :
<1> ;                                                      :
<1> ;------------------------------------------------------------------------------
<1>
<1> ;------------------------------------------------------------------------------
<1> ; OUTPUT                                                     :
```

```
<1> ;AH = STATUS OF CURRENT OPERATION                           :
<1> ;     STATUS BITS ARE DEFINED IN THE EQUATES BELOW                :
<1> ;CY = 0      SUCCESSFUL OPERATION (AH=0 ON RETURN)                      :
<1> ;CY = 1      FAILED OPERATION (AH HAS ERROR REASON)                     :
<1> ;                                                 :
<1> ;NOTE: ERROR 11H  INDICATES THAT THE DATA READ HAD A RECOVERABLE      :
<1> ;      ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM.  THE DATA      :
<1> ;      IS PROBABLY GOOD,   HOWEVER THE BIOS ROUTINE INDICATES AN      :
<1> ;      ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE      :
<1> ;      FOR ITSELF.  THE  ERROR  MAY  NOT  RECUR  IF  THE DATA IS      :
<1> ;      REWRITTEN.                                     :
<1> ;                                                 :
<1> ;IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H),                :
<1> ;   INPUT:                                        :
<1> ;     (DL) = DRIVE NUMBER                               :
<1> ;   OUTPUT:                                       :
<1> ;     (DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2)  :
<1> ;          (CONTROLLER CARD ZERO TALLY ONLY)                  :
<1> ;     (DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER            :
<1> ;     (CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER        :
<1> ;     (CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER          :
<1> ;          AND CYLINDER NUMBER HIGH BITS                    :
<1> ;                                                 :
<1> ;IF READ DASD TYPE WAS REQUESTED,                           :
<1> ;                                                 :
<1> ;AH = 0 - NOT PRESENT                                   :
<1> ;     1 - DISKETTE - NO CHANGE LINE AVAILABLE                 :
<1> ;     2 - DISKETTE - CHANGE LINE AVAILABLE                  :
<1> ;     3 - FIXED DISK                                    :
<1> ;                                                 :
<1> ;CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3                :
<1> ;                                                 :
<1> ;REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN        :
<1> ;INFORMATION.                                           :
<1> ;                                                 :
<1> ;NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE        :
<1> ;      ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION.           :
<1> ;                                                 :
<1> ;------------------------------------------------------------------------------
<1>
<1> SENSE_FAIL   EQU   0FFH        ; NOT IMPLEMENTED
<1> NO_ERR       EQU   0E0H        ; STATUS ERROR/ERROR REGISTER=0
<1> WRITE_FAULT  EQU   0CCH        ; WRITE FAULT ON SELECTED DRIVE
<1> UNDEF_ERR    EQU   0BBH        ; UNDEFINED ERROR OCCURRED
<1> NOT_RDY      EQU   0AAH        ; DRIVE NOT READY
<1> TIME_OUT     EQU   80H         ; ATTACHMENT FAILED TO RESPOND
<1> BAD_SEEK     EQU   40H         ; SEEK OPERATION FAILED
<1> BAD_CNTLR    EQU   20H         ; CONTROLLER HAS FAILED
<1> DATA_CORRECTED     EQU   11H         ; ECC CORRECTED DATA ERROR
<1> BAD_ECC      EQU   10H         ; BAD ECC ON DISK READ
<1> BAD_TRACK    EQU   0BH         ; NOT IMPLEMENTED
<1> BAD_SECTOR   EQU   0AH         ; BAD SECTOR FLAG DETECTED
<1> ;DMA_BOUNDARYEQU   09H         ; DATA EXTENDS TOO FAR
<1> INIT_FAIL    EQU   07H         ; DRIVE PARAMETER ACTIVITY FAILED
<1> BAD_RESET    EQU   05H         ; RESET FAILED
<1> ;RECORD_NOT_FND    EQU   04H         ; REQUESTED SECTOR NOT FOUND
<1> ;BAD_ADDR_MARK     EQU   02H         ; ADDRESS MARK NOT FOUND
<1> ;BAD_CMD     EQU   01H         ; BAD COMMAND PASSED TO DISK I/O
<1>
<1> ;-------------------------------------------------------
<1> ;                                        :
<1> ; FIXED DISK PARAMETER TABLE                          :
<1> ;  -  THE TABLE IS COMPOSED OF A BLOCK DEFINED AS:     :
<1> ;                                        :
<1> ;  +0  (1 WORD) - MAXIMUM NUMBER OF CYLINDERS         :
<1> ;  +2  (1 BYTE) - MAXIMUM NUMBER OF HEADS        :
<1> ;  +3  (1 WORD) - NOT USED/SEE PC-XT             :
<1> ;  +5  (1 WORD) - STARTING WRITE PRECOMPENSATION CYL  :
<1> ;  +7  (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH :
<1> ;  +8  (1 BYTE) - CONTROL BYTE                   :
<1> ;          BIT        7 DISABLE RETRIES -OR-     :
<1> ;          BIT        6 DISABLE RETRIES          :
<1> ;          BIT        3 MORE THAN 8 HEADS        :
<1> ;  +9  (3 BYTES)- NOT USED/SEE PC-XT             :
<1> ; +12  (1 WORD) - LANDING ZONE                   :
<1> ; +14  (1 BYTE) - NUMBER OF SECTORS/TRACK        :
<1> ; +15  (1 BYTE) - RESERVED FOR FUTURE USE        :
<1> ;                                        :
<1> ; - TO DYNAMICALLY DEFINE A SET OF PARAMETERS   :
<1> ;   BUILD A TABLE FOR UP TO 15 TYPES AND PLACE  :
<1> ;   THE CORRESPONDING VECTOR INTO INTERRUPT 41  :
<1> ;   FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1.   :
<1> ;                                        :
<1> ;-------------------------------------------------------
<1>
<1> ;-------------------------------------------------------
<1> ;                                        :
<1> ; HARDWARE SPECIFIC VALUES                       :
<1> ;                                        :
<1> ;  -  CONTROLLER I/O PORT                        :
<1> ;                                        :
<1> ;     > WHEN READ FROM:                  :
```

```
8171                              <1> ;HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU) :
8172                              <1> ;HF_PORT+1 - GET ERROR REGISTER                 :
8173                              <1> ;HF_PORT+2 - GET SECTOR COUNT             :
8174                              <1> ;HF_PORT+3 - GET SECTOR NUMBER            :
8175                              <1> ;HF_PORT+4 - GET CYLINDER LOW             :
8176                              <1> ;HF_PORT+5 - GET CYLINDER HIGH (2 BITS)         :
8177                              <1> ;HF_PORT+6 - GET SIZE/DRIVE/HEAD          :
8178                              <1> ;HF_PORT+7 - GET STATUS REGISTER          :
8179                              <1> ;                                   :
8180                              <1> ;     > WHEN WRITTEN TO:                  :
8181                              <1> ;HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER) :
8182                              <1> ;HF_PORT+1 - SET PRECOMPENSATION CYLINDER :
8183                              <1> ;HF_PORT+2 - SET SECTOR COUNT             :
8184                              <1> ;HF_PORT+3 - SET SECTOR NUMBER            :
8185                              <1> ;HF_PORT+4 - SET CYLINDER LOW             :
8186                              <1> ;HF_PORT+5 - SET CYLINDER HIGH (2 BITS)         :
8187                              <1> ;HF_PORT+6 - SET SIZE/DRIVE/HEAD          :
8188                              <1> ;HF_PORT+7 - SET COMMAND REGISTER         :
8189                              <1> ;                                   :
8190                              <1> ;--------------------------------------------------------
8191                              <1>
8192                              <1> ;HF_PORT     EQU   01F0H ; DISK PORT
8193                              <1> ;HF1_PORT    equ   0170h
8194                              <1> ;HF_REG_PORT EQU   03F6H
8195                              <1> ;HF1_REG_PORTequ   0376h
8196                              <1>
8197                              <1> HDC1_BASEPORTequ   1F0h
8198                              <1> HDC2_BASEPORTequ   170h
8199                              <1>
8200 0000280B 90                  <1> align 2
8201                              <1>
8202                              <1> ;-----       STATUS REGISTER
8203                              <1>
8204                              <1> ST_ERROR     EQU   00000001B   ;
8205                              <1> ST_INDEX     EQU   00000010B   ;
8206                              <1> ST_CORRCTD   EQU   00000100B   ; ECC CORRECTION SUCCESSFUL
8207                              <1> ST_DRQ       EQU   00001000B   ;
8208                              <1> ST_SEEK_COMPLEQU   00010000B   ; SEEK COMPLETE
8209                              <1> ST_WRT_FLT   EQU   00100000B   ; WRITE FAULT
8210                              <1> ST_READY     EQU   01000000B   ;
8211                              <1> ST_BUSY      EQU   10000000B   ;
8212                              <1>
8213                              <1> ;-----       ERROR REGISTER
8214                              <1>
8215                              <1> ERR_DAM      EQU   00000001B   ; DATA ADDRESS MARK NOT FOUND
8216                              <1> ERR_TRK_0    EQU   00000010B   ; TRACK 0 NOT FOUND ON RECAL
8217                              <1> ERR_ABORT    EQU   00000100B   ; ABORTED COMMAND
8218                              <1> ;      EQU   00001000B   ; NOT USED
8219                              <1> ERR_ID       EQU   00010000B   ; ID NOT FOUND
8220                              <1> ;      EQU   00100000B   ; NOT USED
8221                              <1> ERR_DATA_ECC EQU   01000000B
8222                              <1> ERR_BAD_BLOCKEQU   10000000B
8223                              <1>
8224                              <1>
8225                              <1> RECAL_CMD    EQU   00010000B   ; DRIVE RECAL    (10H)
8226                              <1> READ_CMD     EQU   00100000B   ;    READ  (20H)
8227                              <1> WRITE_CMD    EQU   00110000B   ;    WRITE (30H)
8228                              <1> VERIFY_CMD   EQU   01000000B   ;    VERIFY      (40H)
8229                              <1> FMTTRK_CMD   EQU   01010000B   ; FORMAT TRACK   (50H)
8230                              <1> INIT_CMD     EQU   01100000B   ;  INITIALIZE    (60H)
8231                              <1> SEEK_CMD     EQU   01110000B   ;    SEEK  (70H)
8232                              <1> DIAG_CMD     EQU   10010000B   ; DIAGNOSTIC     (90H)
8233                              <1> SET_PARM_CMD EQU   10010001B   ; DRIVE PARMS    (91H)
8234                              <1> NO_RETRIES   EQU   00000001B   ; CHD MODIFIER   (01H)
8235                              <1> ECC_MODE     EQU   00000010B   ; CMD MODIFIER   (02H)
8236                              <1> BUFFER_MODE  EQU   00001000B   ; CMD MODIFIER   (08H)
8237                              <1>
8238                              <1> ;MAX_FILE    EQU   2
8239                              <1> ;S_MAX_FILE  EQU   2
8240                              <1> MAX_FILE     equ   4           ; 22/12/2014
8241                              <1> S_MAX_FILE   equ   4           ; 22/12/2014
8242                              <1>
8243                              <1> DELAY_1      EQU   25H         ; DELAY FOR OPERATION COMPLETE
8244                              <1> DELAY_2      EQU   0600H       ; DELAY FOR READY
8245                              <1> DELAY_3      EQU   0100H       ; DELAY FOR DATA REQUEST
8246                              <1>
8247                              <1> HF_FAIL      EQU   08H         ; CMOS FLAG IN BYTE 0EH
8248                              <1>
8249                              <1> ;-----       COMMAND BLOCK REFERENCE
8250                              <1>
8251                              <1> ;CMD_BLOCK      EQU     BP-8           ; @CMD_BLOCK REFERENCES BLOCK HEAD IN SS
8252                              <1>                          ;  (BP) POINTS TO COMMAND BLOCK TAIL
8253                              <1>                          ;    AS DEFINED BY THE "ENTER" PARMS
8254                              <1> ; 19/12/2014
8255                              <1> ORG_VECTOR   equ   4*13h       ; INT 13h vector
8256                              <1> DISK_VECTOR  equ   4*40h       ; INT 40h vector (for floppy disks)
8257                              <1> ;HDISK_INT   equ   4*76h       ; Primary HDC - Hardware interrupt (IRQ14)
8258                              <1> ;HDISK_INT1  equ   4*76h       ; Primary HDC - Hardware interrupt (IRQ14)
8259                              <1> ;HDISK_INT2  equ   4*77h       ; Secondary HDC - Hardware interrupt (IRQ15)
8260                              <1> ;HF_TBL_VEC  equ   4*41h       ; Pointer to 1st fixed disk parameter table
8261                              <1> ;HF1_TBL_VEC equ   4*46h       ; Pointer to 2nd fixed disk parameter table
8262                              <1>
```

```
                             <1> align 2
                             <1>
                             <1> ;----------------------------------------------------------------
                             <1> ; FIXED DISK I/O SETUP                                       :
                             <1> ;                                                  :
                             <1> ;  -  ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK          :
                             <1> ;  -  PERFORM POWER ON DIAGNOSTICS                           :
                             <1> ;     SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED       :
                             <1> ;                                                  :
                             <1> ;----------------------------------------------------------------
                             <1>
                             <1> DISK_SETUP:
                             <1>  ;CLI
                             <1>  ;;MOV AX,ABS0                 ; GET ABSOLUTE SEGMENT
                             <1>  ;xor  ax,ax
                             <1>  ;MOV  DS,AX                   ; SET SEGMENT REGISTER
                             <1>  ;MOV  AX, [ORG_VECTOR]        ; GET DISKETTE VECTOR
                             <1>  ;MOV  [DISK_VECTOR],AX        ;  INTO INT 40H
                             <1>  ;MOV  AX, [ORG_VECTOR+2]
                             <1>  ;MOV  [DISK_VECTOR+2],AX
                             <1>  ;MOV  word [ORG_VECTOR],DISK_IO     ; FIXED DISK HANDLER
                             <1>  ;MOV  [ORG_VECTOR+2],CS
                             <1>  ; 1st controller (primary master, slave)   - IRQ 14
                             <1>  ;;MOV word [HDISK_INT],HD_INT       ; FIXED DISK INTERRUPT
                             <1>  ;mov  word [HDISK_INT1],HD_INT      ;
                             <1>  ;;MOV [HDISK_INT+2],CS
                             <1>  ;mov  [HDISK_INT1+2],CS
                             <1>  ; 2nd controller (secondary master, slave) - IRQ 15
                             <1>  ;mov  word [HDISK_INT2],HD1_INT     ;
                             <1>  ;mov  [HDISK_INT2+2],CS
                             <1>  ;
                             <1>  ;;MOV word [HF_TBL_VEC],HD0_DPT     ; PARM TABLE DRIVE 80
                             <1>  ;;MOV word [HF_TBL_VEC+2],DPT_SEGM
                             <1>  ;;MOV word [HF1_TBL_VEC],HD1_DPT    ; PARM TABLE DRIVE 81
                             <1>  ;;MOV word [HF1_TBL_VEC+2],DPT_SEGM
                             <1>  ;push cs
                             <1>  ;pop  ds
                             <1>  ;mov  word [HDPM_TBL_VEC],HD0_DPT   ; PARM TABLE DRIVE 80h
                             <1>  ;mov  word [HDPM_TBL_VEC+2],DPT_SEGM
0000280C C705[44710000]0000- <1>  mov   dword [HDPM_TBL_VEC], (DPT_SEGM*16)+HD0_DPT
00002814 0900                <1>
                             <1>  ;mov  word [HDPS_TBL_VEC],HD1_DPT   ; PARM TABLE DRIVE 81h
                             <1>  ;mov  word [HDPS_TBL_VEC+2],DPT_SEGM
00002816 C705[48710000]2000- <1>  mov   dword [HDPS_TBL_VEC], (DPT_SEGM*16)+HD1_DPT
0000281E 0900                <1>
                             <1>  ;mov  word [HDSM_TBL_VEC],HD2_DPT   ; PARM TABLE DRIVE 82h
                             <1>  ;mov  word [HDSM_TBL_VEC+2],DPT_SEGM
00002820 C705[4C710000]4000- <1>  mov   dword [HDSM_TBL_VEC], (DPT_SEGM*16)+HD2_DPT
00002828 0900                <1>
                             <1>  ;mov  word [HDSS_TBL_VEC],HD3_DPT   ; PARM TABLE DRIVE 83h
                             <1>  ;mov  word [HDSS_TBL_VEC+2],DPT_SEGM
0000282A C705[50710000]6000- <1>  mov   dword [HDSS_TBL_VEC], (DPT_SEGM*16)+HD3_DPT
00002832 0900                <1>
                             <1>  ;
                             <1>  ;;IN  AL,INTB01        ; TURN ON SECOND INTERRUPT CHIP
                             <1>  ;;;AND      AL,0BFH
                             <1>  ;;and al, 3Fh                 ; enable IRQ 14 and IRQ 15
                             <1>  ;;;JMP      $+2
                             <1>  ;;IODELAY
                             <1>  ;;OUT INTB01,AL
                             <1>  ;;IODELAY
                             <1>  ;;IN  AL,INTA01        ; LET INTERRUPTS PASS THRU TO
                             <1>  ;;AND AL,0FBH          ;  SECOND CHIP
                             <1>  ;;;JMP      $+2
                             <1>  ;;IODELAY
                             <1>  ;;OUT INTA01,AL
                             <1>  ;
                             <1>  ;STI
                             <1>  ;;PUSH      DS                ; MOVE ABS0 POINTER TO
                             <1>  ;;POP ES               ; EXTRA SEGMENT POINTER
                             <1>  ;;;CALL     DDS               ; ESTABLISH DATA SEGMENT
                             <1>  ;;MOV byte [DISK_STATUS1],0   ; RESET THE STATUS INDICATOR
                             <1>  ;;MOV byte [HF_NUM],0         ; ZERO NUMBER OF FIXED DISKS
                             <1>  ;;MOV byte [CONTROL_BYTE],0
                             <1>  ;;MOV byte [PORT_OFF],0 ; ZERO CARD OFFSET
                             <1>  ; 20/12/2014 - private code by Erdogan Tan
                             <1>            ; (out of original PC-AT, PC-XT BIOS code)
                             <1>  ;mov  si, hd0_type
00002834 BE[DA6D0000]        <1>  mov   esi, hd0_type
                             <1>  ;mov  cx, 4
00002839 B904000000          <1>  mov   ecx, 4
                             <1> hde_l:
0000283E AC                  <1>  lodsb
0000283F 3C80                <1>  cmp   al, 80h                 ; 8?h = existing
00002841 7206                <1>  jb    short _L4
00002843 FE05[40710000]      <1>  inc   byte [HF_NUM]           ; + 1 hard (fixed) disk drives
                             <1> _L4: ; 26/02/2015
00002849 E2F3                <1>  loop  hde_l
                             <1> ;_L4:                          ; 0 <= [HF_NUM] =< 4
                             <1> ;L4:
                             <1>  ;
                             <1>  ;; 31/12/2014 - cancel controller diagnostics here
```

```
                            <1>  ;;;mov      cx, 3  ; 26/12/2014 (Award BIOS 1999)
                            <1>  ;;mov      cl, 3
                            <1>  ;;
                            <1>  ;;MOV DL,80H                  ; CHECK THE CONTROLLER
                            <1> ;;hdc_dl:
                            <1>  ;;MOV AH,14H                  ; USE CONTROLLER DIAGNOSTIC COMMAND
                            <1>  ;;INT 13H              ; CALL BIOS WITH DIAGNOSTIC COMMAND
                            <1>  ;;;JC short CTL_ERRX          ; DISPLAY ERROR MESSAGE IF BAD RETURN
                            <1>  ;;;jc short POD_DONE ;22/12/2014
                            <1>  ;;jnc short hdc_reset0
                            <1>  ;;loop      hdc_dl
                            <1>  ;;; 27/12/2014
                            <1>  ;;stc
                            <1>  ;;retn
                            <1>  ;
                            <1> ;;hdc_reset0:
                            <1>  ; 18/01/2015
0000284B 8A0D[40710000]     <1>  mov   cl, [HF_NUM]
00002851 20C9               <1>  and   cl, cl
00002853 740D               <1>  jz    short POD_DONE
                            <1>  ;
00002855 B27F               <1>  mov   dl, 7Fh
                            <1> hdc_reset1:
00002857 FEC2               <1>  inc   dl
                            <1>  ;; 31/12/2015
                            <1>  ;;push      dx
                            <1>  ;;push      cx
                            <1>  ;;push      ds
                            <1>  ;;sub ax, ax
                            <1>  ;;mov ds, ax
                            <1>  ;;MOV AX, [TIMER_LOW]         ; GET START TIMER COUNTS
                            <1>  ;;pop ds
                            <1>  ;;MOV BX,AX
                            <1>  ;;ADD AX,6*182          ; 60 SECONDS* 18.2
                            <1>  ;;MOV CX,AX
                            <1>  ;;mov word [wait_count], 0    ; 22/12/2014 (reset wait counter)
                            <1>  ;;
                            <1>  ;; 31/12/2014 - cancel HD_RESET_1
                            <1>  ;;CALL      HD_RESET_1        ; SET UP DRIVE 0, (1,2,3)
                            <1>  ;;pop cx
                            <1>  ;;pop dx
                            <1>  ;;
                            <1>  ; 18/01/2015
00002859 B40D               <1>  mov   ah, 0Dh ; ALTERNATE RESET
                            <1>  ;int  13h
0000285B E8A3FFFFFF         <1>  call  int13h
00002860 E2F5               <1>  loop  hdc_reset1
                            <1> POD_DONE:
00002862 C3                 <1>  RETn
                            <1>
                            <1> ;;-----POD_ERROR
                            <1>
                            <1> ;;CTL_ERRX:
                            <1> ;;MOV  SI,OFFSET F1782   ; CONTROLLER ERROR
                            <1> ;;CALL SET_FAIL          ; DO NOT IPL FROM DISK
                            <1> ;;CALL E_MSG             ; DISPLAY ERROR AND SET (BP) ERROR FLAG
                            <1> ;;JMP  short POD_DONE
                            <1>
                            <1> ;;HD_RESET_1:
                            <1> ;;      ;PUSH BX               ; SAVE TIMER LIMITS
                            <1> ;;      ;PUSH CX
                            <1> ;;RES_1: MOV AH,09H                  ; SET DRIVE PARAMETERS
                            <1> ;;      INT   13H
                            <1> ;;      JC    short RES_2
                            <1> ;;      MOV   AH,11H                 ; RECALIBRATE DRIVE
                            <1> ;;      INT   13H
                            <1> ;;      JNC   short RES_CK           ; DRIVE OK
                            <1> ;;RES_2: ;CALL     POD_TCHK          ; CHECK TIME OUT
                            <1> ;;      cmp   word [wait_count], 6*182 ; waiting time (in timer ticks)
                            <1> ;;                             ; (30 seconds)
                            <1> ;;      ;cmc
                            <1> ;;      ;JNC  short RES_1
                            <1> ;;      jb    short RES_1
                            <1> ;;;RES_FL: ;MOV    SI,OFFSET F1781   ; INDICATE DISK 1 FAILURE;
                            <1> ;;      ;TEST DL,1
                            <1> ;;      ;JNZ  RES_E1
                            <1> ;;      ;MOV  SI,OFFSET F1780   ; INDICATE DISK 0 FAILURE
                            <1> ;;      ;CALL SET_FAIL          ; DO NOT TRY TO IPL DISK 0
                            <1> ;;      ;JMP  SHORT RES_E1
                            <1> ;;RES_ER: ; 22/12/2014
                            <1> ;;RES_OK:
                            <1> ;;      ;POP  CX               ; RESTORE TIMER LIMITS
                            <1> ;;      ;POP  BX
                            <1> ;;      RETn
                            <1> ;;
                            <1> ;;RES_RS: MOVAH,00H                  ; RESET THE DRIVE
                            <1> ;;      INT   13H
                            <1> ;;RES_CK: MOVAH,08H                  ; GET MAX CYLINDER,HEAD,SECTOR
                            <1> ;;      MOV   BL,DL             ; SAVE DRIVE CODE
                            <1> ;;      INT   13H
                            <1> ;;      JC    short RES_ER
                            <1> ;;      MOV   [NEC_STATUS],CX   ; SAVE MAX CYLINDER, SECTOR
```

```
                             <1> ;;     MOV   DL,BL             ; RESTORE DRIVE CODE
                             <1> ;;RES_3: MOV AX,0401H          ; VERIFY THE LAST SECTOR
                             <1> ;;     INT   13H
                             <1> ;;     JNC   short RES_OK            ; VERIFY OK
                             <1> ;;     CMP   AH,BAD_SECTOR           ; OK ALSO IF JUST ID READ
                             <1> ;;     JE    short RES_OK
                             <1> ;;     CMP   AH,DATA_CORRECTED
                             <1> ;;     JE    short RES_OK
                             <1> ;;     CMP   AH,BAD_ECC
                             <1> ;;     JE    short RES_OK
                             <1> ;;     ;CALL POD_TCHK          ; CHECK FOR TIME OUT
                             <1> ;;     cmp   word [wait_count], 6*182 ; waiting time (in timer ticks)
                             <1> ;;                             ; (60 seconds)
                             <1> ;;     cmc
                             <1> ;;     JC    short RES_ER            ; FAILED
                             <1> ;;     MOV   CX,[NEC_STATUS]   ; GET SECTOR ADDRESS, AND CYLINDER
                             <1> ;;     MOV   AL,CL             ; SEPARATE OUT SECTOR NUMBER
                             <1> ;;     AND   AL,3FH
                             <1> ;;     DEC   AL                ; TRY PREVIOUS ONE
                             <1> ;;     JZ    short RES_RS            ; WE'VE TRIED ALL SECTORS ON TRACK
                             <1> ;;     AND   CL,0C0H           ; KEEP CYLINDER BITS
                             <1> ;;     OR    CL,AL             ; MERGE SECTOR WITH CYLINDER BITS
                             <1> ;;     MOV   [NEC_STATUS],CX   ; SAVE CYLINDER, NEW SECTOR NUMBER
                             <1> ;;     JMP   short RES_3       ; TRY AGAIN
                             <1> ;;;RES_ER: MOV     SI,OFFSET F1791   ; INDICATE DISK 1 ERROR
                             <1> ;;     ;TEST DL,1
                             <1> ;;     ;JNZ  short RES_E1
                             <1> ;;     ;MOV  SI,OFFSET F1790   ; INDICATE DISK 0 ERROR
                             <1> ;;;RES_E1:
                             <1> ;;     ;CALL E_MSG             ; DISPLAY ERROR AND SET (BP) ERROR FLAG
                             <1> ;;;RES_OK:
                             <1> ;;     ;POP  CX                ; RESTORE TIMER LIMITS
                             <1> ;;     ;POP  BX
                             <1> ;;     ;RETn
                             <1> ;
                             <1> ;;SET_FAIL:
                             <1> ;;MOV  AX,X*(CMOS_DIAG+NMI)    ; GET CMOS ERROR BYTE
                             <1> ;;CALL CMOS_READ
                             <1> ;;OR   AL,HF_FAIL        ; SET DO NOT IPL FROM DISK FLAG
                             <1> ;;XCHG AH,AL             ; SAVE IT
                             <1> ;;CALL CMOS_WRITE        ; PUT IT OUT
                             <1> ;;RETn
                             <1> ;
                             <1> ;;POD_TCHK:                    ; CHECK FOR 30 SECOND TIME OUT
                             <1> ;;POP  AX                ; SAVE RETURN
                             <1> ;;POP  CX                ; GET TIME OUT LIMITS
                             <1> ;;POP  BX
                             <1> ;;PUSH BX                ; AND SAVE THEM AGAIN
                             <1> ;;PUSH CX
                             <1> ;;PUSH AX
                             <1> ;;push ds
                             <1> ;;xor  ax, ax
                             <1> ;;mov  ds, ax                  ; RESTORE RETURN
                             <1> ;;MOV  AX, [TIMER_LOW]         ; AX = CURRENT TIME
                             <1> ;;                       ; BX = START TIME
                             <1> ;;                       ; CX = END TIME
                             <1> ;;pop  ds
                             <1> ;;CMP  BX,CX
                             <1> ;;JB   short TCHK1       ; START < END
                             <1> ;;CMP  BX,AX
                             <1> ;;JB   short TCHKG       ; END < START < CURRENT
                             <1> ;;JMP  SHORT TCHK2       ; END, CURRENT < START
                             <1> ;;TCHK1: CMP AX,BX
                             <1> ;;     JB    short TCHKNG            ; CURRENT < START < END
                             <1> ;;TCHK2: CMP AX,CX
                             <1> ;;     JB    short TCHKG       ; START < CURRENT < END
                             <1> ;;                             ; OR CURRENT < END < START
                             <1> ;;TCHKNG: STC                  ; CARRY SET INDICATES TIME OUT
                             <1> ;;     RETn
                             <1> ;;TCHKG: CLC                   ; INDICATE STILL TIME
                             <1> ;;     RETn
                             <1> ;;
                             <1> ;;int_13h:
                             <1>
                             <1> ;----------------------------------------
                             <1> ;FIXED DISK BIOS ENTRY POINT   :
                             <1> ;----------------------------------------
                             <1>
                             <1> DISK_IO:
00002863 80FA80              <1>  CMP   DL,80H                  ; TEST FOR FIXED DISK DRIVE
                             <1>  ;JAE  short A1          ; YES, HANDLE HERE
                             <1>  ;;;INT      40H               ; DISKETTE HANDLER
                             <1>  ;;call      int40h
00002866 0F8225F1FFFF        <1>  jb    DISKETTE_IO_1
                             <1> ;RET_2:
                             <1>  ;RETf 2                 ; BACK TO CALLER
                             <1> ;retf  4
                             <1> A1:
0000286C FB                  <1>  STI                     ; ENABLE INTERRUPTS
                             <1>  ;; 04/01/2015
                             <1>  ;;OR  AH,AH
                             <1>  ;;JNZ short A2
```

```
8539                                 <1>  ;;INT 40H              ; RESET NEC WHEN AH=0
8540                                 <1>  ;;SUB AH,AH
8541 0000286D 80FA83                 <1>  CMP   DL,(80H + S_MAX_FILE - 1)
8542 00002870 772C                   <1>  JA    short RET_2
8543                                 <1>  ; 18/01/2015
8544 00002872 08E4                   <1>  or    ah,ah
8545 00002874 742B                   <1>  jz    short A4
8546 00002876 80FC0D                 <1>  cmp   ah, 0Dh     ; Alternate reset
8547 00002879 7504                   <1>  jne   short A2
8548 0000287B 28E4                   <1>  sub   ah,ah ; Reset
8549 0000287D EB22                   <1>  jmp   short A4
8550                                 <1> A2:
8551 0000287F 80FC08                 <1>  CMP   AH,08H                 ; GET PARAMETERS IS A SPECIAL CASE
8552                                 <1>  ;JNZ  short A3
8553                                 <1>        ;JMP    GET_PARM_N
8554 00002882 0F841C030000           <1>  je    GET_PARM_N
8555 00002888 80FC15                 <1> A3:    CMP   AH,15H                 ; READ DASD TYPE IS ALSO
8556                                 <1>  ;JNZ  short A4
8557                                 <1>        ;JMP    READ_DASD_TYPE
8558 0000288B 0F84C7020000           <1>        je      READ_DASD_TYPE
8559                                 <1>  ; 02/02/2015
8560 00002891 80FC1D                 <1>  cmp   ah, 1Dh                ;(Temporary for Retro UNIX 386 v1)
8561                                 <1>  ; 12/01/2015
8562 00002894 F5                     <1>  cmc
8563 00002895 730A                   <1>  jnc   short A4
8564                                 <1>  ; 30/01/2015
8565                                 <1>  ;mov     byte [CS:DISK_STATUS1],BAD_CMD  ; COMMAND ERROR
8566 00002897 C605[3F710000]01       <1>        mov     byte [DISK_STATUS1], BAD_CMD
8567                                 <1>  ;jmp  short RET_2
8568                                 <1> RET_2:
8569 0000289E CA0400                 <1>  retf  4
8570                                 <1> A4:                          ; SAVE REGISTERS DURING OPERATION
8571 000028A1 C8080000               <1>  ENTER 8,0              ; SAVE (BP) AND MAKE ROOM FOR @CMD_BLOCK
8572 000028A5 53                     <1>  PUSH  eBX              ;  IN THE STACK, THE COMMAND BLOCK IS:
8573 000028A6 51                     <1>  PUSH  eCX              ;   @CMD_BLOCK == BYTE PTR [BP]-8
8574 000028A7 52                     <1>  PUSH  eDX
8575 000028A8 1E                     <1>  PUSH  DS
8576 000028A9 06                     <1>  PUSH  ES
8577 000028AA 56                     <1>  PUSH  eSI
8578 000028AB 57                     <1>  PUSH  eDI
8579                                 <1>  ;;04/01/2015
8580                                 <1>  ;;OR  AH,AH             ; CHECK FOR RESET
8581                                 <1>  ;;JNZ short A5
8582                                 <1>  ;;MOV DL,80H                  ; FORCE DRIVE 80 FOR RESET
8583                                 <1> ;;A5:
8584                                 <1>  ;push cs
8585                                 <1>  ;pop  ds
8586                                 <1>  ; 21/02/2015
8587 000028AC 6650                   <1>  push  ax
8588 000028AE 66B81000               <1>  mov   ax, KDATA
8589 000028B2 8ED8                   <1>  mov   ds, ax
8590 000028B4 8EC0                   <1>  mov   es, ax
8591 000028B6 6658                   <1>  pop   ax
8592 000028B8 E889000000             <1>  CALL  DISK_IO_CONT           ; PERFORM THE OPERATION
8593                                 <1>  ;;CALL      DDS               ; ESTABLISH SEGMENT
8594 000028BD 8A25[3F710000]         <1>  MOV   AH,[DISK_STATUS1] ; GET STATUS FROM OPERATION
8595 000028C3 80FC01                 <1>  CMP   AH,1             ; SET THE CARRY FLAG TO INDICATE
8596 000028C6 F5                     <1>  CMC                    ; SUCCESS OR FAILURE
8597 000028C7 5F                     <1>  POP   eDI              ; RESTORE REGISTERS
8598 000028C8 5E                     <1>  POP   eSI
8599 000028C9 07                     <1>        POP     ES
8600 000028CA 1F                     <1>        POP     DS
8601 000028CB 5A                     <1>  POP   eDX
8602 000028CC 59                     <1>  POP   eCX
8603 000028CD 5B                     <1>  POP   eBX
8604 000028CE C9                     <1>  LEAVE                  ; ADJUST (SP) AND RESTORE (BP)
8605                                 <1>  ;RETf 2                ; THROW AWAY SAVED FLAGS
8606 000028CF CA0400                 <1>  retf  4
8607                                 <1> ; 21/02/2015
8608                                 <1> ;       dw --> dd
8609                                 <1> M1:                          ; FUNCTION TRANSFER TABLE
8610 000028D2 [942A0000]             <1>  dd    DISK_RESET       ; 000H
8611 000028D6 [0B2B0000]             <1>  dd    RETURN_STATUS          ; 001H
8612 000028DA [182B0000]             <1>  dd    DISK_READ        ; 002H
8613 000028DE [212B0000]             <1>  dd    DISK_WRITE       ; 003H
8614 000028E2 [2A2B0000]             <1>  dd    DISK_VERF        ; 004H
8615 000028E6 [422B0000]             <1>  dd    FMT_TRK          ; 005H
8616 000028EA [8A2A0000]             <1>  dd    BAD_COMMAND      ; 006H      FORMAT BAD SECTORS
8617 000028EE [8A2A0000]             <1>  dd    BAD_COMMAND      ; 007H      FORMAT DRIVE
8618 000028F2 [8A2A0000]             <1>  dd    BAD_COMMAND      ; 008H      RETURN PARAMETERS
8619 000028F6 [092C0000]             <1>  dd    INIT_DRV         ; 009H
8620 000028FA [682C0000]             <1>  dd    RD_LONG          ; 00AH
8621 000028FE [712C0000]             <1>  dd    WR_LONG          ; 00BH
8622 00002902 [7A2C0000]             <1>  dd    DISK_SEEK        ; 00CH
8623 00002906 [942A0000]             <1>  dd    DISK_RESET       ; 00DH
8624 0000290A [8A2A0000]             <1>  dd    BAD_COMMAND      ; 00EH      READ BUFFER
8625 0000290E [8A2A0000]             <1>  dd    BAD_COMMAND      ; 00FH      WRITE BUFFER
8626 00002912 [A22C0000]             <1>  dd    TST_RDY          ; 010H
8627 00002916 [C62C0000]             <1>  dd    HDISK_RECAL      ; 011H
8628 0000291A [8A2A0000]             <1>  dd    BAD_COMMAND      ; 012H      MEMORY DIAGNOSTIC
8629 0000291E [8A2A0000]             <1>  dd    BAD_COMMAND      ; 013H      DRIVE DIAGNOSTIC
8630 00002922 [FC2C0000]             <1>  dd    CTLR_DIAGNOSTIC  ; 014H      CONTROLLER DIAGNOSTIC
```

```
                           <1>  ; 02/02/2015 (Temporary - Retro UNIX 386 v1 - DISK I/O test)
00002926 [8A2A0000]        <1>  dd    BAD_COMMAND      ; 015h
0000292A [8A2A0000]        <1>  dd    BAD_COMMAND      ; 016h
0000292E [8A2A0000]        <1>  dd    BAD_COMMAND      ; 017h
00002932 [8A2A0000]        <1>  dd    BAD_COMMAND      ; 018h
00002936 [8A2A0000]        <1>  dd    BAD_COMMAND      ; 019h
0000293A [8A2A0000]        <1>  dd    BAD_COMMAND      ; 01Ah
0000293E [182B0000]        <1>  dd    DISK_READ        ; 01Bh ; LBA read
00002942 [212B0000]        <1>  dd    DISK_WRITE       ; 01Ch ; LBA write
                           <1> M1L     EQU    $-M1
                           <1> 
                           <1> DISK_IO_CONT:
                           <1>  ;;CALL      DDS               ; ESTABLISH SEGMENT
00002946 80FC01            <1>  CMP   AH,01H                  ; RETURN STATUS
                           <1>  ;;JNZ short SU0
                           <1>         ;;JMP    RETURN_STATUS
00002949 0F84BC010000      <1>  je    RETURN_STATUS
                           <1> SU0:
0000294F C605[3F710000]00  <1>  MOV   byte [DISK_STATUS1],0   ; RESET THE STATUS INDICATOR
                           <1>  ;;PUSH      BX                ; SAVE DATA ADDRESS
                           <1>  ;mov  si, bx ;; 14/02/2015
00002956 89DE              <1>  mov   esi, ebx ; 21/02/2015
00002958 8A1D[40710000]    <1>  MOV   BL,[HF_NUM]       ; GET NUMBER OF DRIVES
                           <1>  ;; 04/01/2015
                           <1>  ;;PUSH      AX
0000295E 80E27F            <1>  AND   DL,7FH                  ; GET DRIVE AS 0 OR 1
                           <1>                         ; (get drive number as 0 to 3)
00002961 38D3              <1>  CMP   BL,DL
                           <1>         ;;JBE   BAD_COMMAND_POP         ; INVALID DRIVE
00002963 0F8621010000      <1>         jbe     BAD_COMMAND ;; 14/02/2015
                           <1>         ;
                           <1>  ;;03/01/2015
00002969 29DB              <1>  sub   ebx, ebx
0000296B 88D3              <1>  mov   bl, dl
                           <1>  ;sub  bh, bh
0000296D 883D[54710000]    <1>  mov   [LBAMode], bh     ; 0
                           <1>  ;;test      byte [bx+hd0_type], 1   ; LBA ready ?
                           <1>  ;test byte [ebx+hd0_type], 1
                           <1>  ;jz   short su1         ; no
                           <1>  ;inc  byte [LBAMode]
                           <1> ;su1:
                           <1>  ; 21/02/2015 (32 bit modification)
                           <1>  ;04/01/2015
00002973 6650              <1>  push  ax ; ***
                           <1>  ;PUSH ES ; **
00002975 6652              <1>  PUSH  DX ; *
00002977 6650              <1>  push  ax
00002979 E85C060000        <1>  CALL  GET_VEC           ; GET DISK PARAMETERS
                           <1>  ; 02/02/2015
                           <1>  ;mov  ax, [ES:BX+16] ; I/O port base address (1F0h, 170h)
0000297E 668B4310          <1>  mov   ax, [ebx+16]
00002982 66A3[546B0000]    <1>  mov   [HF_PORT], ax
                           <1>  ;mov  dx, [ES:BX+18] ; control port address (3F6h, 376h)
00002988 668B5312          <1>  mov   dx, [ebx+18]
0000298C 668915[566B0000]  <1>  mov   [HF_REG_PORT], dx
                           <1>  ;mov  al, [ES:BX+20] ; head register upper nibble (A0h,B0h,E0h,F0h)
00002993 8A4314            <1>  mov   al, [ebx+20]
                           <1>  ; 23/02/2015
00002996 A840              <1>  test  al, 40h      ; LBA bit (bit 6)
00002998 7406              <1>  jz    short su1
0000299A FE05[54710000]    <1>  inc   byte [LBAMode] ; 1
                           <1> su1:
000029A0 C0E804            <1>  shr   al, 4
000029A3 2401              <1>  and   al, 1
000029A5 A2[586B0000]      <1>  mov   [hf_m_s], al
                           <1>  ;
                           <1>  ; 03/01/2015
                           <1>  ;MOV  AL,byte [ES:BX+8] ; GET CONTROL BYTE MODIFIER
000029AA 8A4308            <1>  mov   al, [ebx+8]
                           <1>  ;MOV  DX,[HF_REG_PORT]  ; Device Control register
000029AD EE                <1>  OUT   DX,AL             ; SET EXTRA HEAD OPTION
                           <1>                         ; Control Byte:  (= 08h, here)
                           <1>                         ; bit 0 - 0
                           <1>                         ; bit 1 - nIEN (1 = disable irq)
                           <1>                         ; bit 2 - SRST (software RESET)
                           <1>                         ; bit 3 - use extra heads (8 to 15)
                           <1>                         ;         -always set to 1-
                           <1>                         ; (bits 3 to 7 are reserved
                           <1>                         ;          for ATA devices)
000029AE 8A25[41710000]    <1>  MOV   AH,[CONTROL_BYTE] ; SET EXTRA HEAD OPTION IN
000029B4 80E4C0            <1>  AND   AH,0C0H           ; CONTROL BYTE
000029B7 08C4              <1>  OR    AH,AL
000029B9 8825[41710000]    <1>  MOV   [CONTROL_BYTE],AH
                           <1>  ; 04/01/2015
000029BF 6658              <1>  pop   ax
000029C1 665A              <1>  pop   dx ; * ;; 14/02/2015
000029C3 20E4              <1>  and   ah, ah      ; Reset function ?
000029C5 7507              <1>  jnz   short su2
                           <1>  ;;pop dx ; * ;; 14/02/2015
                           <1>  ;pop  es ; **
000029C7 6658              <1>  pop   ax ; ***
                           <1>  ;;pop bx
```

```
000029C9 E9C6000000          <1>         jmp     DISK_RESET
                             <1> su2:
000029CE 803D[54710000]00    <1>  cmp   byte [LBAMode], 0
000029D5 7661                <1>  jna   short su3
                             <1>  ;
                             <1>  ; 02/02/2015 (LBA read/write function calls)
000029D7 80FC1B              <1>  cmp   ah, 1Bh
000029DA 720B                <1>  jb    short lbarw1
000029DC 80FC1C              <1>  cmp   ah, 1Ch
000029DF 775C                <1>  ja    short invldfnc
                             <1>  ;;pop dx ; * ; 14/02/2015
                             <1>  ;mov  ax, cx ; Lower word of LBA address (bits 0-15)
000029E1 89C8                <1>  mov   eax, ecx ; LBA address (21/02/2015)
                             <1>  ;; 14/02/2015
000029E3 88D1                <1>  mov   cl, dl ; 14/02/2015
                             <1>  ;;mov dx, bx
                             <1>  ;mov  dx, si ; higher word of LBA address (bits 16-23)
                             <1>  ;;mov bx, di
                             <1>  ;mov  si, di ; Buffer offset
000029E5 EB31                <1>  jmp   short lbarw2
                             <1> lbarw1:
                             <1>  ; convert CHS to LBA
                             <1>  ;
                             <1>  ; LBA calculation - AWARD BIOS - 1999 - AHDSK.ASM
                             <1>  ; LBA = "# of Heads" * Sectors/Track * Cylinder + Head * Sectors/Track
                             <1>  ;     + Sector - 1
000029E7 6652                <1>  push  dx ; * ;; 14/02/2015
                             <1>  ;xor  dh, dh
000029E9 31D2                <1>  xor   edx, edx
                             <1>  ;mov  dl, [ES:BX+14]    ; sectors per track (logical)
000029EB 8A530E              <1>  mov   dl, [ebx+14]
                             <1>  ;xor  ah, ah
000029EE 31C0                <1>  xor   eax, eax
                             <1>  ;mov  al, [ES:BX+2]     ; heads (logical)
000029F0 8A4302              <1>  mov   al, [ebx+2]
000029F3 FEC8                <1>  dec   al
000029F5 6640                <1>  inc   ax          ; 0 =  256
000029F7 66F7E2              <1>  mul   dx
                             <1>        ; AX = # of Heads" * Sectors/Track
000029FA 6689CA              <1>  mov   dx, cx
                             <1>  ;and  cx, 3Fh      ; sector  (1 to 63)
000029FD 83E13F              <1>  and   ecx, 3fh
00002A00 86D6                <1>  xchg  dl, dh
00002A02 C0EE06              <1>  shr   dh, 6
                             <1>        ; DX = cylinder (0 to 1023)
                             <1>  ;mul  dx
                             <1>        ; DX:AX = # of Heads" * Sectors/Track * Cylinder
00002A05 F7E2                <1>  mul   edx
00002A07 FEC9                <1>  dec   cl  ; sector - 1
                             <1>  ;add  ax, cx
                             <1>  ;adc  dx, 0
                             <1>        ; DX:AX = # of Heads" * Sectors/Track * Cylinder + Sector -1
00002A09 01C8                <1>  add   eax, ecx
00002A0B 6659                <1>  pop   cx ; * ; ch = head, cl = drive number (zero based)
                             <1>  ;push dx
                             <1>  ;push ax
00002A0D 50                  <1>  push  eax
                             <1>  ;mov  al, [ES:BX+14]    ; sectors per track (logical)
00002A0E 8A430E              <1>  mov   al, [ebx+14]
00002A11 F6E5                <1>  mul   ch
                             <1>        ;  AX = Head * Sectors/Track
00002A13 6699                <1>         cwd
                             <1>  ;pop  dx
00002A15 5A                  <1>  pop   edx
                             <1>  ;add  ax, dx
                             <1>  ;pop  dx
                             <1>  ;adc  dx, 0 ; add carry bit
00002A16 01D0                <1>  add   eax, edx
                             <1> lbarw2:
00002A18 29D2                <1>  sub   edx, edx ; 21/02/2015
00002A1A 88CA                <1>  mov   dl, cl ; 21/02/2015
00002A1C C645F800            <1>         mov     byte [CMD_BLOCK], 0 ; Features Register
                             <1>                    ; NOTE: Features register (1F1h, 171h)
                             <1>                    ; is not used for ATA device R/W functions.
                             <1>                    ; It is old/obsolete 'write precompensation'
                             <1>                    ; register and error register
                             <1>                    ; for old ATA/IDE devices.
                             <1>  ; 18/01/2014
                             <1>  ;mov  ch, [hf_m_s]      ; Drive 0 (master) or 1 (slave)
00002A20 8A0D[586B0000]      <1>  mov   cl, [hf_m_s]
                             <1>  ;shl  ch, 4       ; bit 4 (drive bit)
                             <1>  ;or   ch, 0E0h    ; bit 5 = 1
                             <1>                    ; bit 6 = 1 = LBA mode
                             <1>                    ; bit 7 = 1
00002A26 80C90E              <1>  or    cl, 0Eh ; 1110b
                             <1>  ;and  dh, 0Fh           ; LBA byte 4 (bits 24 to 27)
00002A29 25FFFFFF0F          <1>  and   eax, 0FFFFFFFh
00002A2E C1E11C              <1>  shl   ecx, 28 ; 21/02/2015
                             <1>  ;or   dh, ch
00002A31 09C8                <1>  or    eax, ecx
                             <1>  ;;mov [CMD_BLOCK+2], al ; LBA byte 1 (bits 0 to 7)
                             <1>                      ; (Sector Number Register)
```

```
                            <1>  ;;mov [CMD_BLOCK+3], ah ; LBA byte 2 (bits 8 to 15)
                            <1>                      ; (Cylinder Low Register)
                            <1>  ;mov  [CMD_BLOCK+2], ax ; LBA byte 1, 2
                            <1>  ;mov  [CMD_BLOCK+4], dl ; LBA byte 3 (bits 16 to 23)
                            <1>                      ; (Cylinder High Register)
                            <1>  ;;mov [CMD_BLOCK+5], dh ; LBA byte 4 (bits 24 to 27)
                            <1>                      ; (Drive/Head Register)
                            <1>
                            <1>  ;mov  [CMD_BLOCK+4], dx ; LBA byte 4, LBA & DEV select bits
00002A33 8945FA             <1>  mov   [CMD_BLOCK+2], eax ; 21/02/2015
                            <1>  ;14/02/2015
                            <1>  ;mov  dl, cl ; Drive number (INIT_DRV)
00002A36 EB38               <1>  jmp   short su4
                            <1> su3:
                            <1>  ; 02/02/2015
                            <1>  ; (Temporary functions 1Bh & 1Ch are not valid for CHS mode)
00002A38 80FC14             <1>  cmp   ah, 14h
00002A3B 7604               <1>  jna   short chsfnc
                            <1> invldfnc:
                            <1>         ; 14/02/2015
                            <1>  ;pop  es ; **
00002A3D 6658               <1>         pop     ax ; ***
                            <1>         ;jmp     short BAD_COMMAND_POP
00002A3F EB49               <1>         jmp     short BAD_COMMAND
                            <1> chsfnc:
                            <1>  ;MOV  AX,[ES:BX+5]            ; GET WRITE PRE-COMPENSATION CYLINDER
00002A41 668B4305           <1>  mov   ax, [ebx+5]
00002A45 66C1E802           <1>  SHR   AX,2
00002A49 8845F8             <1>  MOV   [CMD_BLOCK],AL
                            <1>  ;;MOV AL,[ES:BX+8]            ; GET CONTROL BYTE MODIFIER
                            <1>  ;;PUSH      DX
                            <1>  ;;MOV DX,[HF_REG_PORT]
                            <1>  ;;OUT DX,AL            ; SET EXTRA HEAD OPTION
                            <1>  ;;POP DX ; *
                            <1>  ;;POP ES ; **
                            <1>  ;;MOV AH,[CONTROL_BYTE] ; SET EXTRA HEAD OPTION IN
                            <1>  ;;AND AH,0C0H           ; CONTROL BYTE
                            <1>  ;;OR  AH,AL
                            <1>  ;;MOV [CONTROL_BYTE],AH
                            <1>  ;
00002A4C 88C8               <1>  MOV   AL,CL             ; GET SECTOR NUMBER
00002A4E 243F               <1>  AND   AL,3FH
00002A50 8845FA             <1>  MOV   [CMD_BLOCK+2],AL
00002A53 886DFB             <1>  MOV   [CMD_BLOCK+3],CH  ; GET CYLINDER NUMBER
00002A56 88C8               <1>  MOV   AL,CL
00002A58 C0E806             <1>  SHR   AL,6
00002A5B 8845FC             <1>  MOV   [CMD_BLOCK+4],AL  ; CYLINDER HIGH ORDER 2 BITS
                            <1>  ;;05/01/2015
                            <1>  ;;MOV AL,DL             ; DRIVE NUMBER
00002A5E A0[586B0000]       <1>  mov   al, [hf_m_s]
00002A63 C0E004             <1>  SHL   AL,4
00002A66 80E60F             <1>  AND   DH,0FH                  ; HEAD NUMBER
00002A69 08F0               <1>  OR    AL,DH
                            <1>  ;OR   AL,80H or 20H
00002A6B 0CA0               <1>  OR    AL,80h+20h        ; ECC AND 512 BYTE SECTORS
00002A6D 8845FD             <1>  MOV   [CMD_BLOCK+5],AL  ; ECC/SIZE/DRIVE/HEAD
                            <1> su4:
                            <1>  ;POP  ES ; **
                            <1>         ;; 14/02/2015
                            <1>         ;;POP   AX
                            <1>         ;;MOV   [CMD_BLOCK+1],AL        ; SECTOR COUNT
                            <1>         ;;PUSH  AX
                            <1>         ;;MOV   AL,AH                   ; GET INTO LOW BYTE
                            <1>         ;;XOR   AH,AH                   ; ZERO HIGH BYTE
                            <1>         ;;SAL   AX,1                    ; *2 FOR TABLE LOOKUP
00002A70 6658               <1>         pop     ax ; ***
00002A72 8845F9             <1>         mov     [CMD_BLOCK+1], al
00002A75 29DB               <1>         sub  ebx, ebx
00002A77 88E3               <1>  mov     bl, ah
                            <1>         ;xor     bh, bh
                            <1>         ;sal     bx, 1
00002A79 66C1E302           <1>         sal  bx, 2 ; 32 bit offset (21/02/2015)
                            <1>  ;;MOV   SI,AX                   ; PUT INTO SI FOR BRANCH
                            <1>         ;;CMP   AX,M1L                  ; TEST WITHIN RANGE
                            <1>         ;;JNB   short BAD_COMMAND_POP
                            <1>         ;cmp     bx, M1L
00002A7D 83FB74             <1>  cmp   ebx, M1L
00002A80 7308               <1>  jnb   short BAD_COMMAND
                            <1>         ;xchg    bx, si
00002A82 87DE               <1>         xchg ebx, esi
                            <1>  ;;;POP      AX                ; RESTORE AX
                            <1>  ;;;POP      BX                ; AND DATA ADDRESS
                            <1>
                            <1>  ;;PUSH      CX
                            <1>  ;;PUSH      AX                ; ADJUST ES:BX
                            <1>  ;MOV  CX,BX             ; GET 3 HIGH ORDER NIBBLES OF BX
                            <1>  ;SHR  CX,4
                            <1>  ;MOV  AX,ES
                            <1>  ;ADD  AX,CX
                            <1>  ;MOV  ES,AX
                            <1>  ;AND  BX,000FH          ; ES:BX CHANGED TO ES:000X
                            <1>  ;;POP AX
                            <1>  ;;POP CX
                            <1>  ;;JMP word [CS:SI+M1]
                            <1>  ;jmp  word [SI+M1]
00002A84 FFA6[D2280000]     <1>  jmp   dword [esi+M1]
                            <1> ;;BAD_COMMAND_POP:
                            <1> ;;      POP   AX
                            <1> ;;      POP   BX
                            <1> BAD_COMMAND:
00002A8A C605[3F710000]01   <1>         MOV     byte [DISK_STATUS1],BAD_CMD  ; COMMAND ERROR
00002A91 B000               <1>  MOV   AL,0
00002A93 C3                 <1>  RETn
                            <1>
                            <1> ;----------------------------------------
                            <1> ; RESET THE DISK SYSTEM  (AH=00H) :
                            <1> ;----------------------------------------
                            <1>
                            <1> ; 18-1-2015 : one controller reset (not other one)
                            <1>
                            <1> DISK_RESET:
00002A94 FA                 <1>  CLI
00002A95 E4A1               <1>  IN    AL,INTB01         ; GET THE MASK REGISTER
                            <1>  ;JMP  $+2
                            <1>  IODELAY
00002A97 EB00               <2>  jmp short $+2
00002A99 EB00               <2>  jmp short $+2
                            <1>  ;AND  AL,0BFH           ; ENABLE FIXED DISK INTERRUPT
00002A9B 243F               <1>  and   al,3Fh                 ; 22/12/2014 (IRQ 14 & IRQ 15)
00002A9D E6A1               <1>  OUT   INTB01,AL
00002A9F FB                 <1>  STI                     ; START INTERRUPTS
                            <1>  ; 14/02/2015
00002AA0 6689D7             <1>  mov   di, dx
                            <1>  ; 04/01/2015
                            <1>  ;xor  di,di
                            <1> drst0:
00002AA3 B004               <1>  MOV   AL,04H  ; bit 2 - SRST
                            <1>  ;MOV  DX,HF_REG_PORT
00002AA5 668B15[566B0000]   <1>  MOV   DX,[HF_REG_PORT]
00002AAC EE                 <1>  OUT   DX,AL             ; RESET
                            <1> ;MOV   CX,10             ; DELAY COUNT
                            <1> ;DRD:  DEC   CX
                            <1> ;JNZ   short DRD         ; WAIT 4.8 MICRO-SEC
                            <1>  ;mov  cx,2              ; wait for 30 micro seconds
00002AAD B902000000         <1>         mov  ecx, 2 ; 21/02/2015
00002AB2 E875EBFFFF         <1>  call    WAITF                    ; (Award Bios 1999 - WAIT_REFRESH,
                            <1>                                           ; 40 micro seconds)
00002AB7 A0[41710000]       <1>  mov   al,[CONTROL_BYTE]
00002ABC 240F               <1>  AND   AL,0FH                 ; SET HEAD OPTION
00002ABE EE                 <1>  OUT   DX,AL             ; TURN RESET OFF
00002ABF E80E040000         <1>  CALL  NOT_BUSY
00002AC4 7515               <1>  JNZ   short DRERR       ; TIME OUT ON RESET
00002AC6 668B15[546B0000]   <1>  MOV   DX,[HF_PORT]
00002ACD FEC2               <1>  inc   dl  ; HF_PORT+1
                            <1>  ; 02/01/2015 - Award BIOS 1999 - AHDSK.ASM
                            <1>         ;mov     cl, 10
00002ACF B90A000000         <1>         mov     ecx, 10 ; 21/02/2015
                            <1> drst1:
00002AD4 EC                 <1>  IN    AL,DX             ; GET RESET STATUS
00002AD5 3C01               <1>  CMP   AL,1
                            <1>  ; 04/01/2015
00002AD7 740A               <1>  jz    short drst2
                            <1>  ;JNZ  short DRERR       ; BAD RESET STATUS
                            <1>             ; Drive/Head Register - bit 4
00002AD9 E2F9               <1>  loop  drst1
                            <1> DRERR:
00002ADB C605[3F710000]05   <1>  MOV   byte [DISK_STATUS1],BAD_RESET ; CARD FAILED
00002AE2 C3                 <1>  RETn
                            <1> drst2:
                            <1>  ; 14/02/2015
00002AE3 6689FA             <1>  mov   dx,di
                            <1> ;drst3:
                            <1> ;; 05/01/2015
                            <1> ;shl   di,1
                            <1> ;; 04/01/2015
                            <1> ;mov   ax,[di+hd_cports]
                            <1> ;cmp   ax,[HF_REG_PORT]
                            <1> ;je    short drst4
                            <1> ;mov   [HF_REG_PORT], ax
                            <1> ;; 03/01/2015
                            <1> ;mov   ax,[di+hd_ports]
                            <1> ;       mov     [HF_PORT], ax
                            <1> ;; 05/01/2014
                            <1> ;shr   di,1
                            <1> ;; 04/01/2015
                            <1> ;jmp   short drst0 ; reset other controller
                            <1> ;drst4:
                            <1> ;; 05/01/2015
                            <1> ;shr   di,1
                            <1> ;mov   al,[di+hd_dregs]
                            <1> ;and   al,10h ; bit 4 only
                            <1> ;shr   al,4 ; bit 4  -> bit 0
                            <1> ;mov   [hf_m_s], al ; (0 = master, 1 = slave)
                            <1>  ;
```

```
8999 00002AE6 A0[586B0000]        <1>  mov   al, [hf_m_s] ; 18/01/2015
9000 00002AEB A801                <1>  test  al,1
9001                              <1> ;jnz   short drst6
9002 00002AED 7516                <1>         jnz     short drst4
9003 00002AEF 8065FDEF            <1>  AND     byte [CMD_BLOCK+5],0EFH ; SET TO DRIVE 0
9004                              <1> ;drst5:
9005                              <1> drst3:
9006 00002AF3 E811010000          <1>  CALL  INIT_DRV          ; SET MAX HEADS
9007                              <1>  ;mov  dx,di
9008 00002AF8 E8C9010000          <1>  CALL  HDISK_RECAL       ; RECAL TO RESET SEEK SPEED
9009                              <1>  ; 04/01/2014
9010                              <1> ;inc   di
9011                              <1> ;mov   dx,di
9012                              <1> ;cmp   dl,[HF_NUM]
9013                              <1> ;jb    short drst3
9014                              <1> ;DRE:
9015 00002AFD C605[3F710000]00    <1>  MOV   byte [DISK_STATUS1],0   ; IGNORE ANY SET UP ERRORS
9016 00002B04 C3                  <1>  RETn
9017                              <1> ;drst6:
9018                              <1> drst4:       ; Drive/Head Register - bit 4
9019 00002B05 804DFD10            <1>  OR      byte [CMD_BLOCK+5],010H ; SET TO DRIVE 1
9020                              <1>         ;jmp    short drst5
9021 00002B09 EBE8                <1>         jmp     short drst3
9022                              <1>
9023                              <1> ;----------------------------------------
9024                              <1> ;DISK STATUS ROUTINE  (AH = 01H) :
9025                              <1> ;----------------------------------------
9026                              <1>
9027                              <1> RETURN_STATUS:
9028 00002B0B A0[3F710000]        <1>  MOV   AL,[DISK_STATUS1] ; OBTAIN PREVIOUS STATUS
9029 00002B10 C605[3F710000]00    <1>         MOV     byte [DISK_STATUS1],0   ; RESET STATUS
9030 00002B17 C3                  <1>  RETn
9031                              <1>
9032                              <1> ;----------------------------------------
9033                              <1> ;DISK READ ROUTINE    (AH = 02H) :
9034                              <1> ;----------------------------------------
9035                              <1>
9036                              <1> DISK_READ:
9037 00002B18 C645FE20            <1>  MOV   byte [CMD_BLOCK+6],READ_CMD
9038 00002B1C E930020000          <1>         JMP     COMMANDI
9039                              <1>
9040                              <1> ;----------------------------------------
9041                              <1> ;DISK WRITE ROUTINE   (AH = 03H) :
9042                              <1> ;----------------------------------------
9043                              <1>
9044                              <1> DISK_WRITE:
9045 00002B21 C645FE30            <1>  MOV   byte [CMD_BLOCK+6],WRITE_CMD
9046 00002B25 E97C020000          <1>         JMP     COMMANDO
9047                              <1>
9048                              <1> ;----------------------------------------
9049                              <1> ;DISK VERIFY      (AH = 04H) :
9050                              <1> ;----------------------------------------
9051                              <1>
9052                              <1> DISK_VERF:
9053 00002B2A C645FE40            <1>  MOV   byte [CMD_BLOCK+6],VERIFY_CMD
9054 00002B2E E8EA020000          <1>  CALL  COMMAND
9055 00002B33 750C                <1>  JNZ   short VERF_EXIT         ; CONTROLLER STILL BUSY
9056 00002B35 E85C030000          <1>  CALL  _WAIT             ; (Original: CALL WAIT)
9057 00002B3A 7505                <1>  JNZ   short VERF_EXIT         ; TIME OUT
9058 00002B3C E8E9030000          <1>  CALL  CHECK_STATUS
9059                              <1> VERF_EXIT:
9060 00002B41 C3                  <1>  RETn
9061                              <1>
9062                              <1> ;----------------------------------------
9063                              <1> ;FORMATTING       (AH = 05H) :
9064                              <1> ;----------------------------------------
9065                              <1>
9066                              <1> FMT_TRK:                     ; FORMAT TRACK    (AH = 005H)
9067 00002B42 C645FE50            <1>  MOV   byte [CMD_BLOCK+6],FMTTRK_CMD
9068                              <1>  ;PUSH ES
9069                              <1>  ;PUSH BX
9070 00002B46 53                  <1>  push  ebx
9071 00002B47 E88E040000          <1>  CALL  GET_VEC           ; GET DISK PARAMETERS ADDRESS
9072                              <1>  ;MOV  AL,[ES:BX+14]           ; GET SECTORS/TRACK
9073 00002B4C 8A430E              <1>  mov   al, [ebx+14]
9074 00002B4F 8845F9              <1>  MOV   [CMD_BLOCK+1],AL  ; SET SECTOR COUNT IN COMMAND
9075 00002B52 5B                  <1>  pop   ebx
9076                              <1>  ;POP  BX
9077                              <1>  ;POP  ES
9078 00002B53 E955020000          <1>         JMP     CMD_OF                  ; GO EXECUTE THE COMMAND
9079                              <1>
9080                              <1> ;----------------------------------------
9081                              <1> ;READ DASD TYPE         (AH = 15H) :
9082                              <1> ;----------------------------------------
9083                              <1>
9084                              <1> READ_DASD_TYPE:
9085                              <1> READ_D_T:                    ; GET DRIVE PARAMETERS
9086 00002B58 1E                  <1>  PUSH  DS                ; SAVE REGISTERS
9087                              <1>  ;PUSH ES
9088 00002B59 53                  <1>  PUSH  eBX
9089                              <1>  ;CALL DDS               ; ESTABLISH ADDRESSING
9090                              <1>  ;push cs
```

```
                             <1>  ;pop  ds
00002B5A 66BB1000            <1>         mov  bx, KDATA
00002B5E 8EDB                <1>  mov   ds, bx
                             <1>  ;mov  es, bx
00002B60 C605[3F710000]00    <1>  MOV     byte [DISK_STATUS1],0
00002B67 8A1D[40710000]      <1>  MOV   BL,[HF_NUM]        ; GET NUMBER OF DRIVES
00002B6D 80E27F              <1>  AND   DL,7FH                  ; GET DRIVE NUMBER
00002B70 38D3                <1>  CMP   BL,DL
00002B72 7625                <1>  JBE   short RDT_NOT_PRESENT   ; RETURN DRIVE NOT PRESENT
00002B74 E861040000          <1>  CALL  GET_VEC            ; GET DISK PARAMETER ADDRESS
                             <1>  ;MOV  AL,[ES:BX+2]            ; HEADS
00002B79 8A4302              <1>  mov   al, [ebx+2]
                             <1>  ;MOV  CL,[ES:BX+14]
00002B7C 8A4B0E              <1>  mov   cl, [ebx+14]
00002B7F F6E9                <1>  IMUL  CL                 ; * NUMBER OF SECTORS
                             <1>  ;MOV  CX,[ES:BX]          ; MAX NUMBER OF CYLINDERS
00002B81 668B0B              <1>  mov   cx ,[ebx]
                             <1>  ;
                             <1>  ; 02/01/2015
                             <1>  ; ** leave the last cylinder as reserved for diagnostics **
                             <1>  ; (Also in Award BIOS - 1999, AHDSK.ASM, FUN15 -> sub ax, 1)
00002B84 6649                <1>  DEC   CX                 ; LEAVE ONE FOR DIAGNOSTICS
                             <1>  ;
00002B86 66F7E9              <1>  IMUL  CX                 ; NUMBER OF SECTORS
00002B89 6689D1              <1>  MOV   CX,DX              ; HIGH ORDER HALF
00002B8C 6689C2              <1>  MOV   DX,AX              ; LOW ORDER HALF
                             <1>  ;SUB  AX,AX
00002B8F 28C0                <1>  sub   al, al
00002B91 B403                <1>  MOV   AH,03H                  ; INDICATE FIXED DISK
00002B93 5B                  <1> RDT2:  POP   eBX               ; RESTORE REGISTERS
                             <1>  ;POP  ES
00002B94 1F                  <1>  POP   DS
00002B95 F8                  <1>  CLC                      ; CLEAR CARRY
                             <1>  ;RETf 2
00002B96 CA0400              <1>  retf  4
                             <1> RDT_NOT_PRESENT:
00002B99 6629C0              <1>  SUB   AX,AX              ; DRIVE NOT PRESENT RETURN
00002B9C 6689C1              <1>  MOV   CX,AX              ; ZERO BLOCK COUNT
00002B9F 6689C2              <1>  MOV   DX,AX
00002BA2 EBEF                <1>  JMP   short RDT2
                             <1>
                             <1> ;-----------------------------------------
                             <1> ;GET PARAMETERS          (AH = 08H) :
                             <1> ;-----------------------------------------
                             <1>
                             <1> GET_PARM_N:
                             <1> ;GET_PARM:                     ; GET DRIVE PARAMETERS
00002BA4 1E                  <1>  PUSH  DS                 ; SAVE REGISTERS
                             <1>  ;PUSH ES
00002BA5 53                  <1>  PUSH  eBX
                             <1>  ;MOV  AX,ABS0            ; ESTABLISH ADDRESSING
                             <1>  ;MOV  DS,AX
                             <1>  ;TEST DL,1               ; CHECK FOR DRIVE 1
                             <1>  ;JZ   short G0
                             <1>  ;LES  BX,@HF1_TBL_VEC
                             <1>  ;JMP  SHORT G1
                             <1> ;G0:   LES   BX,@HF_TBL_VEC
                             <1> ;G1:
                             <1>  ;CALL DDS                ; ESTABLISH SEGMENT
                             <1>  ; 22/12/2014
                             <1>  ;push cs
                             <1>  ;pop  ds
00002BA6 66BB1000            <1>  mov   bx, KDATA
00002BAA 8EDB                <1>  mov   ds, bx
                             <1>  ;mov  es, bx
                             <1>  ;
00002BAC 80EA80              <1>  SUB   DL,80H
00002BAF 80FA04              <1>  CMP   DL,MAX_FILE        ; TEST WITHIN RANGE
00002BB2 7341                <1>  JAE   short G4
                             <1>  ;
00002BB4 31DB                <1>  xor   ebx, ebx ; 21/02/2015
                             <1>  ; 22/12/2014
00002BB6 88D3                <1>  mov   bl, dl
                             <1>  ;xor  bh, bh
00002BB8 C0E302              <1>  shl   bl, 2              ; convert index to offset
                             <1>  ;add  bx, HF_TBL_VEC
00002BBB 81C3[44710000]      <1>  add   ebx, HF_TBL_VEC
                             <1>  ;mov  ax, [bx+2]
                             <1>  ;mov  es, ax                  ; dpt segment
                             <1>  ;mov  bx, [bx]           ; dpt offset
00002BC1 8B1B                <1>  mov   ebx, [ebx] ; 32 bit offset
                             <1>
00002BC3 C605[3F710000]00    <1>  MOV   byte [DISK_STATUS1],0
                             <1>         ;MOV     AX,[ES:BX]               ; MAX NUMBER OF CYLINDERS
00002BCA 668B03              <1>  mov   ax, [ebx]
                             <1>  ;;SUB AX,2               ; ADJUST FOR 0-N
00002BCD 6648                <1>  dec   ax                 ; max. cylinder number
00002BCF 88C5                <1>  MOV   CH,AL
00002BD1 66250003            <1>  AND   AX,0300H           ; HIGH TWO BITS OF CYLINDER
00002BD5 66D1E8              <1>  SHR   AX,1
00002BD8 66D1E8              <1>  SHR   AX,1
                             <1>  ;OR   AL,[ES:BX+14]           ; SECTORS
```

```
00002BDB 0A430E              <1>  or    al, [ebx+14]
00002BDE 88C1                <1>  MOV   CL,AL
                             <1>  ;MOV  DH,[ES:BX+2]            ; HEADS
00002BE0 8A7302              <1>  mov   dh, [ebx+2]
00002BE3 FECE                <1>  DEC   DH                ; 0-N RANGE
00002BE5 8A15[40710000]      <1>  MOV   DL,[HF_NUM]       ; DRIVE COUNT
00002BEB 6629C0              <1>  SUB   AX,AX
                             <1>         ;27/12/2014
                             <1>  ; ES:DI = Address of disk parameter table from BIOS
                             <1>  ;(Programmer's Guide to the AMIBIOS - 1993)
                             <1>  ;mov  di, bx                  ; HDPT offset
00002BEE 89DF                <1>  mov   edi, ebx
                             <1> G5:
00002BF0 5B                  <1>  POP   eBX               ; RESTORE REGISTERS
                             <1>  ;POP  ES
00002BF1 1F                  <1>  POP   DS
                             <1>  ;RETf 2
00002BF2 CA0400              <1>  retf  4
                             <1> G4:
00002BF5 C605[3F710000]07    <1>  MOV     byte [DISK_STATUS1],INIT_FAIL ; OPERATION FAILED
00002BFC B407                <1>  MOV   AH,INIT_FAIL
00002BFE 28C0                <1>  SUB   AL,AL
00002C00 6629D2              <1>  SUB   DX,DX
00002C03 6629C9              <1>  SUB   CX,CX
00002C06 F9                  <1>  STC                     ; SET ERROR FLAG
00002C07 EBE7                <1>  JMP   short G5
                             <1>
                             <1> ;----------------------------------------
                             <1> ;INITIALIZE DRIVE     (AH = 09H) :
                             <1> ;----------------------------------------
                             <1>  ; 03/01/2015
                             <1>  ; According to ATA-ATAPI specification v2.0 to v5.0
                             <1>  ; logical sector per logical track
                             <1>  ; and logical heads - 1 would be set but
                             <1>  ; it is seen as it will be good
                             <1>  ; if physical parameters will be set here
                             <1>  ; because, number of heads <= 16.
                             <1>  ; (logical heads usually more than 16)
                             <1>  ; NOTE: ATA logical parameters (software C, H, S)
                             <1>  ;     == INT 13h physical parameters
                             <1>
                             <1> ;INIT_DRV:
                             <1> ;MOV   byte [CMD_BLOCK+6],SET_PARM_CMD
                             <1> ;CALL  GET_VEC           ; ES:BX -> PARAMETER BLOCK
                             <1> ;MOV   AL,[ES:BX+2]            ; GET NUMBER OF HEADS
                             <1> ;DEC   AL                ; CONVERT TO 0-INDEX
                             <1> ;MOV   AH,[CMD_BLOCK+5]  ; GET SDH REGISTER
                             <1> ;AND   AH,0F0H           ; CHANGE HEAD NUMBER
                             <1> ;OR    AH,AL             ; TO MAX HEAD
                             <1> ;MOV   [CMD_BLOCK+5],AH
                             <1> ;MOV   AL,[ES:BX+14]           ; MAX SECTOR NUMBER
                             <1> ;MOV   [CMD_BLOCK+1],AL
                             <1> ;SUB   AX,AX
                             <1> ;MOV   [CMD_BLOCK+3],AL  ; ZERO FLAGS
                             <1> ;CALL  COMMAND           ; TELL CONTROLLER
                             <1> ;JNZ   short INIT_EXIT         ; CONTROLLER BUSY ERROR
                             <1> ;CALL  NOT_BUSY          ; WAIT FOR IT TO BE DONE
                             <1> ;JNZ   short INIT_EXIT         ; TIME OUT
                             <1> ;CALL  CHECK_STATUS
                             <1> ;INIT_EXIT:
                             <1> ;RETn
                             <1>
                             <1> ; 04/01/2015
                             <1> ; 02/01/2015 - Derived from from AWARD BIOS 1999
                             <1> ;                   AHDSK.ASM - INIT_DRIVE
                             <1> INIT_DRV:
                             <1>  ;xor  ah,ah
00002C09 31C0                <1>  xor   eax, eax ; 21/02/2015
00002C0B B00B                <1>  mov   al,11 ; Physical heads from translated HDPT
00002C0D 3825[54710000]      <1>         cmp     [LBAMode], ah   ; 0
00002C13 7702                <1>  ja    short idrv0
00002C15 B002                <1>  mov   al,2  ; Physical heads from standard HDPT
                             <1> idrv0:
                             <1>  ; DL = drive number (0 based)
00002C17 E8BE030000          <1>  call  GET_VEC
                             <1>  ;push bx
00002C1C 53                  <1>  push  ebx ; 21/02/2015
                             <1>  ;add  bx,ax
00002C1D 01C3                <1>  add   ebx, eax
                             <1>  ;; 05/01/2015
00002C1F 8A25[586B0000]      <1>  mov   ah, [hf_m_s] ; drive number (0= master, 1= slave)
                             <1>  ;;and       ah,1
00002C25 C0E404              <1>  shl   ah,4
00002C28 80CCA0              <1>  or    ah,0A0h  ; Drive/Head register - 10100000b (A0h)
                             <1>  ;mov  al,[es:bx]
00002C2B 8A03                <1>  mov   al, [ebx] ; 21/02/2015
00002C2D FEC8                <1>  dec   al     ; last head number
                             <1>  ;and  al,0Fh
00002C2F 08E0                <1>  or    al,ah  ; lower 4 bits for head number
                             <1>  ;
00002C31 C645FE91            <1>  mov   byte [CMD_BLOCK+6],SET_PARM_CMD
00002C35 8845FD              <1>  mov   [CMD_BLOCK+5],al
```

```
9275                                <1>  ;pop  bx
9276 00002C38 5B                    <1>  pop   ebx
9277 00002C39 29C0                  <1>  sub   eax, eax ; 21/02/2015
9278 00002C3B B004                  <1>  mov   al,4 ; Physical sec per track from translated HDPT
9279 00002C3D 803D[54710000]00      <1>  cmp   byte [LBAMode], 0
9280 00002C44 7702                  <1>  ja    short idrv1
9281 00002C46 B00E                  <1>  mov   al,14 ; Physical sec per track from standard HDPT
9282                                <1> idrv1:
9283                                <1>  ;xor  ah,ah
9284                                <1>  ;add  bx,ax
9285 00002C48 01C3                  <1>  add   ebx, eax ; 21/02/2015
9286                                <1>  ;mov  al,[es:bx]
9287                                <1>              ; sector number
9288 00002C4A 8A03                  <1>  mov   al, [ebx]
9289 00002C4C 8845F9                <1>  mov   [CMD_BLOCK+1],al
9290 00002C4F 28C0                  <1>  sub   al,al
9291 00002C51 8845FB                <1>  mov   [CMD_BLOCK+3],al  ; ZERO FLAGS
9292 00002C54 E8C4010000            <1>  call  COMMAND       ; TELL CONTROLLER
9293 00002C59 750C                  <1>  jnz   short INIT_EXIT     ; CONTROLLER BUSY ERROR
9294 00002C5B E872020000            <1>  call  NOT_BUSY      ; WAIT FOR IT TO BE DONE
9295 00002C60 7505                  <1>  jnz   short INIT_EXIT     ; TIME OUT
9296 00002C62 E8C3020000            <1>  call  CHECK_STATUS
9297                                <1> INIT_EXIT:
9298 00002C67 C3                    <1>  RETn
9299                                <1>
9300                                <1> ;----------------------------------------
9301                                <1> ; READ LONG        (AH = 0AH) :
9302                                <1> ;----------------------------------------
9303                                <1>
9304                                <1> RD_LONG:
9305                                <1>  ;MOV  @CMD_BLOCK+6,READ_CMD OR ECC_MODE
9306 00002C68 C645FE22              <1>         mov     byte [CMD_BLOCK+6],READ_CMD + ECC_MODE
9307 00002C6C E9E0000000            <1>         JMP     COMMANDI
9308                                <1>
9309                                <1> ;----------------------------------------
9310                                <1> ; WRITE LONG       (AH = 0BH) :
9311                                <1> ;----------------------------------------
9312                                <1>
9313                                <1> WR_LONG:
9314                                <1>  ;MOV  @CMD_BLOCK+6,WRITE_CMD OR ECC_MODE
9315 00002C71 C645FE32              <1>         MOV     byte [CMD_BLOCK+6],WRITE_CMD + ECC_MODE
9316 00002C75 E92C010000            <1>         JMP     COMMANDO
9317                                <1>
9318                                <1> ;----------------------------------------
9319                                <1> ; SEEK             (AH = 0CH) :
9320                                <1> ;----------------------------------------
9321                                <1>
9322                                <1> DISK_SEEK:
9323 00002C7A C645FE70              <1>         MOV     byte [CMD_BLOCK+6],SEEK_CMD
9324 00002C7E E89A010000            <1>  CALL  COMMAND
9325 00002C83 751C                  <1>  JNZ   short DS_EXIT          ; CONTROLLER BUSY ERROR
9326 00002C85 E80C020000            <1>  CALL  _WAIT
9327 00002C8A 7515                  <1>         JNZ     DS_EXIT                 ; TIME OUT ON SEEK
9328 00002C8C E899020000            <1>  CALL  CHECK_STATUS
9329 00002C91 803D[3F710000]40      <1>         CMP     byte [DISK_STATUS1],BAD_SEEK
9330 00002C98 7507                  <1>  JNE   short DS_EXIT
9331 00002C9A C605[3F710000]00      <1>         MOV     byte [DISK_STATUS1],0
9332                                <1> DS_EXIT:
9333 00002CA1 C3                    <1>  RETn
9334                                <1>
9335                                <1> ;----------------------------------------
9336                                <1> ; TEST DISK READY      (AH = 10H) :
9337                                <1> ;----------------------------------------
9338                                <1>
9339                                <1> TST_RDY:                        ; WAIT FOR CONTROLLER
9340 00002CA2 E82B020000            <1>  CALL  NOT_BUSY
9341 00002CA7 751C                  <1>  JNZ   short TR_EX
9342 00002CA9 8A45FD                <1>  MOV   AL,[CMD_BLOCK+5]  ; SELECT DRIVE
9343 00002CAC 668B15[546B0000]      <1>  MOV   DX,[HF_PORT]
9344 00002CB3 80C206                <1>  add   dl,6
9345 00002CB6 EE                    <1>  OUT   DX,AL
9346 00002CB7 E886020000            <1>  CALL  CHECK_ST          ; CHECK STATUS ONLY
9347 00002CBC 7507                  <1>  JNZ   short TR_EX
9348 00002CBE C605[3F710000]00      <1>  MOV   byte [DISK_STATUS1],0   ; WIPE OUT DATA CORRECTED ERROR
9349                                <1> TR_EX:
9350 00002CC5 C3                    <1>  RETn
9351                                <1>
9352                                <1> ;----------------------------------------
9353                                <1> ; RECALIBRATE      (AH = 11H) :
9354                                <1> ;----------------------------------------
9355                                <1>
9356                                <1> HDISK_RECAL:
9357 00002CC6 C645FE10              <1>         MOV     byte [CMD_BLOCK+6],RECAL_CMD ; 10h, 16
9358 00002CCA E84E010000            <1>  CALL  COMMAND          ; START THE OPERATION
9359 00002CCF 7523                  <1>  JNZ   short RECAL_EXIT  ; ERROR
9360 00002CD1 E8C0010000            <1>  CALL  _WAIT            ; WAIT FOR COMPLETION
9361 00002CD6 7407                  <1>  JZ    short RECAL_X          ; TIME OUT ONE OK ?
9362 00002CD8 E8B9010000            <1>  CALL  _WAIT            ; WAIT FOR COMPLETION LONGER
9363 00002CDD 7515                  <1>  JNZ   short RECAL_EXIT  ; TIME OUT TWO TIMES IS ERROR
9364                                <1> RECAL_X:
9365 00002CDF E846020000            <1>  CALL  CHECK_STATUS
9366 00002CE4 803D[3F710000]40      <1>  CMP   byte [DISK_STATUS1],BAD_SEEK ; SEEK NOT COMPLETE
```

```
00002CEB 7507                <1>  JNE   short RECAL_EXIT  ; IS OK
00002CED C605[3F710000]00    <1>  MOV   byte [DISK_STATUS1],0
                             <1> RECAL_EXIT:
00002CF4 803D[3F710000]00    <1>        CMP     byte [DISK_STATUS1],0
00002CFB C3                  <1>  RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ;      CONTROLLER DIAGNOSTIC (AH = 14H) :
                             <1> ;----------------------------------------
                             <1>
                             <1> CTLR_DIAGNOSTIC:
00002CFC FA                  <1>        CLI                         ; DISABLE INTERRUPTS WHILE CHANGING MASK
00002CFD E4A1                <1>  IN    AL,INTB01         ; TURN ON SECOND INTERRUPT CHIP
                             <1>  ;AND  AL,0BFH
00002CFF 243F                <1>  and   al, 3Fh                 ; enable IRQ 14 & IRQ 15
                             <1>  ;JMP  $+2
                             <1>  IODELAY
00002D01 EB00                <2>  jmp short $+2
00002D03 EB00                <2>  jmp short $+2
00002D05 E6A1                <1>  OUT   INTB01,AL
                             <1>  IODELAY
00002D07 EB00                <2>  jmp short $+2
00002D09 EB00                <2>  jmp short $+2
00002D0B E421                <1>  IN    AL,INTA01         ; LET INTERRUPTS PASS THRU TO
00002D0D 24FB                <1>  AND   AL,0FBH           ;  SECOND CHIP
                             <1>  ;JMP  $+2
                             <1>  IODELAY
00002D0F EB00                <2>  jmp short $+2
00002D11 EB00                <2>  jmp short $+2
00002D13 E621                <1>  OUT   INTA01,AL
00002D15 FB                  <1>  STI
00002D16 E8B7010000          <1>  CALL  NOT_BUSY          ; WAIT FOR CARD
00002D1B 752B                <1>  JNZ   short CD_ERR            ; BAD CARD
                             <1>  ;MOV  DX, HF_PORT+7
00002D1D 668B15[546B0000]    <1>  mov   dx, [HF_PORT]
00002D24 80C207              <1>  add   dl, 7
00002D27 B090                <1>  MOV   AL,DIAG_CMD       ; START DIAGNOSE
00002D29 EE                  <1>  OUT   DX,AL
00002D2A E8A3010000          <1>  CALL  NOT_BUSY          ; WAIT FOR IT TO COMPLETE
00002D2F B480                <1>  MOV   AH,TIME_OUT
00002D31 7517                <1>  JNZ   short CD_EXIT           ; TIME OUT ON DIAGNOSTIC
                             <1>  ;MOV  DX,HF_PORT+1            ; GET ERROR REGISTER
00002D33 668B15[546B0000]    <1>  mov   dx, [HF_PORT]
00002D3A FEC2                <1>  inc   dl
00002D3C EC                  <1>  IN    AL,DX
00002D3D A2[36710000]        <1>  MOV   [HF_ERROR],AL           ; SAVE IT
00002D42 B400                <1>  MOV   AH,0
00002D44 3C01                <1>  CMP   AL,1              ; CHECK FOR ALL OK
00002D46 7402                <1>  JE    SHORT CD_EXIT
00002D48 B420                <1> CD_ERR: MOV  AH,BAD_CNTLR
                             <1> CD_EXIT:
00002D4A 8825[3F710000]      <1>  MOV   [DISK_STATUS1],AH
00002D50 C3                  <1>  RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ; COMMANDI                        :
                             <1> ;REPEATEDLY INPUTS DATA TILL    :
                             <1> ;NSECTOR RETURNS ZERO           :
                             <1> ;----------------------------------------
                             <1> COMMANDI:
00002D51 E85A020000          <1>  CALL  CHECK_DMA         ; CHECK 64K BOUNDARY ERROR
00002D56 724D                <1>  JC    short CMD_ABORT
                             <1>  ;MOV  DI,BX
00002D58 89DF                <1>  mov   edi, ebx ; 21/02/2015
00002D5A E8BE000000          <1>  CALL  COMMAND           ; OUTPUT COMMAND
00002D5F 7544                <1>  JNZ   short CMD_ABORT
                             <1> CMD_I1:
00002D61 E830010000          <1>  CALL  _WAIT             ; WAIT FOR DATA REQUEST INTERRUPT
00002D66 753D                <1>  JNZ   short TM_OUT            ; TIME OUT
                             <1>  ;MOV  CX,256                  ; SECTOR SIZE IN WORDS
00002D68 B900010000          <1>  mov   ecx, 256 ; 21/02/2015
                             <1>  ;MOV  DX,HF_PORT
00002D6D 668B15[546B0000]    <1>  mov   dx,[HF_PORT]
00002D74 FA                  <1>  CLI
00002D75 FC                  <1>  CLD
00002D76 F3666D              <1>  REP   INSW              ; GET THE SECTOR
00002D79 FB                  <1>  STI
00002D7A F645FE02            <1>  TEST  byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL INPUT
00002D7E 7419                <1>  JZ    CMD_I3
00002D80 E87A010000          <1>  CALL  WAIT_DRQ          ; WAIT FOR DATA REQUEST
00002D85 721E                <1>  JC    short TM_OUT
                             <1>  ;MOV  DX,HF_PORT
00002D87 668B15[546B0000]    <1>  mov   dx,[HF_PORT]
                             <1>  ;MOV  CX,4                ; GET ECC BYTES
00002D8E B904000000          <1>  mov   ecx, 4 ; mov cx, 4
00002D93 EC                  <1> CMD_I2: IN   AL,DX
                             <1>  ;MOV  [ES:DI],AL        ; GO SLOW FOR BOARD
00002D94 8807                <1>  mov   [edi], al ; 21/02/2015
00002D96 47                  <1>  INC   eDI
00002D97 E2FA                <1>  LOOP  CMD_I2
00002D99 E88C010000          <1> CMD_I3: CALL CHECK_STATUS
00002D9E 7505                <1>  JNZ   short CMD_ABORT         ; ERROR RETURNED
```

```
00002DA0 FE4DF9                <1>  DEC   byte [CMD_BLOCK+1]      ; CHECK FOR MORE
00002DA3 75BC                  <1>  JNZ   SHORT CMD_I1
                               <1> CMD_ABORT:
00002DA5 C3                    <1> TM_OUT: RETn
                               <1>
                               <1> ;----------------------------------------
                               <1> ; COMMANDO                             :
                               <1> ;REPEATEDLY OUTPUTS DATA TILL  :
                               <1> ;NSECTOR RETURNS ZERO          :
                               <1> ;----------------------------------------
                               <1> COMMANDO:
00002DA6 E805020000            <1>  CALL  CHECK_DMA         ; CHECK 64K BOUNDARY ERROR
00002DAB 72F8                  <1>  JC    short CMD_ABORT
00002DAD 89DE                  <1> CMD_OF: MOV  eSI,eBX ; 21/02/2015
00002DAF E869000000            <1>  CALL  COMMAND           ; OUTPUT COMMAND
00002DB4 75EF                  <1>  JNZ   short CMD_ABORT
00002DB6 E844010000            <1>  CALL  WAIT_DRQ          ; WAIT FOR DATA REQUEST
00002DBB 72E8                  <1>  JC    short TM_OUT                  ; TOO LONG
                               <1> CMD_O1: ;PUSHDS
                               <1>  ;PUSH ES                ; MOVE ES TO DS
                               <1>  ;POP  DS
                               <1>  ;MOV  CX,256                  ; PUT THE DATA OUT TO THE CARD
                               <1>  ;MOV  DX,HF_PORT
                               <1>  ; 01/02/2015
00002DBD 668B15[546B0000]      <1>  mov   dx, [HF_PORT]
                               <1>  ;push es
                               <1>  ;pop  ds
                               <1>  ;mov  cx, 256
00002DC4 B900010000            <1>  mov   ecx, 256 ; 21/02/2015
00002DC9 FA                    <1>  CLI
00002DCA FC                    <1>  CLD
00002DCB F3666F                <1>  REP   OUTSW
00002DCE FB                    <1>  STI
                               <1>  ;POP  DS                ; RESTORE DS
00002DCF F645FE02              <1>  TEST  byte [CMD_BLOCK+6],ECC_MODE ; CHECK FOR NORMAL OUTPUT
00002DD3 7419                  <1>  JZ    short CMD_O3
00002DD5 E825010000            <1>  CALL  WAIT_DRQ          ; WAIT FOR DATA REQUEST
00002DDA 72C9                  <1>  JC    short TM_OUT
                               <1>  ;MOV  DX,HF_PORT
00002DDC 668B15[546B0000]      <1>  mov   dx, [HF_PORT]
                               <1>  ;MOV  CX,4                ; OUTPUT THE ECC BYTES
00002DE3 B904000000            <1>  mov   ecx, 4  ; mov cx, 4
                               <1> CMD_O2: ;MOV AL,[ES:SI]
00002DE8 8A06                  <1>  mov   al, [esi]
00002DEA EE                    <1>  OUT   DX,AL
00002DEB 46                    <1>  INC   eSI
00002DEC E2FA                  <1>  LOOP  CMD_O2
                               <1> CMD_O3:
00002DEE E8A3000000            <1>  CALL  _WAIT             ; WAIT FOR SECTOR COMPLETE INTERRUPT
00002DF3 75B0                  <1>  JNZ   short TM_OUT            ; ERROR RETURNED
00002DF5 E830010000            <1>  CALL  CHECK_STATUS
00002DFA 75A9                  <1>  JNZ   short CMD_ABORT
00002DFC F605[35710000]08      <1>  TEST  byte [HF_STATUS],ST_DRQ ; CHECK FOR MORE
00002E03 75B8                  <1>  JNZ   SHORT CMD_O1
                               <1>  ;MOV  DX,HF_PORT+2            ; CHECK RESIDUAL SECTOR COUNT
00002E05 668B15[546B0000]      <1>  mov   dx, [HF_PORT]
                               <1>  ;add  dl, 2
00002E0C FEC2                  <1>  inc   dl
00002E0E FEC2                  <1>  inc   dl
00002E10 EC                    <1>  IN    AL,DX             ;
00002E11 A8FF                  <1>  TEST  AL,0FFH           ;
00002E13 7407                  <1>  JZ    short CMD_O4                  ; COUNT = 0  OK
00002E15 C605[3F710000]BB      <1>  MOV   byte [DISK_STATUS1],UNDEF_ERR
                               <1>                          ; OPERATION ABORTED - PARTIAL TRANSFER
                               <1> CMD_O4:
00002E1C C3                    <1>  RETn
                               <1>
                               <1> ;--------------------------------------------------------
                               <1> ; COMMAND                                              :
                               <1> ;THIS ROUTINE OUTPUTS THE COMMAND BLOCK         :
                               <1> ; OUTPUT                                               :
                               <1> ;BL = STATUS                                 :
                               <1> ;BH = ERROR REGISTER                                   :
                               <1> ;--------------------------------------------------------
                               <1>
                               <1> COMMAND:
00002E1D 53                    <1>  PUSH  eBX               ; WAIT FOR SEEK COMPLETE AND READY
                               <1>  ;;MOV CX,DELAY_2        ; SET INITIAL DELAY BEFORE TEST
                               <1> COMMAND1:
                               <1>  ;;PUSH      CX                ; SAVE LOOP COUNT
00002E1E E87FFEFFFF            <1>  CALL  TST_RDY           ; CHECK DRIVE READY
                               <1>  ;;POP CX
00002E23 7419                  <1>  JZ    short COMMAND2          ; DRIVE IS READY
00002E25 803D[3F710000]80      <1>         CMP     byte [DISK_STATUS1],TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
                               <1>  ;JZ   short CMD_TIMEOUT
                               <1>  ;;LOOP      COMMAND1          ; KEEP TRYING FOR A WHILE
                               <1>  ;JMP  SHORT COMMAND4          ; ITS NOT GOING TO GET READY
00002E2C 7507                  <1>  jne   short COMMAND4
                               <1> CMD_TIMEOUT:
00002E2E C605[3F710000]20      <1>  MOV   byte [DISK_STATUS1],BAD_CNTLR
                               <1> COMMAND4:
00002E35 5B                    <1>  POP   eBX
```

```
00002E36 803D[3F710000]00    <1>         CMP     byte [DISK_STATUS1],0   ; SET CONDITION CODE FOR CALLER
00002E3D C3                  <1>  RETn
                             <1> COMMAND2:
00002E3E 5B                  <1>  POP   eBX
00002E3F 57                  <1>  PUSH  eDI
00002E40 C605[37710000]00    <1>  MOV   byte [HF_INT_FLAG],0    ; RESET INTERRUPT FLAG
00002E47 FA                  <1>  CLI                     ; INHIBIT INTERRUPTS WHILE CHANGING MASK
00002E48 E4A1                <1>  IN    AL,INTB01         ; TURN ON SECOND INTERRUPT CHIP
                             <1>  ;AND  AL,0BFH
00002E4A 243F                <1>  and   al, 3Fh                 ; Enable IRQ 14 & 15
                             <1>  ;JMP  $+2
                             <1>  IODELAY
00002E4C EB00                <2>  jmp short $+2
00002E4E EB00                <2>  jmp short $+2
00002E50 E6A1                <1>  OUT   INTB01,AL
00002E52 E421                <1>  IN    AL,INTA01         ; LET INTERRUPTS PASS THRU TO
00002E54 24FB                <1>  AND   AL,0FBH           ;  SECOND CHIP
                             <1>  ;JMP  $+2
                             <1>  IODELAY
00002E56 EB00                <2>  jmp short $+2
00002E58 EB00                <2>  jmp short $+2
00002E5A E621                <1>  OUT   INTA01,AL
00002E5C FB                  <1>  STI
00002E5D 31FF                <1>  XOR   eDI,eDI                 ; INDEX THE COMMAND TABLE
                             <1>  ;MOV  DX,HF_PORT+1            ; DISK ADDRESS
00002E5F 668B15[546B0000]    <1>  mov   dx, [HF_PORT]
00002E66 FEC2                <1>  inc   dl
00002E68 F605[41710000]C0    <1>  TEST  byte [CONTROL_BYTE],0C0H ; CHECK FOR RETRY SUPPRESSION
00002E6F 7411                <1>  JZ    short COMMAND3
00002E71 8A45FE              <1>  MOV   AL, [CMD_BLOCK+6]       ; YES-GET OPERATION CODE
00002E74 24F0                <1>  AND   AL,0F0H           ; GET RID OF MODIFIERS
00002E76 3C20                <1>  CMP   AL,20H                  ; 20H-40H IS READ, WRITE, VERIFY
00002E78 7208                <1>  JB    short COMMAND3
00002E7A 3C40                <1>  CMP   AL,40H
00002E7C 7704                <1>  JA    short COMMAND3
00002E7E 804DFE01            <1>  OR    byte [CMD_BLOCK+6],NO_RETRIES
                             <1>                          ; VALID OPERATION FOR RETRY SUPPRESS
                             <1> COMMAND3:
00002E82 8A443DF8            <1>  MOV   AL,[CMD_BLOCK+eDI]      ; GET THE COMMAND STRING BYTE
00002E86 EE                  <1>  OUT   DX,AL             ; GIVE IT TO CONTROLLER
                             <1>  IODELAY
00002E87 EB00                <2>  jmp short $+2
00002E89 EB00                <2>  jmp short $+2
00002E8B 47                  <1>  INC   eDI               ; NEXT BYTE IN COMMAND BLOCK
00002E8C 6642                <1>  INC   DX                ; NEXT DISK ADAPTER REGISTER
00002E8E 6683FF07            <1>  cmp   di, 7 ; 1/1/2015  ; ALL DONE?
00002E92 75EE                <1>  JNZ   short COMMAND3          ; NO--GO DO NEXT ONE
00002E94 5F                  <1>  POP   eDI
00002E95 C3                  <1>  RETn                    ; ZERO FLAG IS SET
                             <1>
                             <1> ;CMD_TIMEOUT:
                             <1> ;MOV   byte [DISK_STATUS1],BAD_CNTLR
                             <1> ;COMMAND4:
                             <1> ;POP   BX
                             <1> ;CMP   [DISK_STATUS1],0  ; SET CONDITION CODE FOR CALLER
                             <1> ;RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ;WAIT FOR INTERRUPT            :
                             <1> ;----------------------------------------
                             <1> ;WAIT:
                             <1> _WAIT:
00002E96 FB                  <1>  STI                     ; MAKE SURE INTERRUPTS ARE ON
                             <1>  ;SUB  CX,CX             ; SET INITIAL DELAY BEFORE TEST
                             <1>  ;CLC
                             <1>  ;MOV  AX,9000H          ; DEVICE WAIT INTERRUPT
                             <1>  ;INT  15H
                             <1>  ;JC   WT2               ; DEVICE TIMED OUT
                             <1>  ;MOV  BL,DELAY_1        ; SET DELAY COUNT
                             <1>
                             <1>  ;mov  bl, WAIT_HDU_INT_HI
                             <1>  ;; 21/02/2015
                             <1>  ;;mov bl, WAIT_HDU_INT_HI + 1
                             <1>  ;;mov cx, WAIT_HDU_INT_LO
00002E97 B915160500          <1>  mov   ecx, WAIT_HDU_INT_LH
                             <1>                          ; (AWARD BIOS -> WAIT_FOR_MEM)
                             <1> ;----- WAIT LOOP
                             <1>
                             <1> WT1:
                             <1>  ;TEST byte [HF_INT_FLAG],80H  ; TEST FOR INTERRUPT
00002E9C F605[37710000]C0    <1>  test  byte [HF_INT_FLAG],0C0h
                             <1>  ;LOOPZ      WT1
00002EA3 7517                <1>  JNZ   short WT3         ; INTERRUPT--LETS GO
                             <1>  ;DEC  BL
                             <1>  ;JNZ  short WT1         ; KEEP TRYING FOR A WHILE
                             <1>
                             <1> WT1_hi:
00002EA5 E461                <1>  in    al, SYS1 ; 61h (PORT_B) ; wait for lo to hi
00002EA7 A810                <1>  test  al, 10h                 ; transition on memory
00002EA9 75FA                <1>  jnz   short WT1_hi            ; refresh.
                             <1> WT1_lo:
00002EAB E461                <1>  in    al, SYS1          ; 061h (PORT_B)
```

```
00002EAD A810                <1>  test  al, 10h
00002EAF 74FA                <1>  jz    short WT1_lo
00002EB1 E2E9                <1>  loop  WT1
                             <1>  ;;or  bl, bl
                             <1>  ;;jz  short WT2
                             <1>  ;;dec bl
                             <1>  ;;jmp short WT1
                             <1>  ;dec  bl
                             <1>  ;jnz  short WT1
                             <1>
00002EB3 C605[3F710000]80    <1> WT2:   MOV   byte [DISK_STATUS1],TIME_OUT ; REPORT TIME OUT ERROR
00002EBA EB0E                <1>  JMP   SHORT WT4
00002EBC C605[3F710000]00    <1> WT3:   MOV   byte [DISK_STATUS1],0
00002EC3 C605[37710000]00    <1>  MOV   byte [HF_INT_FLAG],0
00002ECA 803D[3F710000]00    <1> WT4:   CMP   byte [DISK_STATUS1],0   ; SET CONDITION CODE FOR CALLER
00002ED1 C3                  <1>  RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ; WAIT FOR CONTROLLER NOT BUSY  :
                             <1> ;----------------------------------------
                             <1> NOT_BUSY:
00002ED2 FB                  <1>  STI                     ; MAKE SURE INTERRUPTS ARE ON
                             <1>  ;PUSH eBX
                             <1>  ;SUB  CX,CX             ; SET INITIAL DELAY BEFORE TEST
00002ED3 668B15[546B0000]    <1>  mov   DX, [HF_PORT]
00002EDA 80C207              <1>  add   dl, 7             ; Status port (HF_PORT+7)
                             <1>  ;MOV  BL,DELAY_1
                             <1>                          ; wait for 10 seconds
                             <1>  ;mov  cx, WAIT_HDU_INT_LO     ; 1615h
                             <1>  ;;mov       bl, WAIT_HDU_INT_HI     ;   05h
                             <1>  ;mov  bl, WAIT_HDU_INT_HI + 1
00002EDD B915160500          <1>  mov   ecx, WAIT_HDU_INT_LH  ; 21/02/2015
                             <1>  ;
                             <1> ;;      mov     byte [wait_count], 0    ; Reset wait counter
                             <1> NB1:
00002EE2 EC                  <1>  IN    AL,DX             ; CHECK STATUS
                             <1>  ;TEST AL,ST_BUSY
00002EE3 2480                <1>  and   al, ST_BUSY
                             <1>  ;LOOPNZ     NB1
00002EE5 7410                <1>  JZ    short NB2         ; NOT BUSY--LETS GO
                             <1>  ;DEC  BL
                             <1>  ;JNZ  short NB1         ; KEEP TRYING FOR A WHILE
                             <1>
00002EE7 E461                <1> NB1_hi: IN   AL,SYS1                 ; wait for hi to lo
00002EE9 A810                <1>  TEST  AL,010H                 ; transition on memory
00002EEB 75FA                <1>  JNZ   SHORT NB1_hi            ; refresh.
00002EED E461                <1> NB1_lo: IN   AL,SYS1
00002EEF A810                <1>  TEST  AL,010H
00002EF1 74FA                <1>  JZ    short NB1_lo
00002EF3 E2ED                <1>  LOOP  NB1
                             <1>  ;dec  bl
                             <1>  ;jnz  short NB1
                             <1>  ;
                             <1> ;;      cmp     byte [wait_count], 182  ; 10 seconds (182 timer ticks)
                             <1> ;;      jb    short NB1
                             <1>  ;
                             <1>  ;MOV  [DISK_STATUS1],TIME_OUT ; REPORT TIME OUT ERROR
                             <1>  ;JMP  SHORT NB3
00002EF5 B080                <1>  mov   al, TIME_OUT
                             <1> NB2:
                             <1>  ;MOV  byte [DISK_STATUS1],0
                             <1> ;NB3:
                             <1>  ;POP  eBX
00002EF7 A2[3F710000]        <1>  mov   [DISK_STATUS1], al      ;;; will be set after return
                             <1>  ;CMP  byte [DISK_STATUS1],0   ; SET CONDITION CODE FOR CALLER
00002EFC 08C0                <1>  or    al, al                  ; (zf = 0 --> timeout)
00002EFE C3                  <1>  RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ; WAIT FOR DATA REQUEST        :
                             <1> ;----------------------------------------
                             <1> WAIT_DRQ:
                             <1>  ;MOV  CX,DELAY_3
                             <1>  ;MOV  DX,HF_PORT+7
00002EFF 668B15[546B0000]    <1>  mov   dx, [HF_PORT]
00002F06 80C207              <1>  add   dl, 7
                             <1>  ;;MOV bl, WAIT_HDU_DRQ_HI     ; 0
                             <1>  ;MOV  cx, WAIT_HDU_DRQ_LO     ; 1000 (30 milli seconds)
                             <1>                          ; (but it is written as 2000
                             <1>                          ; micro seconds in ATORGS.ASM file
                             <1>                          ; of Award Bios - 1999, D1A0622)
00002F09 B9E8030000          <1>  mov   ecx, WAIT_HDU_DRQ_LH ; 21/02/2015
00002F0E EC                  <1> WQ_1:  IN    AL,DX             ; GET STATUS
00002F0F A808                <1>  TEST  AL,ST_DRQ         ; WAIT FOR DRQ
00002F11 7516                <1>  JNZ   short WQ_OK
                             <1>  ;LOOP WQ_1              ; KEEP TRYING FOR A SHORT WHILE
                             <1> WQ_hi:
00002F13 E461                <1>  IN    AL,SYS1                 ; wait for hi to lo
00002F15 A810                <1>  TEST  AL,010H                 ; transition on memory
00002F17 75FA                <1>  JNZ   SHORT WQ_hi       ; refresh.
00002F19 E461                <1> WQ_lo:  IN      AL,SYS1
00002F1B A810                <1>  TEST  AL,010H
```

```
00002F1D 74FA                <1>  JZ    SHORT WQ_lo
00002F1F E2ED                <1>  LOOP  WQ_1
                             <1> 
00002F21 C605[3F710000]80    <1>         MOV     byte [DISK_STATUS1],TIME_OUT ; ERROR
00002F28 F9                  <1>  STC
                             <1> WQ_OK:
00002F29 C3                  <1>  RETn
                             <1> ;WQ_OK:;CLC
                             <1> ;RETn
                             <1> 
                             <1> ;----------------------------------------
                             <1> ;CHECK FIXED DISK STATUS       :
                             <1> ;----------------------------------------
                             <1> CHECK_STATUS:
00002F2A E813000000          <1>  CALL  CHECK_ST          ; CHECK THE STATUS BYTE
00002F2F 7509                <1>  JNZ   short CHECK_S1          ; AN ERROR WAS FOUND
00002F31 A801                <1>  TEST  AL,ST_ERROR       ; WERE THERE ANY OTHER ERRORS
00002F33 7405                <1>  JZ    short CHECK_S1          ; NO ERROR REPORTED
00002F35 E847000000          <1>  CALL  CHECK_ER          ; ERROR REPORTED
                             <1> CHECK_S1:
00002F3A 803D[3F710000]00    <1>  CMP   byte [DISK_STATUS1],0   ; SET STATUS FOR CALLER
00002F41 C3                  <1>  RETn
                             <1> 
                             <1> ;----------------------------------------
                             <1> ;CHECK FIXED DISK STATUS BYTE  :
                             <1> ;----------------------------------------
                             <1> CHECK_ST:
                             <1>  ;MOV  DX,HF_PORT+7            ; GET THE STATUS
00002F42 668B15[546B0000]    <1>  mov   dx, [HF_PORT]
00002F49 80C207              <1>  add   dl, 7
00002F4C EC                  <1>  IN    AL,DX
00002F4D A2[35710000]        <1>  MOV   [HF_STATUS],AL
00002F52 B400                <1>  MOV   AH,0
00002F54 A880                <1>  TEST  AL,ST_BUSY        ; IF STILL BUSY
00002F56 751A                <1>  JNZ   short CKST_EXIT         ;  REPORT OK
00002F58 B4CC                <1>  MOV   AH,WRITE_FAULT
00002F5A A820                <1>  TEST  AL,ST_WRT_FLT           ; CHECK FOR WRITE FAULT
00002F5C 7514                <1>  JNZ   short CKST_EXIT
00002F5E B4AA                <1>  MOV   AH,NOT_RDY
00002F60 A840                <1>  TEST  AL,ST_READY       ; CHECK FOR NOT READY
00002F62 740E                <1>  JZ    short CKST_EXIT
00002F64 B440                <1>  MOV   AH,BAD_SEEK
00002F66 A810                <1>  TEST  AL,ST_SEEK_COMPL  ; CHECK FOR SEEK NOT COMPLETE
00002F68 7408                <1>  JZ    short CKST_EXIT
00002F6A B411                <1>  MOV   AH,DATA_CORRECTED
00002F6C A804                <1>  TEST  AL,ST_CORRCTD           ; CHECK FOR CORRECTED ECC
00002F6E 7502                <1>  JNZ   short CKST_EXIT
00002F70 B400                <1>  MOV   AH,0
                             <1> CKST_EXIT:
00002F72 8825[3F710000]      <1>  MOV   [DISK_STATUS1],AH ; SET ERROR FLAG
00002F78 80FC11              <1>  CMP   AH,DATA_CORRECTED ; KEEP GOING WITH DATA CORRECTED
00002F7B 7403                <1>  JZ    short CKST_EX1
00002F7D 80FC00              <1>  CMP   AH,0
                             <1> CKST_EX1:
00002F80 C3                  <1>  RETn
                             <1> 
                             <1> ;----------------------------------------
                             <1> ;CHECK FIXED DISK ERROR REGISTER :
                             <1> ;----------------------------------------
                             <1> CHECK_ER:
                             <1>  ;MOV  DX, HF_PORT+1           ; GET THE ERROR REGISTER
00002F81 668B15[546B0000]    <1>  mov   dx, [HF_PORT]           ;
00002F88 FEC2                <1>  inc   dl
00002F8A EC                  <1>  IN    AL,DX
00002F8B A2[36710000]        <1>  MOV   [HF_ERROR],AL
00002F90 53                  <1>  PUSH  eBX ; 21/02/2015
00002F91 B908000000          <1>  MOV   eCX,8             ; TEST ALL 8 BITS
00002F96 D0E0                <1> CK1:   SHL   AL,1              ; MOVE NEXT ERROR BIT TO CARRY
00002F98 7202                <1>  JC    short CK2         ; FOUND THE ERROR
00002F9A E2FA                <1>  LOOP  CK1               ; KEEP TRYING
00002F9C BB[486B0000]        <1> CK2:   MOV   eBX, ERR_TBL            ; COMPUTE ADDRESS OF
00002FA1 01CB                <1>  ADD   eBX,eCX                 ; ERROR CODE
                             <1>  ;;MOV AH,BYTE [CS:BX]         ; GET ERROR CODE
                             <1>  ;mov  ah, [bx]
00002FA3 8A23                <1>  mov   ah, [ebx] ; 21/02/2015
00002FA5 8825[3F710000]      <1> CKEX:  MOV   [DISK_STATUS1],AH ; SAVE ERROR CODE
00002FAB 5B                  <1>  POP   eBX
00002FAC 80FC00              <1>  CMP   AH,0
00002FAF C3                  <1>  RETn
                             <1> 
                             <1> ;--------------------------------------------------------
                             <1> ; CHECK_DMA                                :
                             <1> ;  -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :
                             <1> ;   FIT WITHOUT SEGMENT OVERFLOW.                :
                             <1> ;  -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X     :
                             <1> ;  -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE)  :
                             <1> ;  -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H)    :
                             <1> ;  -ERROR OTHERWISE                              :
                             <1> ;--------------------------------------------------------
                             <1> CHECK_DMA:
00002FB0 6650                <1>  PUSH  AX                ; SAVE REGISTERS
00002FB2 66B80080            <1>  MOV   AX,8000H          ; AH = MAX # SECTORS AL = MAX OFFSET
```

```
00002FB6 F645FE02            <1>         TEST    byte [CMD_BLOCK+6],ECC_MODE
00002FBA 7404                <1>  JZ    short CKD1
00002FBC 66B8047F            <1>  MOV   AX,7F04H          ; ECC IS 4 MORE BYTES
00002FC0 3A65F9              <1> CKD1:  CMP   AH, [CMD_BLOCK+1]       ; NUMBER OF SECTORS
00002FC3 7706                <1>  JA    short CKDOK       ; IT WILL FIT
00002FC5 7208                <1>  JB    short CKDERR            ; TOO MANY
00002FC7 38D8                <1>  CMP   AL,BL             ; CHECK OFFSET ON MAX SECTORS
00002FC9 7204                <1>  JB    short CKDERR            ; ERROR
00002FCB F8                  <1> CKDOK: CLC                     ; CLEAR CARRY
00002FCC 6658                <1>  POP   AX
00002FCE C3                  <1>  RETn                    ; NORMAL RETURN
00002FCF F9                  <1> CKDERR: STC                    ; INDICATE ERROR
00002FD0 C605[3F710000]09    <1>         MOV     byte [DISK_STATUS1],DMA_BOUNDARY
00002FD7 6658                <1>  POP   AX
00002FD9 C3                  <1>  RETn
                             <1>
                             <1> ;----------------------------------------
                             <1> ;SET UP ES:BX-> DISK PARMS     :
                             <1> ;----------------------------------------
                             <1>
                             <1> ; INPUT -> DL = 0 based drive number
                             <1> ; OUTPUT -> ES:BX = disk parameter table address
                             <1>
                             <1> GET_VEC:
                             <1>  ;SUB  AX,AX             ; GET DISK PARAMETER ADDRESS
                             <1>  ;MOV  ES,AX
                             <1>  ;TEST DL,1
                             <1>  ;JZ   short GV_0
                             <1> ;LES   BX,[HF1_TBL_VEC]  ; ES:BX -> DRIVE PARAMETERS
                             <1> ;JMP   SHORT GV_EXIT
                             <1> ;GV_0:
                             <1> ;LES   BX,[HF_TBL_VEC]         ; ES:BX -> DRIVE PARAMETERS
                             <1> ;
                             <1>  ;xor  bh, bh
00002FDA 31DB                <1>  xor   ebx, ebx
00002FDC 88D3                <1>  mov   bl, dl
                             <1>  ;;02/01/2015
                             <1>  ;;shl bl, 1             ; port address offset
                             <1>  ;;mov ax, [bx+hd_ports] ; Base port address (1F0h, 170h)
                             <1>  ;;shl bl, 1             ; dpt pointer offset
00002FDE C0E302              <1>  shl   bl, 2 ;;
                             <1>  ;add  bx, HF_TBL_VEC          ; Disk parameter table pointer
00002FE1 81C3[44710000]      <1>  add   ebx, HF_TBL_VEC ; 21/02/2015
                             <1>  ;push word [bx+2]       ; dpt segment
                             <1>  ;pop  es
                             <1>  ;mov  bx, [bx]          ; dpt offset
00002FE7 8B1B                <1>  mov   ebx, [ebx]
                             <1> ;GV_EXIT:
00002FE9 C3                  <1>  RETn
                             <1>
                             <1> hdc1_int: ; 21/02/2015
                             <1> ;--- HARDWARE INT 76H -- ( IRQ LEVEL  14 ) ----------------------
                             <1> ;                                          :
                             <1> ;FIXED DISK INTERRUPT ROUTINE                    :
                             <1> ;                                          :
                             <1> ;----------------------------------------------------------------
                             <1>
                             <1> ; 22/12/2014
                             <1> ; IBM PC-XT Model 286 System BIOS Source Code - DISK.ASM (HD_INT)
                             <1> ; '11/15/85'
                             <1> ; AWARD BIOS 1999 (D1A0622)
                             <1> ;Source Code - ATORGS.ASM (INT_HDISK, INT_HDISK1)
                             <1>
                             <1> ;int_76h:
                             <1> HD_INT:
00002FEA 6650                <1>  PUSH  AX
00002FEC 1E                  <1>  PUSH  DS
                             <1>  ;CALL DDS
                             <1>  ; 21/02/2015 (32 bit, 386 pm modification)
00002FED 66B81000            <1>  mov   ax, KDATA
00002FF1 8ED8                <1>  mov   ds, ax
                             <1>  ;
                             <1>  ;;MOV @HF_INT_FLAG,0FFH ; ALL DONE
                             <1>         ;mov     byte [CS:HF_INT_FLAG], 0FFh
00002FF3 C605[37710000]FF    <1>  mov   byte [HF_INT_FLAG], 0FFh
                             <1>  ;
00002FFA 6652                <1>  push  dx
00002FFC 66BAF701            <1>  mov   dx, HDC1_BASEPORT+7     ; Status Register (1F7h)
                             <1>                          ; Clear Controller
                             <1> Clear_IRQ1415:                       ; (Award BIOS - 1999)
00003000 EC                  <1>  in    al, dx                  ;
00003001 665A                <1>  pop   dx
                             <1>  NEWIODELAY
00003003 E6EB                <2>  out 0ebh,al
                             <1>  ;
00003005 B020                <1>  MOV   AL,EOI                  ; NON-SPECIFIC END OF INTERRUPT
00003007 E6A0                <1>  OUT   INTB00,AL         ; FOR CONTROLLER #2
                             <1>  ;JMP  $+2               ; WAIT
                             <1>  NEWIODELAY
00003009 E6EB                <2>  out 0ebh,al
0000300B E620                <1>  OUT   INTA00,AL         ; FOR CONTROLLER #1
0000300D 1F                  <1>  POP   DS
```

```
                              <1>  ;STI                    ; RE-ENABLE INTERRUPTS
                              <1>  ;MOV  AX,9100H          ; DEVICE POST
                              <1>  ;INT  15H               ;  INTERRUPT
                              <1> irq15_iret: ; 25/02/2015
0000300E 6658                 <1>  POP   AX
00003010 CF                   <1>  IRETd                   ; RETURN FROM INTERRUPT
                              <1> 
                              <1> hdc2_int: ; 21/02/2015
                              <1> ;++++ HARDWARE INT 77H ++ ( IRQ LEVEL  15 ) ++++++++++++++++++++
                              <1> ;                                           :
                              <1> ;FIXED DISK INTERRUPT ROUTINE                     :
                              <1> ;                                           :
                              <1> ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                              <1> 
                              <1> ;int_77h:
                              <1> HD1_INT:
00003011 6650                 <1>  PUSH  AX
                              <1>  ; Check if that is a spurious IRQ (from slave PIC)
                              <1>  ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
00003013 B00B                 <1>  mov   al, 0Bh  ; In-Service Register
00003015 E6A0                 <1>  out   0A0h, al
00003017 EB00                 <1>         jmp short $+2
00003019 EB00                 <1>  jmp short $+2
0000301B E4A0                 <1>  in    al, 0A0h
0000301D 2480                 <1>  and   al, 80h ; bit 7 (is it real IRQ 15 or fake?)
0000301F 74ED                 <1>  jz    short irq15_iret ; Fake (spurious)IRQ, do not send EOI)
                              <1>  ;
00003021 1E                   <1>  PUSH  DS
                              <1>  ;CALL DDS
                              <1>  ; 21/02/2015 (32 bit, 386 pm modification)
00003022 66B81000             <1>  mov   ax, KDATA
00003026 8ED8                 <1>  mov   ds, ax
                              <1>  ;
                              <1>  ;;MOV @HF_INT_FLAG,0FFH ; ALL DONE
                              <1>         ;or      byte [CS:HF_INT_FLAG],0C0h
00003028 800D[37710000]C0     <1>  or    byte [HF_INT_FLAG], 0C0h
                              <1>  ;
0000302F 6652                 <1>  push  dx
00003031 66BA7701             <1>  mov   dx, HDC2_BASEPORT+7     ; Status Register (177h)
                              <1>                          ; Clear Controller (Award BIOS 1999)
00003035 EBC9                 <1>  jmp   short Clear_IRQ1415
                              <1> 
                              <1> 
                              <1> ;%include 'diskdata.inc' ; 11/03/2015
                              <1> ;%include 'diskbss.inc' ; 11/03/2015
                              <1> 
                              <1> 
                              <1> ;//////////////////////////////////////////////////////////////////
                              <1> ;; END OF DISK I/O SYTEM ///
                                  %include 'memory.inc'  ; 09/03/2015
                              <1> ; MEMORY.ASM - Retro UNIX 386 v1 MEMORY MANAGEMENT FUNCTIONS (PROCEDURES)
                              <1> ; Retro UNIX 386 v1 Kernel (unix386.s, v0.2.0.14) - MEMORY.INC
                              <1> ; Last Modification: 18/10/2015 (!not completed!)
                              <1> ;
                              <1> ; Source code for NASM - Netwide Assembler (2.11)
                              <1> 
                              <1> ; ///////// MEMORY MANAGEMENT FUNCTIONS (PROCEDURES) //////////////
                              <1> 
                              <1> ;;04/11/2014 (unix386.s)
                              <1> ;PDE_A_PRESENT     equ   1           ; Present flag for PDE
                              <1> ;PDE_A_WRITE equ   2           ; Writable (write permission) flag
                              <1> ;PDE_A_USER  equ   4           ; User (non-system/kernel) page flag
                              <1> ;;
                              <1> ;PTE_A_PRESENT     equ   1           ; Present flag for PTE (bit 0)
                              <1> ;PTE_A_WRITE equ   2           ; Writable (write permission) flag (bit 1)
                              <1> ;PTE_A_USER  equ   4           ; User (non-system/kernel) page flag (bit 2)
                              <1> ;PTE_A_ACCESS   equ      32          ; Accessed flag (bit 5) ; 09/03/2015
                              <1> 
                              <1> ; 27/04/2015
                              <1> ; 09/03/2015
                              <1> PAGE_SIZE    equ 4096          ; page size in bytes
                              <1> PAGE_SHIFT   equ 12                  ; page table shift count
                              <1> PAGE_D_SHIFT equ 22      ; 12 + 10   ; page directory shift count
                              <1> PAGE_OFF     equ 0FFFh         ; 12 bit byte offset in page frame
                              <1> PTE_MASK     equ 03FFh         ; page table entry mask
                              <1> PTE_DUPLICATED  equ 200h       ; duplicated page sign (AVL bit 0)
                              <1> PDE_A_CLEAR  equ 0F000h        ; to clear PDE attribute bits
                              <1> PTE_A_CLEAR  equ 0F000h        ; to clear PTE attribute bits
                              <1> LOGIC_SECT_SIZE equ 512                ; logical sector size
                              <1> ERR_MAJOR_PF equ 0E0h          ; major error: page fault
                              <1> ERR_MINOR_IM equ 1             ; insufficient (out of) memory
                              <1> ERR_MINOR_DSKequ 2             ; disk read/write error
                              <1> ERR_MINOR_PV equ 3             ; protection violation
                              <1> SWP_DISK_READ_ERR equ 4
                              <1> SWP_DISK_NOT_PRESENT_ERR equ 5
                              <1> SWP_SECTOR_NOT_PRESENT_ERR equ 6
                              <1> SWP_NO_FREE_SPACE_ERR equ 7
                              <1> SWP_DISK_WRITE_ERR equ 8
                              <1> SWP_NO_PAGE_TO_SWAP_ERR equ 9
                              <1> PTE_A_ACCESS_BIT equ 5  ; Bit 5 (accessed flag)
                              <1> SECTOR_SHIFT    equ 3   ; sector shift (to convert page block number)
                              <1> 
```

```
<1> ;
<1> ;; Retro Unix 386 v1 - paging method/principles
<1> ;;
<1> ;; 10/10/2014
<1> ;; RETRO UNIX 386 v1 - PAGING METHOD/PRINCIPLES
<1> ;;
<1> ;; KERNEL PAGE MAP: 1 to 1 physical memory page map
<1> ;;     (virtual address = physical address)
<1> ;; KERNEL PAGE TABLES:
<1> ;;     Kernel page directory and all page tables are
<1> ;;     on memory as initialized, as equal to physical memory
<1> ;;     layout. Kernel pages can/must not be swapped out/in.
<1> ;;
<1> ;;     what for: User pages may be swapped out, when accessing
<1> ;;     a page in kernel/system mode, if it would be swapped out,
<1> ;;     kernel would have to swap it in! But it is also may be
<1> ;;     in use by a user process. (In system/kernel mode
<1> ;;     kernel can access all memory pages even if they are
<1> ;;     reserved/allocated for user processes. Swap out/in would
<1> ;;     cause conflicts.)
<1> ;;
<1> ;;     As result of these conditions,
<1> ;;     all kernel pages must be initialized as equal to
<1> ;;     physical layout for preventing page faults.
<1> ;;     Also, calling "allocate page" procedure after
<1> ;;     a page fault can cause another page fault (double fault)
<1> ;;     if all kernel page tables would not be initialized.
<1> ;;
<1> ;;     [first_page] = Beginning of users space, as offset to
<1> ;;     memory allocation table. (double word aligned)
<1> ;;
<1> ;;     [next_page] = first/next free space to be searched
<1> ;;     as offset to memory allocation table. (dw aligned)
<1> ;;
<1> ;;     [last_page] = End of memory (users space), as offset
<1> ;;     to memory allocation table. (double word aligned)
<1> ;;
<1> ;; USER PAGE TABLES:
<1> ;;     Demand paging (& 'copy on write' allocation method) ...
<1> ;;           'ready only' marked copies of the
<1> ;;           parent process's page table entries (for
<1> ;;           same physical memory).
<1> ;;           (A page will be copied to a new page after
<1> ;;            if it causes R/W page fault.)
<1> ;;
<1> ;;     Every user process has own (different)
<1> ;;     page directory and page tables.
<1> ;;
<1> ;;     Code starts at virtual address 0, always.
<1> ;;     (Initial value of EIP is 0 in user mode.)
<1> ;;     (Programs can be written/developed as simple
<1> ;;      flat memory programs.)
<1> ;;
<1> ;; MEMORY ALLOCATION STRATEGY:
<1> ;;     Memory page will be allocated by kernel only
<1> ;;           (in kernel/system mode only).
<1> ;;     * After a
<1> ;;       - 'not present' page fault
<1> ;;       - 'writing attempt on read only page' page fault
<1> ;;     * For loading (opening, reading) a file or disk/drive
<1> ;;     * As responce to 'allocate additional memory blocks'
<1> ;;       request by running process.
<1> ;;     * While creating a process, allocating a new buffer,
<1> ;;       new page tables etc.
<1> ;;
<1> ;;     At first,
<1> ;;     - 'allocate page' procedure will be called;
<1> ;,        if it will return with a valid (>0) physical address
<1> ;;        (that means the relevant M.A.T. bit has been RESET)
<1> ;;        relevant memory page/block will be cleared (zeroed).
<1> ;;     - 'allocate page' will be called for allocating page
<1> ;;        directory, page table and running space (data/code).
<1> ;;     - every successful 'allocate page' call will decrease
<1> ;;       'free_pages' count (pointer).
<1> ;;     - 'out of (insufficient) memory error' will be returned
<1> ;;       if 'free_pages' points to a ZERO.
<1> ;;     - swapping out and swapping in (if it is not a new page)
<1> ;;       procedures will be called as responce to 'out of memory'
<1> ;;       error except errors caused by attribute conflicts.
<1> ;;      (swapper functions)
<1> ;;
<1> ;;     At second,
<1> ;;     - page directory entry will be updated then page table
<1> ;;       entry will be updated.
<1> ;;
<1> ;; MEMORY ALLOCATION TABLE FORMAT:
<1> ;;     - M.A.T. has a size according to available memory as
<1> ;;       follows:
<1> ;;             - 1 (allocation) bit per 1 page (4096 bytes)
<1> ;;             - a bit with value of 0 means allocated page
<1> ;;             - a bit with value of 1 means a free page
<1> ;,     - 'free_pages' pointer holds count of free pages
```

```
<1> ;;       depending on M.A.T.
<1> ;;           (NOTE: Free page count will not be checked
<1> ;;           again -on M.A.T.- after initialization.
<1> ;;           Kernel will trust on initial count.)
<1> ;;     - 'free_pages' count will be decreased by allocation
<1> ;;       and it will be increased by deallocation procedures.
<1> ;;
<1> ;;     - Available memory will be calculated during
<1> ;;       the kernel's initialization stage (in real mode).
<1> ;;       Memory allocation table and kernel page tables
<1> ;;       will be formatted/sized as result of available
<1> ;;       memory calculation before paging is enabled.
<1> ;;
<1> ;; For 4GB Available/Present Memory: (max. possible memory size)
<1> ;;     - Memory Allocation Table size will be 128 KB.
<1> ;;     - Memory allocation for kernel page directory size
<1> ;;       is always 4 KB. (in addition to total allocation size
<1> ;;       for page tables)
<1> ;;     - Memory allocation for kernel page tables (1024 tables)
<1> ;;       is 4 MB (1024*4*1024 bytes).
<1> ;;     - User (available) space will be started
<1> ;;       at 6th MB of the memory (after 1MB+4MB).
<1> ;;     - The first 640 KB is for kernel's itself plus
<1> ;;       memory allocation table and kernel's page directory
<1> ;;       (D0000h-EFFFFh may be used as kernel space...)
<1> ;;     - B0000h to B7FFFh address space (32 KB) will be used
<1> ;;       for buffers.
<1> ;;     - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
<1> ;;       (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
<1> ;;     - Kernel page tables start at 100000h (2nd MB)
<1> ;;
<1> ;; For 1GB Available Memory:
<1> ;;     - Memory Allocation Table size will be 32 KB.
<1> ;;     - Memory allocation for kernel page directory size
<1> ;;       is always 4 KB. (in addition to total allocation size
<1> ;;       for page tables)
<1> ;;     - Memory allocation for kernel page tables (256 tables)
<1> ;;       is 1 MB (256*4*1024 bytes).
<1> ;;     - User (available) space will be started
<1> ;;       at 3th MB of the memory (after 1MB+1MB).
<1> ;;     - The first 640 KB is for kernel's itself plus
<1> ;;       memory allocation table and kernel's page directory
<1> ;;       (D0000h-EFFFFh may be used as kernel space...)
<1> ;;     - B0000h to B7FFFh address space (32 KB) will be used
<1> ;;       for buffers.
<1> ;;     - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
<1> ;;       (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
<1> ;;     - Kernel page tables start at 100000h (2nd MB).
<1> ;;
<1> ;;
<1>
<1>
<1> ;;**************************************************************************************
<1> ;;
<1> ;; RETRO UNIX 386 v1 - Paging (Method for Copy On Write paging principle)
<1> ;; DEMAND PAGING - PARENT&CHILD PAGE TABLE DUPLICATION PRINCIPLES (23/04/2015)
<1>
<1> ;; Main factor: "sys fork" system call
<1> ;;
<1> ;;           FORK
<1> ;;                      |----> parent - duplicated PTEs, read only pages
<1> ;;  writable pages ---->|
<1> ;;                      |----> child - duplicated PTEs, read only pages
<1> ;;
<1> ;; AVL bit (0) of Page Table Entry is used as duplication sign
<1> ;;
<1> ;; AVL Bit 0 [PTE Bit 9] = 'Duplicated PTE belongs to child' sign/flag (if it is set)
<1> ;; Note: Dirty bit (PTE bit 6) may be used instead of AVL bit 0 (PTE bit 9)
<1> ;;        -while R/W bit is 0-.
<1> ;;
<1> ;; Duplicate page tables with writable pages (the 1st sys fork in the process):
<1> ;; # Parent's Page Table Entries are updated to point same pages as read only,
<1> ;;   as duplicated PTE bit  -AVL bit 0, PTE bit 9- are reset/clear.
<1> ;; # Then Parent's Page Table is copied to Child's Page Table.
<1> ;; # Child's Page Table Entries are updated as duplicated child bit
<1> ;;   -AVL bit 0, PTE bit 9- is set.
<1> ;;
<1> ;; Duplicate page tables with read only pages (several sys fork system calls):
<1> ;; # Parent's read only pages are copied to new child pages.
<1> ;;   Parent's PTE attributes are not changed.
<1> ;;   (Because, there is another parent-child fork before this fork! We must not
<1> ;;    destroy/mix previous fork result).
<1> ;; # Child's Page Table Entries (which are corresponding to Parent's
<1> ;;   read only pages) are set as writable (while duplicated PTE bit is clear).
<1> ;; # Parent's PTEs with writable page attribute are updated to point same pages
<1> ;;   as read only, (while) duplicated PTE bit is reset (clear).
<1> ;; # Parent's Page Table Entries (with writable page attribute) are duplicated
<1> ;;   as Child's Page Table Entries without copying actual page.
<1> ;; # Child 's Page Table Entries (which are corresponding to Parent's writable
<1> ;;   pages) are updated as duplicated PTE bit (AVL bit 0, PTE bit 9- is set.
```

```
<1> ;;
<1> ;; !? WHAT FOR (duplication after duplication):
<1> ;; In UNIX method for sys fork (a typical 'fork' application in /etc/init)
<1> ;; program/executable code continues from specified location as child process,
<1> ;; returns back previous code location as parent process, every child after
<1> ;; every sys fork uses last image of code and data just prior the fork.
<1> ;; Even if the parent code changes data, the child will not see the changed data
<1> ;; after the fork. In Retro UNIX 8086 v1, parent's process segment (32KB)
<1> ;; was copied to child's process segment (all of code and data) according to
<1> ;; original UNIX v1 which copies all of parent process code and data -core-
<1> ;; to child space -core- but swaps that core image -of child- on to disk.
<1> ;; If I (Erdogan Tan) would use a method of to copy parent's core
<1> ;; (complete running image of parent process) to the child process;
<1> ;; for big sizes, i would force Retro UNIX 386 v1 to spend many memory pages
<1> ;; and times only for a sys fork. (It would excessive reservation for sys fork,
<1> ;; because sys fork usually is prior to sys exec; sys exec always establishes
<1> ;; a new/fresh core -running space-, by clearing all code/data content).
<1> ;; 'Read Only' page flag ensures page fault handler is needed only for a few write
<1> ;; attempts between sys fork and sys exec, not more... (I say so by thinking
<1> ;; of "/etc/init" content, specially.) sys exec will clear page tables and
<1> ;; new/fresh pages will be used to load and run new executable/program.
<1> ;; That is what for i have preferred "copy on write", "duplication" method
<1> ;; for sharing same read only pages between parent and child processes.
<1> ;; That is a pitty i have to use new private flag (AVL bit 0, "duplicated PTE
<1> ;; belongs to child" sign) for cooperation on duplicated pages between a parent
<1> ;; and it's child processes; otherwise parent process would destroy data belongs
<1> ;; to its child or vice versa; or some pages would remain unclaimed
<1> ;; -deallocation problem-.
<1> ;; Note: to prevent conflicts, read only pages must not be swapped out...
<1> ;;
<1> ;; WHEN PARENT TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
<1> ;; # Page fault handler will do those:
<1> ;;   - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
<1> ;;   - If it is reset/clear, there is a child uses same page.
<1> ;;   - Parent's read only page -previous page- is copied to a new writable page.
<1> ;;   - Parent's PTE is updated as writable page, as unique page (AVL=0)
<1> ;;   - (Page fault handler whill check this PTE later, if child process causes to
<1> ;;     page fault due to write attempt on read only page. Of course, the previous
<1> ;;     read only page will be converted to writable and unique page which belongs
<1> ;;     to child process.)
<1> ;; WHEN CHILD TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
<1> ;; # Page fault handler will do those:
<1> ;;   - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
<1> ;;   - If it is set, there is a parent uses -or was using- same page.
<1> ;;   - Same PTE address within parent's page table is checked if it has same page
<1> ;;     address or not.
<1> ;;   - If parent's PTE has same address, child will continue with a new writable page.
<1> ;;     Parent's PTE will point to same (previous) page as writable, unique (AVL=0).
<1> ;;   - If parent's PTE has different address, child will continue with it's
<1> ;;     own/same page but read only flag (0) will be changed to writable flag (1) and
<1> ;;     'duplicated PTE (belongs to child)' flag/sign will be cleared/reset.

<1> ;;
<1> ;; NOTE: When a child process is terminated, read only flags of parent's page tables
<1> ;;       will be set as writable (and unique) in case of child process was using
<1> ;;       same pages with duplicated child PTE sign... Depending on sys fork and
<1> ;;       duplication method details, it is not possible multiple child processes
<1> ;;       were using same page with duplicated PTEs.
<1> ;;
<1> ;;************************************************************************************
<1>
<1> ;; 08/10/2014
<1> ;; 11/09/2014 - Retro UNIX 386 v1 PAGING (further) draft
<1> ;;              by Erdogan Tan (Based on KolibriOS 'memory.inc')
<1>
<1> ;; 'allocate_page' code is derived and modified from KolibriOS
<1> ;; 'alloc_page' procedure in 'memory.inc'
<1> ;; (25/08/2014, Revision: 5057) file
<1> ;; by KolibriOS Team (2004-2012)
<1>
<1> allocate_page:
<1>  ; 01/07/2015
<1>  ; 05/05/2015
<1>  ; 30/04/2015
<1>  ; 16/10/2014
<1>  ; 08/10/2014
<1>  ; 09/09/2014 (Retro UNIX 386 v1 - beginning)
<1>  ;
<1>  ; INPUT -> none
<1>  ;
<1>  ; OUTPUT ->
<1>  ;     EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
<1>  ;     (corresponding MEMORY ALLOCATION TABLE bit is RESET)
<1>  ;
<1>  ;     CF = 1 and EAX = 0
<1>  ;              if there is not a free page to be allocated
<1>  ;
```

```
                             <1>  ; Modified Registers -> none (except EAX)
                             <1>  ;
00003037 A1[B0700000]        <1>  mov   eax, [free_pages]
0000303C 21C0                <1>  and   eax, eax
0000303E 7438                <1>  jz    short out_of_memory
                             <1>  ;
00003040 53                  <1>  push  ebx
00003041 51                  <1>  push  ecx
                             <1>  ;
00003042 BB00001000          <1>  mov   ebx, MEM_ALLOC_TBL   ; Memory Allocation Table offset
00003047 89D9                <1>  mov   ecx, ebx
                             <1>                          ; NOTE: 32 (first_page) is initial
                             <1>                          ; value of [next_page].
                             <1>                          ; It points to the first available
                             <1>                          ; page block for users (ring 3) ...
                             <1>                          ; (MAT offset 32 = 1024/32)
                             <1>                          ; (at the of the first 4 MB)
00003049 031D[B4700000]      <1>  add   ebx, [next_page] ; Free page searching starts from here
                             <1>                      ; next_free_page >> 5
0000304F 030D[B8700000]      <1>  add   ecx, [last_page] ; Free page searching ends here
                             <1>                      ; (total_pages - 1) >> 5
                             <1> al_p_scan:
00003055 39CB                <1>  cmp   ebx, ecx
00003057 770A                <1>  ja    short al_p_notfound
                             <1>  ;
                             <1>  ; 01/07/2015
                             <1>  ; AMD64 Architecture ProgrammerÆs Manual
                             <1>  ; Volume 3:
                             <1>  ; General-Purpose and System Instructions
                             <1>  ;
                             <1>  ; BSF - Bit Scan Forward
                             <1>  ;
                             <1>  ;   Searches the value in a register or a memory location
                             <1>  ;   (second operand) for the least-significant set bit.
                             <1>  ;   If a set bit is found, the instruction clears the zero flag (ZF)
                             <1>  ;   and stores the index of the least-significant set bit in a destination
                             <1>  ;   register (first operand). If the second operand contains 0,
                             <1>  ;   the instruction sets ZF to 1 and does not change the contents of the
                             <1>  ;   destination register. The bit index is an unsigned offset from bit 0
                             <1>  ;   of the searched value
                             <1>  ;
00003059 0FBC03              <1>  bsf   eax, [ebx] ; Scans source operand for first bit set (1).
                             <1>                  ; Clear ZF if a bit is found set (1) and
                             <1>                  ; loads the destination with an index to
                             <1>                  ; first set bit. (0 -> 31)
                             <1>                  ; Sets ZF to 1 if no bits are found set.
0000305C 7525                <1>  jnz   short al_p_found ; ZF = 0 -> a free page has been found
                             <1>                ;
                             <1>                ; NOTE:  a Memory Allocation Table bit
                             <1>                ;      with value of 1 means
                             <1>                ;      the corresponding page is free
                             <1>                ;      (Retro UNIX 386 v1 feaure only!)
0000305E 83C304              <1>  add   ebx, 4
                             <1>                ; We return back for searching next page block
                             <1>                ; NOTE: [free_pages] is not ZERO; so,
                             <1>                ;     we always will find at least 1 free page here.
00003061 EBF2                <1>        jmp     short al_p_scan
                             <1>  ;
                             <1> al_p_notfound:
00003063 81E900001000        <1>  sub   ecx, MEM_ALLOC_TBL
00003069 890D[B4700000]      <1>  mov   [next_page], ecx ; next/first free page = last page
                             <1>                      ; (deallocate_page procedure will change it)
0000306F 31C0                <1>  xor   eax, eax
00003071 A3[B0700000]        <1>  mov   [free_pages], eax ; 0
00003076 59                  <1>  pop   ecx
00003077 5B                  <1>  pop   ebx
                             <1>  ;
                             <1> out_of_memory:
00003078 E857040000          <1>  call  swap_out
0000307D 7325                <1>  jnc   short al_p_ok  ; [free_pages] = 0, re-allocation by swap_out
                             <1>  ;
0000307F 29C0                <1>  sub   eax, eax ; 0
00003081 F9                  <1>  stc
00003082 C3                  <1>  retn
                             <1>
                             <1> al_p_found:
00003083 89D9                <1>  mov   ecx, ebx
00003085 81E900001000        <1>  sub   ecx, MEM_ALLOC_TBL
0000308B 890D[B4700000]      <1>  mov   [next_page], ecx ; Set first free page searching start
                             <1>                      ; address/offset (to the next)
00003091 FF0D[B0700000]      <1>        dec     dword [free_pages] ; 1 page has been allocated (X = X-1)
                             <1>  ;
00003097 0FB303              <1>  btr   [ebx], eax   ; The destination bit indexed by the source value
                             <1>                      ; is copied into the Carry Flag and then cleared
                             <1>                      ; in the destination.
                             <1>                      ;
                             <1>                      ; Reset the bit which is corresponding to the
                             <1>                      ; (just) allocated page.
                             <1>  ; 01/07/2015 (4*8 = 32, 1 allocation byte = 8 pages)
0000309A C1E103              <1>  shl   ecx, 3            ; (page block offset * 32) + page index
0000309D 01C8                <1>  add   eax, ecx     ; = page number
0000309F C1E00C              <1>  shl   eax, 12           ; physical address of the page (flat/real value)
```

```
                             <1>  ; EAX = physical address of memory page
                             <1>  ;
                             <1>  ; NOTE: The relevant page directory and page table entry will be updated
                             <1>  ;       according to this EAX value...
000030A2 59                  <1>  pop   ecx
000030A3 5B                  <1>  pop   ebx
                             <1> al_p_ok:
000030A4 C3                  <1>  retn
                             <1>
                             <1>
                             <1> make_page_dir:
                             <1>  ; 18/04/2015
                             <1>  ; 12/04/2015
                             <1>  ; 23/10/2014
                             <1>  ; 16/10/2014
                             <1>  ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     none
                             <1>  ; OUTPUT ->
                             <1>  ;     (EAX = 0)
                             <1>  ;     cf = 1 -> insufficient (out of) memory error
                             <1>  ;     cf = 0 ->
                             <1>  ;     u.pgdir = page directory (physical) address of the current
                             <1>  ;             process/user.
                             <1>  ;
                             <1>  ; Modified Registers -> EAX
                             <1>  ;
000030A5 E88DFFFFFF          <1>  call  allocate_page
000030AA 7216                <1>  jc    short mkpd_error
                             <1>  ;
000030AC A3[D3740000]        <1>  mov   [u.pgdir], eax    ; Page dir address for current user/process
                             <1>                      ; (Physical address)
                             <1> clear_page:
                             <1>  ; 18/04/2015
                             <1>  ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     EAX = physical address of the page
                             <1>  ; OUTPUT ->
                             <1>  ;     all bytes of the page will be cleared
                             <1>  ;
                             <1>  ; Modified Registers -> none
                             <1>  ;
000030B1 57                  <1>  push  edi
000030B2 51                  <1>  push  ecx
000030B3 50                  <1>  push  eax
000030B4 B900040000          <1>  mov   ecx, PAGE_SIZE / 4
000030B9 89C7                <1>  mov   edi, eax
000030BB 31C0                <1>  xor   eax, eax
000030BD F3AB                <1>  rep   stosd
000030BF 58                  <1>  pop   eax
000030C0 59                  <1>  pop   ecx
000030C1 5F                  <1>  pop   edi
                             <1> mkpd_error:
                             <1> mkpt_error:
000030C2 C3                  <1>  retn
                             <1>
                             <1> make_page_table:
                             <1>  ; 23/06/2015
                             <1>  ; 18/04/2015
                             <1>  ; 12/04/2015
                             <1>  ; 16/10/2014
                             <1>  ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     EBX = virtual (linear) address
                             <1>  ;     ECX = page table attributes (lower 12 bits)
                             <1>  ;           (higher 20 bits must be ZERO)
                             <1>  ;           (bit 0 must be 1)
                             <1>  ;     u.pgdir = page directory (physical) address
                             <1>  ; OUTPUT ->
                             <1>  ;     EDX = Page directory entry address
                             <1>  ;     EAX = Page table address
                             <1>  ;     cf = 1 -> insufficient (out of) memory error
                             <1>  ;     cf = 0 -> page table address in the PDE (EDX)
                             <1>  ;
                             <1>  ; Modified Registers -> EAX, EDX
                             <1>  ;
000030C3 E86FFFFFFF          <1>  call  allocate_page
000030C8 72F8                <1>  jc    short mkpt_error
000030CA E811000000          <1>  call  set_pde
000030CF EBE0                <1>  jmp   short clear_page
                             <1>
                             <1> make_page:
                             <1>  ; 24/07/2015
                             <1>  ; 23/06/2015 ; (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     EBX = virtual (linear) address
                             <1>  ;     ECX = page attributes (lower 12 bits)
                             <1>  ;           (higher 20 bits must be ZERO)
```

```
                              <1>  ;           (bit 0 must be 1)
                              <1>  ;     u.pgdir = page directory (physical) address
                              <1>  ; OUTPUT ->
                              <1>  ;     EBX = Virtual address
                              <1>  ;     (EDX = PTE value)
                              <1>  ;     EAX = Physical address
                              <1>  ;     cf = 1 -> insufficient (out of) memory error
                              <1>  ;
                              <1>  ; Modified Registers -> EAX, EDX
                              <1>  ;
000030D1 E861FFFFFF           <1>  call  allocate_page
000030D6 7207                 <1>  jc    short mkp_err
000030D8 E821000000           <1>  call  set_pte
000030DD 73D2                 <1>  jnc   short clear_page ; 18/04/2015
                              <1> mkp_err:
000030DF C3                   <1>  retn
                              <1>
                              <1>
                              <1> set_pde:     ; Set page directory entry (PDE)
                              <1>  ; 20/07/2015
                              <1>  ; 18/04/2015
                              <1>  ; 12/04/2015
                              <1>  ; 23/10/2014
                              <1>  ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
                              <1>  ;
                              <1>  ; INPUT ->
                              <1>  ;     EAX = physical address
                              <1>  ;           (use present value if EAX = 0)
                              <1>  ;     EBX = virtual (linear) address
                              <1>  ;     ECX = page table attributes (lower 12 bits)
                              <1>  ;           (higher 20 bits must be ZERO)
                              <1>  ;           (bit 0 must be 1)
                              <1>  ;     u.pgdir = page directory (physical) address
                              <1>  ; OUTPUT ->
                              <1>  ;     EDX = PDE address
                              <1>  ;     EAX = page table address (physical)
                              <1>  ;     ;(CF=1 -> Invalid page address)
                              <1>  ;
                              <1>  ; Modified Registers -> EDX
                              <1>  ;
000030E0 89DA                 <1>  mov   edx, ebx
000030E2 C1EA16               <1>  shr   edx, PAGE_D_SHIFT ; 22
000030E5 C1E202               <1>  shl   edx, 2 ; offset to page directory (1024*4)
000030E8 0315[D3740000]       <1>  add   edx, [u.pgdir]
                              <1>  ;
000030EE 21C0                 <1>  and   eax, eax
000030F0 7506                 <1>  jnz   short spde_1
                              <1>  ;
000030F2 8B02                 <1>  mov   eax, [edx]  ; old PDE value
                              <1>  ;test al, 1
                              <1>  ;jz   short spde_2
000030F4 662500F0             <1>  and   ax, PDE_A_CLEAR ; 0F000h  ; clear lower 12 bits
                              <1> spde_1:
                              <1>  ;and  cx, 0FFFh
000030F8 8902                 <1>  mov   [edx], eax
000030FA 66090A               <1>  or    [edx], cx
000030FD C3                   <1>  retn
                              <1> ;spde_2: ; error
                              <1> ;stc
                              <1> ;retn
                              <1>
                              <1> set_pte:     ; Set page table entry (PTE)
                              <1>  ; 24/07/2015
                              <1>  ; 20/07/2015
                              <1>  ; 23/06/2015
                              <1>  ; 18/04/2015
                              <1>  ; 12/04/2015
                              <1>  ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
                              <1>  ;
                              <1>  ; INPUT ->
                              <1>  ;     EAX = physical page address
                              <1>  ;           (use present value if EAX = 0)
                              <1>  ;     EBX = virtual (linear) address
                              <1>  ;     ECX = page attributes (lower 12 bits)
                              <1>  ;           (higher 20 bits must be ZERO)
                              <1>  ;           (bit 0 must be 1)
                              <1>  ;     u.pgdir = page directory (physical) address
                              <1>  ; OUTPUT ->
                              <1>  ;     EAX = physical page address
                              <1>  ;     (EDX = PTE value)
                              <1>  ;     EBX = virtual address
                              <1>  ;
                              <1>  ;     CF = 1 -> error
                              <1>  ;
                              <1>  ; Modified Registers -> EAX, EDX
                              <1>  ;
000030FE 50                   <1>  push  eax
000030FF A1[D3740000]         <1>  mov   eax, [u.pgdir] ; 20/07/2015
00003104 E837000000           <1>  call  get_pde
                              <1>        ; EDX = PDE address
                              <1>        ; EAX = PDE value
00003109 5A                   <1>  pop   edx ; physical page address
```

```
0000310A 722A              <1>  jc    short spte_err ; PDE not present
                           <1>  ;
0000310C 53                <1>  push  ebx ; 24/07/2015
0000310D 662500F0          <1>  and   ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
                           <1>                  ; EDX = PT address (physical)
00003111 C1EB0C            <1>  shr   ebx, PAGE_SHIFT ; 12
00003114 81E3FF030000      <1>  and   ebx, PTE_MASK     ; 03FFh
                           <1>               ; clear higher 10 bits (PD bits)
0000311A C1E302            <1>  shl   ebx, 2   ; offset to page table (1024*4)
0000311D 01C3              <1>  add   ebx, eax
                           <1>  ;
0000311F 8B03              <1>  mov   eax, [ebx] ; Old PTE value
00003121 A801              <1>  test  al, 1
00003123 740C              <1>  jz    short spte_0
00003125 09D2              <1>  or    edx, edx
00003127 750F              <1>  jnz   short spte_1
00003129 662500F0          <1>  and   ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 bits
0000312D 89C2              <1>  mov   edx, eax
0000312F EB09              <1>  jmp   short spte_2
                           <1> spte_0:
                           <1>  ; If this PTE contains a swap (disk) address,
                           <1>  ; it can be updated by using 'swap_in' procedure
                           <1>  ; only!
00003131 21C0              <1>  and   eax, eax
00003133 7403              <1>  jz    short spte_1
                           <1>  ; 24/07/2015
                           <1>  ; swapped page ! (on disk)
00003135 5B                <1>  pop   ebx
                           <1> spte_err:
00003136 F9                <1>  stc
00003137 C3                <1>  retn
                           <1> spte_1:
00003138 89D0              <1>  mov   eax, edx
                           <1> spte_2:
0000313A 09CA              <1>  or    edx, ecx
                           <1>  ; 23/06/2015
0000313C 8913              <1>  mov   [ebx], edx ; PTE value in EDX
                           <1>  ; 24/07/2015
0000313E 5B                <1>  pop   ebx
0000313F C3                <1>  retn
                           <1>
                           <1> get_pde:     ; Get present value of the relevant PDE
                           <1>  ; 20/07/2015
                           <1>  ; 18/04/2015
                           <1>  ; 12/04/2015
                           <1>  ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
                           <1>  ;
                           <1>  ; INPUT ->
                           <1>  ;     EBX = virtual (linear) address
                           <1>  ;     EAX = page directory (physical) address
                           <1>  ; OUTPUT ->
                           <1>  ;     EDX = Page directory entry address
                           <1>  ;     EAX = Page directory entry value
                           <1>  ;     CF = 1 -> PDE not present or invalid ?
                           <1>  ; Modified Registers -> EDX, EAX
                           <1>  ;
00003140 89DA              <1>  mov   edx, ebx
00003142 C1EA16            <1>  shr   edx, PAGE_D_SHIFT ; 22  (12+10)
00003145 C1E202            <1>  shl   edx, 2 ; offset to page directory (1024*4)
00003148 01C2              <1>  add   edx, eax ; page directory address (physical)
0000314A 8B02              <1>  mov   eax, [edx]
0000314C A801              <1>  test  al, PDE_A_PRESENT ; page table is present or not !
0000314E 751F              <1>  jnz   short gpte_retn
00003150 F9                <1>  stc
                           <1> gpde_retn:
00003151 C3                <1>  retn
                           <1>
                           <1> get_pte:
                           <1>        ; Get present value of the relevant PTE
                           <1>  ; 29/07/2015
                           <1>  ; 20/07/2015
                           <1>  ; 18/04/2015
                           <1>  ; 12/04/2015
                           <1>  ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
                           <1>  ;
                           <1>  ; INPUT ->
                           <1>  ;     EBX = virtual (linear) address
                           <1>  ;     EAX = page directory (physical) address
                           <1>  ; OUTPUT ->
                           <1>  ;     EDX = Page table entry address (if CF=0)
                           <1>  ;           Page directory entry address (if CF=1)
                           <1>  ;            (Bit 0 value is 0 if PT is not present)
                           <1>  ;     EAX = Page table entry value (page address)
                           <1>  ;     CF = 1 -> PDE not present or invalid ?
                           <1>  ; Modified Registers -> EAX, EDX
                           <1>  ;
00003152 E8E9FFFFFF        <1>  call  get_pde
00003157 72F8              <1>  jc    short gpde_retn   ; page table is not present
                           <1>  ;jnc  short gpte_1
                           <1>  ;retn
                           <1> ;gpte_1:
00003159 662500F0          <1>  and   ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
```

```
0000315D 89DA                 <1>  mov   edx, ebx
0000315F C1EA0C               <1>  shr   edx, PAGE_SHIFT ; 12
00003162 81E2FF030000         <1>  and   edx, PTE_MASK     ; 03FFh
                              <1>                ; clear higher 10 bits (PD bits)
00003168 C1E202               <1>  shl   edx, 2 ; offset from start of page table (1024*4)
0000316B 01C2                 <1>  add   edx, eax
0000316D 8B02                 <1>  mov   eax, [edx]
                              <1> gpte_retn:
0000316F C3                   <1>  retn
                              <1>
                              <1> deallocate_page_dir:
                              <1>  ; 15/09/2015
                              <1>  ; 05/08/2015
                              <1>  ; 30/04/2015
                              <1>  ; 28/04/2015
                              <1>  ; 17/10/2014
                              <1>  ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
                              <1>  ;
                              <1>  ; INPUT ->
                              <1>  ;     EAX = PHYSICAL ADDRESS OF THE PAGE DIRECTORY (CHILD)
                              <1>  ;     EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
                              <1>  ; OUTPUT ->
                              <1>  ;     All of page tables in the page directory
                              <1>  ;     and page dir's itself will be deallocated
                              <1>  ;     except 'read only' duplicated pages (will be converted
                              <1>  ;     to writable pages).
                              <1>  ;
                              <1>  ; Modified Registers -> EAX
                              <1>  ;
                              <1>  ;
00003170 56                   <1>  push  esi
00003171 51                   <1>  push  ecx
00003172 50                   <1>  push  eax
00003173 89C6                 <1>  mov   esi, eax
00003175 31C9                 <1>  xor   ecx, ecx
                              <1>  ; The 1st PDE points to Kernel Page Table 0 (the 1st 4MB),
                              <1>  ; it must not be deallocated
00003177 890E                 <1>  mov   [esi], ecx ; 0 ; clear PDE 0
                              <1> dapd_0:
00003179 AD                   <1>  lodsd
0000317A A801                 <1>  test  al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
0000317C 7409                 <1>  jz    short dapd_1
0000317E 662500F0             <1>  and   ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
00003182 E812000000           <1>  call  deallocate_page_table
                              <1> dapd_1:
00003187 41                   <1>  inc   ecx ; page directory entry index
00003188 81F900040000         <1>  cmp   ecx, PAGE_SIZE / 4 ; 1024
0000318E 72E9                 <1>  jb    short dapd_0
                              <1> dapd_2:
00003190 58                   <1>  pop   eax
00003191 E879000000           <1>  call  deallocate_page   ; deallocate the page dir's itself
00003196 59                   <1>  pop   ecx
00003197 5E                   <1>  pop   esi
00003198 C3                   <1>  retn
                              <1>
                              <1> deallocate_page_table:
                              <1>  ; 19/09/2015
                              <1>  ; 15/09/2015
                              <1>  ; 05/08/2015
                              <1>  ; 30/04/2015
                              <1>  ; 28/04/2015
                              <1>  ; 24/10/2014
                              <1>  ; 23/10/2014
                              <1>  ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
                              <1>  ;
                              <1>  ; INPUT ->
                              <1>  ;     EAX = PHYSICAL (real/flat) ADDRESS OF THE PAGE TABLE
                              <1>  ;     EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
                              <1>  ;     (ECX = page directory entry index)
                              <1>  ; OUTPUT ->
                              <1>  ;     All of pages in the page table and page table's itself
                              <1>  ;     will be deallocated except 'read only' duplicated pages
                              <1>  ;     (will be converted to writable pages).
                              <1>  ;
                              <1>  ; Modified Registers -> EAX
                              <1>  ;
00003199 56                   <1>  push  esi
0000319A 57                   <1>  push  edi
0000319B 52                   <1>  push  edx
0000319C 50                   <1>  push  eax ; *
0000319D 89C6                 <1>  mov   esi, eax
0000319F 31FF                 <1>  xor   edi, edi ; 0
                              <1> dapt_0:
000031A1 AD                   <1>  lodsd
000031A2 A801                 <1>  test  al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
000031A4 7441                 <1>  jz    short dapt_1
                              <1>  ;
000031A6 A802                 <1>  test  al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
                              <1>                      ; (must be 1)
000031A8 754C                 <1>  jnz   short dapt_3
                              <1>  ; Read only -duplicated- page (belongs to a parent or a child)
000031AA 66A90002             <1>         test    ax, PTE_DUPLICATED ; Was this page duplicated
```

```
                             <1>                        ; as child's page ?
000031AE 744B                <1>  jz    short dapt_4 ; Clear PTE but don't deallocate the page!
                             <1>  ; check the parent's PTE value is read only & same page or not..
                             <1>  ; ECX = page directory entry index (0-1023)
000031B0 53                  <1>  push  ebx
000031B1 51                  <1>  push  ecx
000031B2 66C1E102            <1>  shl   cx, 2 ; *4
000031B6 01CB                <1>  add   ebx, ecx ; PDE offset (for the parent)
000031B8 8B0B                <1>  mov   ecx, [ebx]
000031BA F6C101              <1>  test  cl, PDE_A_PRESENT ; present (valid) or not ?
000031BD 7435                <1>  jz    short dapt_2      ; parent process does not use this page
000031BF 6681E100F0          <1>  and   cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
                             <1>  ; EDI = page table entry index (0-1023)
000031C4 89FA                <1>  mov   edx, edi
000031C6 66C1E202            <1>  shl   dx, 2 ; *4
000031CA 01CA                <1>  add   edx, ecx ; PTE offset (for the parent)
000031CC 8B1A                <1>  mov   ebx, [edx]
000031CE F6C301              <1>  test  bl, PTE_A_PRESENT ; present or not ?
000031D1 7421                <1>  jz    short dapt_2      ; parent process does not use this page
000031D3 662500F0            <1>  and   ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
000031D7 6681E300F0          <1>  and   bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
000031DC 39D8                <1>  cmp   eax, ebx    ; parent's and child's pages are same ?
000031DE 7514                <1>  jne   short dapt_2      ; not same page
                             <1>                    ; deallocate the child's page
000031E0 800A02              <1>         or      byte [edx], PTE_A_WRITE ; convert to writable page (parent)
000031E3 59                  <1>  pop   ecx
000031E4 5B                  <1>  pop   ebx
000031E5 EB14                <1>  jmp   short dapt_4
                             <1> dapt_1:
000031E7 09C0                <1>  or    eax, eax    ; swapped page ?
000031E9 7417                <1>  jz    short dapt_5      ; no
                             <1>                    ; yes
000031EB D1E8                <1>  shr   eax, 1
000031ED E848040000          <1>  call  unlink_swap_block ; Deallocate swapped page block
                             <1>                      ; on the swap disk (or in file)
000031F2 EB0E                <1>  jmp   short dapt_5
                             <1> dapt_2:
000031F4 59                  <1>  pop   ecx
000031F5 5B                  <1>  pop   ebx
                             <1> dapt_3:
                             <1>  ;and  ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
000031F6 E814000000          <1>  call  deallocate_page
                             <1> dapt_4:
000031FB C746FC00000000      <1>  mov   dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
                             <1> dapt_5:
00003202 47                  <1>  inc   edi ; page table entry index
00003203 81FF00040000        <1>  cmp   edi, PAGE_SIZE / 4 ; 1024
00003209 7296                <1>  jb    short dapt_0
                             <1>  ;
0000320B 58                  <1>  pop   eax ; *
0000320C 5A                  <1>  pop   edx
0000320D 5F                  <1>  pop   edi
0000320E 5E                  <1>  pop   esi
                             <1>  ;
                             <1>  ;call deallocate_page   ; deallocate the page table's itself
                             <1>  ;retn
                             <1>
                             <1> deallocate_page:
                             <1>  ; 15/09/2015
                             <1>  ; 28/04/2015
                             <1>  ; 10/03/2015
                             <1>  ; 17/10/2014
                             <1>  ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
                             <1>  ; OUTPUT ->
                             <1>  ;     [free_pages] is increased
                             <1>  ;     (corresponding MEMORY ALLOCATION TABLE bit is SET)
                             <1>  ;     CF = 1 if the page is already deallocated
                             <1>  ;            (or not allocated) before.
                             <1>  ;
                             <1>  ; Modified Registers -> EAX
                             <1>  ;
0000320F 53                  <1>  push  ebx
00003210 52                  <1>  push  edx
                             <1>  ;
00003211 C1E80C              <1>  shr   eax, PAGE_SHIFT      ; shift physical address to
                             <1>                         ; 12 bits right
                             <1>                         ; to get page number
00003214 89C2                <1>  mov   edx, eax
                             <1>  ; 15/09/2015
00003216 C1EA03              <1>  shr   edx, 3                 ; to get offset to M.A.T.
                             <1>                         ; (1 allocation bit = 1 page)
                             <1>                         ; (1 allocation bytes = 8 pages)
00003219 80E2FC              <1>  and   dl, 0FCh         ; clear lower 2 bits
                             <1>                         ; (to get 32 bit position)
                             <1>  ;
0000321C BB00001000          <1>  mov   ebx, MEM_ALLOC_TBL   ; Memory Allocation Table address
00003221 01D3                <1>  add   ebx, edx
00003223 83E01F              <1>  and   eax, 1Fh         ; lower 5 bits only
                             <1>                         ; (allocation bit position)
```

```
00003226 3B15[B4700000]      <1>  cmp   edx, [next_page]     ; is the new free page address lower
                             <1>                         ; than the address in 'next_page' ?
                             <1>                         ; (next/first free page value)
0000322C 7306                <1>  jnb   short dap_1      ; no
0000322E 8915[B4700000]      <1>  mov   [next_page], edx     ; yes
                             <1> dap_1:
00003234 0FAB03              <1>  bts   [ebx], eax       ; unlink/release/deallocate page
                             <1>                         ; set relevant bit to 1.
                             <1>                         ; set CF to the previous bit value
                             <1>  ;cmc                   ; complement carry flag
                             <1>  ;jc   short dap_2      ; do not increase free_pages count
                             <1>                         ; if the page is already deallocated
                             <1>                         ; before.
00003237 FF05[B0700000]      <1>         inc     dword [free_pages]
                             <1> dap_2:
0000323D 5A                  <1>  pop   edx
0000323E 5B                  <1>  pop   ebx
0000323F C3                  <1>  retn
                             <1>
                             <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             <1> ;;                                                               ;;
                             <1> ;; Copyright (C) KolibriOS team 2004-2012. All rights reserved. ;;
                             <1> ;; Distributed under terms of the GNU General Public License    ;;
                             <1> ;;                                                               ;;
                             <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             <1>
                             <1> ;;$Revision: 5057 $
                             <1>
                             <1>
                             <1> ;;align 4
                             <1> ;;proc alloc_page
                             <1>
                             <1> ;;        pushfd
                             <1> ;;        cli
                             <1> ;;        push    ebx
                             <1> ;;;//-
                             <1> ;;        cmp     [pg_data.pages_free], 1
                             <1> ;;        jle     .out_of_memory
                             <1> ;;;//-
                             <1> ;;
                             <1> ;;        mov     ebx, [page_start]
                             <1> ;;        mov     ecx, [page_end]
                             <1> ;;.l1:
                             <1> ;;        bsf     eax, [ebx];
                             <1> ;;        jnz     .found
                             <1> ;;        add     ebx, 4
                             <1> ;;        cmp     ebx, ecx
                             <1> ;;        jb      .l1
                             <1> ;;        pop     ebx
                             <1> ;;        popfd
                             <1> ;;        xor     eax, eax
                             <1> ;;        ret
                             <1> ;;.found:
                             <1> ;;;//-
                             <1> ;;        dec     [pg_data.pages_free]
                             <1> ;;        jz      .out_of_memory
                             <1> ;;;//-
                             <1> ;;        btr     [ebx], eax
                             <1> ;;        mov     [page_start], ebx
                             <1> ;;        sub     ebx, sys_pgmap
                             <1> ;;        lea     eax, [eax+ebx*8]
                             <1> ;;        shl     eax, 12
                             <1> ;;;//-       dec [pg_data.pages_free]
                             <1> ;;        pop     ebx
                             <1> ;;        popfd
                             <1> ;;        ret
                             <1> ;;;//-
                             <1> ;;.out_of_memory:
                             <1> ;;        mov     [pg_data.pages_free], 1
                             <1> ;;        xor     eax, eax
                             <1> ;;        pop     ebx
                             <1> ;;        popfd
                             <1> ;;        ret
                             <1> ;;;//-
                             <1> ;;endp
                             <1>
                             <1> duplicate_page_dir:
                             <1>  ; 21/09/2015
                             <1>  ; 31/08/2015
                             <1>  ; 20/07/2015
                             <1>  ; 28/04/2015
                             <1>  ; 27/04/2015
                             <1>  ; 18/04/2015
                             <1>  ; 12/04/2015
                             <1>  ; 18/10/2014
                             <1>  ; 16/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
                             <1>  ;               page directory.
                             <1>  ; OUTPUT ->
                             <1>  ;     EAX =  PHYSICAL (real/flat) ADDRESS of the child's
                             <1>  ;           page directory.
                             <1>  ;     (New page directory with new page table entries.)
                             <1>  ;     (New page tables with read only copies of the parent's
                             <1>  ;     pages.)
                             <1>  ;     EAX = 0 -> Error (CF = 1)
                             <1>  ;
                             <1>  ; Modified Registers -> none (except EAX)
                             <1>  ;
00003240 E8F2FDFFFF          <1>  call  allocate_page
00003245 723E                <1>  jc    short dpd_err
                             <1>  ;
00003247 55                  <1>  push  ebp ; 20/07/2015
00003248 56                  <1>  push  esi
00003249 57                  <1>  push  edi
0000324A 53                  <1>  push  ebx
0000324B 51                  <1>  push  ecx
0000324C 8B35[D3740000]      <1>  mov   esi, [u.pgdir]
00003252 89C7                <1>  mov   edi, eax
00003254 50                  <1>  push  eax ; save child's page directory address
                             <1>  ; 31/08/2015
                             <1>  ; copy PDE 0 from the parent's page dir to the child's page dir
                             <1>  ; (use same system space for all user page tables)
00003255 A5                  <1>  movsd
00003256 BD00004000          <1>  mov   ebp, 1024*4096 ; pass the 1st 4MB (system space)
0000325B B9FF030000          <1>  mov   ecx, (PAGE_SIZE / 4) - 1 ; 1023
                             <1> dpd_0:
00003260 AD                  <1>  lodsd
                             <1>  ;or   eax, eax
                             <1>        ;jnz      short dpd_1
00003261 A801                <1>  test  al, PDE_A_PRESENT ;  bit 0 =  1
00003263 7508                <1>  jnz   short dpd_1
                             <1>  ; 20/07/2015 (virtual address at the end of the page table)
00003265 81C500004000        <1>  add   ebp, 1024*4096 ; page size * PTE count
0000326B EB0F                <1>  jmp   short dpd_2
                             <1> dpd_1:
0000326D 662500F0            <1>  and   ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
00003271 89C3                <1>  mov   ebx, eax
                             <1>  ; EBX = Parent's page table address
00003273 E81F000000          <1>  call  duplicate_page_table
00003278 720C                <1>  jc    short dpd_p_err
                             <1>  ; EAX = Child's page table address
0000327A 0C07                <1>  or    al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
                             <1>               ; set bit 0, bit 1 and bit 2 to 1
                             <1>               ; (present, writable, user)
                             <1> dpd_2:
0000327C AB                  <1>  stosd
0000327D E2E1                <1>  loop  dpd_0
                             <1>  ;
0000327F 58                  <1>  pop   eax  ; restore child's page directory address
                             <1> dpd_3:
00003280 59                  <1>  pop   ecx
00003281 5B                  <1>  pop   ebx
00003282 5F                  <1>  pop   edi
00003283 5E                  <1>  pop   esi
00003284 5D                  <1>  pop   ebp ; 20/07/2015
                             <1> dpd_err:
00003285 C3                  <1>  retn
                             <1> dpd_p_err:
                             <1>  ; release the allocated pages missing (recover free space)
00003286 58                  <1>  pop   eax  ; the new page directory address (physical)
00003287 8B1D[D3740000]      <1>  mov   ebx, [u.pgdir] ; parent's page directory address
0000328D E8DEFEFFFF          <1>  call  deallocate_page_dir
00003292 29C0                <1>  sub   eax, eax ; 0
00003294 F9                  <1>  stc
00003295 EBE9                <1>  jmp   short dpd_3
                             <1>
                             <1> duplicate_page_table:
                             <1>  ; 21/09/2015
                             <1>  ; 20/07/2015
                             <1>  ; 05/05/2015
                             <1>  ; 28/04/2015
                             <1>  ; 27/04/2015
                             <1>  ; 18/04/2015
                             <1>  ; 18/10/2014
                             <1>  ; 16/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
                             <1>  ;     EBP = page table entry index (from 'duplicate_page_dir')
                             <1>  ; OUTPUT ->
                             <1>  ;     EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
                             <1>  ;           (with 'read only' attribute of page table entries)
                             <1>  ;     EBP = (recent) page table index (for 'add_to_swap_queue')
                             <1>  ;     CF = 1 -> error
                             <1>  ;
                             <1>  ; Modified Registers -> EBP (except EAX)
                             <1>  ;
00003297 E89BFDFFFF          <1>  call  allocate_page
0000329C 726A                <1>  jc    short dpt_err
                             <1>  ;
0000329E 50                  <1>  push  eax ; *
0000329F 56                  <1>  push  esi
```

```
000032A0 57                  <1>  push  edi
000032A1 52                  <1>  push  edx
000032A2 51                  <1>  push  ecx
                             <1>  ;
000032A3 89DE                <1>  mov   esi, ebx
000032A5 89C7                <1>  mov   edi, eax
000032A7 89C2                <1>  mov   edx, eax
000032A9 81C200100000        <1>  add   edx, PAGE_SIZE
                             <1> dpt_0:
000032AF AD                  <1>  lodsd
000032B0 21C0                <1>  and   eax, eax
000032B2 7444                <1>  jz    short dpt_3
000032B4 A801                <1>  test  al, PTE_A_PRESENT ;  bit 0 =  1
000032B6 7507                <1>  jnz   short dpt_1
                             <1>  ; 20/07/2015
                             <1>  ; ebp = virtual (linear) address of the memory page
000032B8 E887040000          <1>  call  reload_page ; 28/04/2015
000032BD 7244                <1>  jc    short dpt_p_err
                             <1> dpt_1:
                             <1>  ; 21/09/2015
000032BF 89C1                <1>  mov   ecx, eax
000032C1 662500F0            <1>  and   ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
000032C5 F6C102              <1>  test  cl, PTE_A_WRITE ; writable page ?
000032C8 7525                <1>  jnz   short dpt_2
                             <1>  ; Read only (parent) page
                             <1>  ;     - there is a third process which uses this page -
                             <1>  ; Allocate a new page for the child process
000032CA E868FDFFFF          <1>  call  allocate_page
000032CF 7232                <1>  jc    short dpt_p_err
000032D1 57                  <1>  push  edi
000032D2 56                  <1>  push  esi
000032D3 89CE                <1>  mov   esi, ecx
000032D5 89C7                <1>  mov   edi, eax
000032D7 B900040000          <1>  mov   ecx, PAGE_SIZE/4
000032DC F3A5                <1>  rep   movsd ; copy page (4096 bytes)
000032DE 5E                  <1>  pop   esi
000032DF 5F                  <1>  pop   edi
                             <1>  ;
000032E0 53                  <1>  push  ebx
000032E1 50                  <1>  push  eax
                             <1>  ; 20/07/2015
000032E2 89EB                <1>  mov   ebx, ebp
                             <1>  ; ebx = virtual address of the memory page
000032E4 E80B030000          <1>  call  add_to_swap_queue
000032E9 58                  <1>  pop   eax
000032EA 5B                  <1>  pop   ebx
                             <1>  ; 21/09/2015
000032EB 0C07                <1>  or    al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
                             <1>        ; user + writable + present page
000032ED EB09                <1>  jmp   short dpt_3
                             <1> dpt_2:
                             <1>  ;or   ax, PTE_A_USER+PTE_A_PRESENT
000032EF 0C05                <1>  or    al, PTE_A_USER+PTE_A_PRESENT
                             <1>            ; (read only page!)
000032F1 8946FC              <1>  mov   [esi-4], eax ; update parent's PTE
000032F4 660D0002            <1>  or      ax, PTE_DUPLICATED  ; (read only page & duplicated PTE!)
                             <1> dpt_3:
000032F8 AB                  <1>  stosd  ; EDI points to child's PTE
                             <1>  ;
000032F9 81C500100000        <1>  add   ebp, 4096 ; 20/07/2015 (next page)
                             <1>  ;
000032FF 39D7                <1>  cmp   edi, edx
00003301 72AC                <1>  jb    short dpt_0
                             <1> dpt_p_err:
00003303 59                  <1>  pop   ecx
00003304 5A                  <1>  pop   edx
00003305 5F                  <1>  pop   edi
00003306 5E                  <1>  pop   esi
00003307 58                  <1>  pop   eax ; *
                             <1> dpt_err:
00003308 C3                  <1>  retn
                             <1>
                             <1> page_fault_handler:      ; CPU EXCEPTION 0Eh (14) : Page Fault !
                             <1>  ; 21/09/2015
                             <1>  ; 19/09/2015
                             <1>  ; 17/09/2015
                             <1>  ; 28/08/2015
                             <1>  ; 20/07/2015
                             <1>  ; 28/06/2015
                             <1>  ; 03/05/2015
                             <1>  ; 30/04/2015
                             <1>  ; 18/04/2015
                             <1>  ; 12/04/2015
                             <1>  ; 30/10/2014
                             <1>  ; 11/09/2014
                             <1>  ; 10/09/2014 (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; Note: This is not an interrupt/exception handler.
                             <1>  ;     This is a 'page fault remedy' subroutine
                             <1>  ;     which will be called by standard/uniform
                             <1>  ;     exception handler.
                             <1>  ;
```

```
<1>  ; INPUT ->
<1>  ;     [error_code] = 32 bit ERROR CODE (lower 5 bits are valid)
<1>  ;
<1>  ;     cr2 = the virtual (linear) address
<1>  ;           which has caused to page fault (19/09/2015)
<1>  ;
<1>  ; OUTPUT ->
<1>  ;     (corresponding PAGE TABLE ENTRY is mapped/set)
<1>  ;     EAX = 0 -> no error
<1>  ;     EAX > 0 -> error code in EAX (also CF = 1)
<1>  ;
<1>  ; Modified Registers -> none (except EAX)
<1>  ;
<1>         ;
<1>         ; ERROR CODE:
<1>  ;      31  .....  4   3 2   1 0
<1>  ;     +---+-- --+---+---+---+---+---+---+
<1>  ;     |   Reserved  | I | R | U | W | P |
<1>  ;     +---+-- --+---+---+---+---+---+---+
<1>  ;
<1>  ; P : PRESENT -   When set, the page fault was caused by
<1>        ;           a page-protection violation. When not set,
<1>  ;           it was caused by a non-present page.
<1>  ; W : WRITE   -   When set, the page fault was caused by
<1>  ;           a page write. When not set, it was caused
<1>  ;           by a page read.
<1>  ; U : USER    -   When set, the page fault was caused
<1>  ;           while CPL = 3.
<1>  ;           This does not necessarily mean that
<1>  ;           the page fault was a privilege violation.
<1>  ; R : RESERVD -   When set, the page fault was caused by
<1>  ;     WRITE reading a 1 in a reserved field.
<1>  ; I : INSTRUC -   When set, the page fault was caused by
<1>  ;     FETCH an instruction fetch
<1>  ;
<1>  ;; x86 (32 bit) VIRTUAL ADDRESS TRANSLATION
<1>  ;  31               22                  12 11                    0
<1>  ; +-------------------+-------------------+-----------------------+
<1>        ; | PAGE DIR. ENTRY # | PAGE TAB. ENTRY # |        OFFSET         |
<1>        ; +-------------------+-------------------+-----------------------+
<1>  ;
<1>
<1>  ;; CR3 REGISTER (Control Register 3)
<1>  ;  31                                   12             5 4 3 2   0
<1>  ; +---------------------------------------+-------------+---+-----+
<1>        ; |                                       |          |P|P|     |
<1>        ; |   PAGE DIRECTORY TABLE BASE ADDRESS   |  reserved|C|W|rsvrd|
<1>        ; |                                       |          |D|T|     |
<1>        ; +---------------------------------------+-------------+---+-----+
<1>  ;
<1>  ;     PWT    - WRITE THROUGH
<1>  ;     PCD    - CACHE DISABLE
<1>  ;
<1>  ;
<1>  ;; x86 PAGE DIRECTORY ENTRY (4 KByte Page)
<1>  ;  31                                   12 11  9 8 7 6 5 4 3 2 1 0
<1>  ; +---------------------------------------+-----+---+-+-+---+-+-+-+
<1>        ; |                                       |     | | | | |P|P|U|R| |
<1>        ; |     PAGE TABLE BASE ADDRESS 31..12    | AVL |G|0|D|A|C|W|/|/|P|
<1>        ; |                                       |     | | | | |D|T|S|W| |
<1>        ; +---------------------------------------+-----+---+-+-+---+-+-+-+
<1>  ;
<1>         ;       P      - PRESENT
<1>         ;       R/W    - READ/WRITE
<1>         ;       U/S    - USER/SUPERVISOR
<1>  ;     PWT    - WRITE THROUGH
<1>  ;     PCD    - CACHE DISABLE
<1>  ;     A      - ACCESSED
<1>         ;       D      - DIRTY (IGNORED)
<1>  ;     PAT    - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
<1>  ;     G      - GLOBAL   (IGNORED)
<1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
<1>  ;
<1>  ;
<1>  ;; x86 PAGE TABLE ENTRY (4 KByte Page)
<1>  ;  31                                   12 11  9 8 7 6 5 4 3 2 1 0
<1>  ; +---------------------------------------+-----+---+-+-+---+-+-+-+
<1>        ; |                                       |     | |P| | |P|P|U|R| |
<1>        ; |     PAGE FRAME BASE ADDRESS 31..12    | AVL |G|A|D|A|C|W|/|/|P|
<1>        ; |                                       |     | |T| | |D|T|S|W| |
<1>        ; +---------------------------------------+-----+---+-+-+---+-+-+-+
<1>  ;
<1>         ;       P      - PRESENT
<1>         ;       R/W    - READ/WRITE
<1>         ;       U/S    - USER/SUPERVISOR
<1>  ;     PWT    - WRITE THROUGH
<1>  ;     PCD    - CACHE DISABLE
<1>  ;     A      - ACCESSED
<1>         ;       D      - DIRTY
<1>  ;     PAT    - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
<1>  ;     G      - GLOBAL
<1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
```

```
                             <1>  ;
                             <1>  ;
                             <1>  ;; 80386 PAGE TABLE ENTRY (4 KByte Page)
                             <1>  ;  31                                  12 11  9 8 7 6 5 4 3 2 1 0
                             <1>  ; +-------------------------------------+-----+-+-+-+-+---+-+-+-+
                             <1>        ; |                                     |     | | | | | | |U|R| |
                             <1>        ; |     PAGE FRAME BASE ADDRESS 31..12    | AVL |0|0|D|A|0|0|/|/|P|
                             <1>        ; |                                     |     | | | | | | |S|W| |
                             <1>        ; +-------------------------------------+-----+-+-+-+-+---+-+-+-+
                             <1>  ;
                             <1>         ;       P      - PRESENT
                             <1>         ;       R/W    - READ/WRITE
                             <1>         ;       U/S    - USER/SUPERVISOR
                             <1>         ;       D      - DIRTY
                             <1>         ;       AVL    - AVAILABLE FOR SYSTEMS PROGRAMMER USE
                             <1>  ;
                             <1>         ;       NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.
                             <1>  ;
                             <1>  ;
                             <1>  ;; Invalid Page Table Entry
                             <1>  ; 31                                                           1 0
                             <1>        ; +-----------------------------------------------------------+-+
                             <1>        ; |                                                           | |
                             <1>        ; |                         AVAILABLE                         |0|
                             <1>        ; |                                                           | |
                             <1>        ; +-----------------------------------------------------------+-+
                             <1>  ;
                             <1>
00003309 53                  <1>  push  ebx
0000330A 52                  <1>  push  edx
0000330B 51                  <1>  push  ecx
                             <1>  ;
                             <1>  ; 21/09/2015 (debugging)
0000330C FF05[E3740000]      <1>  inc   dword [u.pfcount] ; page fault count for running process
00003312 FF05[50810000]      <1>  inc   dword [PF_Count] ; total page fault count
                             <1>  ; 28/06/2015
                             <1>  ;mov  edx, [error_code] ; Lower 5 bits are valid
00003318 8A15[48810000]      <1>  mov   dl, [error_code]
                             <1>  ;
0000331E F6C201              <1>  test  dl, 1 ; page fault was caused by a non-present page
                             <1>              ; sign
00003321 7422                <1>  jz    short pfh_alloc_np
                             <1>  ;
                             <1>  ; If it is not a 'write on read only page' type page fault
                             <1>  ; major page fault error with minor reason must be returned without
                             <1>  ; fixing the problem. 'sys_exit with error' will be needed
                             <1>  ; after return here!
                             <1>  ; Page fault will be remedied, by copying page contents
                             <1>  ; to newly allocated page with write permission;
                             <1>  ; sys_fork -> sys_exec -> copy on write, demand paging method is
                             <1>  ; used for working with minimum possible memory usage.
                             <1>  ; sys_fork will duplicate page directory and tables of parent
                             <1>  ; process with 'read only' flag. If the child process attempts to
                             <1>  ; write on these read only pages, page fault will be directed here
                             <1>  ; for allocating a new page with same data/content.
                             <1>  ;
                             <1>  ; IMPORTANT : Retro UNIX 386 v1 (and SINGLIX and TR-DOS)
                             <1>  ; will not force to separate CODE and DATA space
                             <1>  ; in a process/program...
                             <1>  ; CODE segment/section may contain DATA!
                             <1>  ; It is flat, smoth and simplest programming method already as in
                             <1>  ; Retro UNIX 8086 v1 and MS-DOS programs.
                             <1>  ;
00003323 F6C202              <1>  test  dl, 2 ; page fault was caused by a page write
                             <1>              ; sign
00003326 0F84AB000000        <1>        jz      pfh_p_err
                             <1>  ; 31/08/2015
0000332C F6C204              <1>  test  dl, 4 ; page fault was caused while CPL = 3 (user mode)
                             <1>              ; sign.  (U+W+P = 4+2+1 = 7)
0000332F 0F84A2000000        <1>        jz   pfh_pv_err
                             <1>  ;
                             <1>  ; make a new page and copy the parent's page content
                             <1>  ; as the child's new page content
                             <1>  ;
00003335 0F20D3              <1>  mov   ebx, cr2 ; CR2 contains the linear address
                             <1>               ; which has caused to page fault
00003338 E8A2000000          <1>  call  copy_page
0000333D 0F828D000000        <1>        jc      pfh_im_err ; insufficient memory
                             <1>  ;
00003343 EB7D                <1>        jmp     pfh_cpp_ok
                             <1>  ;
                             <1> pfh_alloc_np:
00003345 E8EDFCFFFF          <1>  call  allocate_page     ; (allocate a new page)
0000334A 0F8280000000        <1>        jc      pfh_im_err     ; 'insufficient memory' error
                             <1> pfh_chk_cpl:
                             <1>  ; EAX = Physical (base) address of the allocated (new) page
                             <1>        ; (Lower 12 bits are ZERO, because
                             <1>        ;     the address is on a page boundary)
00003350 80E204              <1>  and   dl, 4 ; CPL = 3 ?
00003353 7505                <1>  jnz   short pfh_um
                             <1>              ; Page fault handler for kernel/system mode (CPL=0)
00003355 0F20DB              <1>  mov   ebx, cr3 ; CR3 (Control Register 3) contains physical address
```

```
                             <1>               ; of the current/active page directory
                             <1>               ; (Always kernel/system mode page directory, here!)
                             <1>               ; Note: Lower 12 bits are 0. (page boundary)
00003358 EB06                <1>  jmp   short pfh_get_pde
                             <1>  ;
                             <1> pfh_um:            ; Page fault handler for user/appl. mode (CPL=3)
0000335A 8B1D[D3740000]      <1>  mov   ebx, [u.pgdir] ; Page directory of current/active process
                             <1>              ; Physical address of the USER's page directory
                             <1>              ; Note: Lower 12 bits are 0. (page boundary)
                             <1> pfh_get_pde:
00003360 80CA03              <1>  or    dl, 3 ; USER + WRITE + PRESENT or SYSTEM + WRITE + PRESENT
00003363 0F20D1              <1>  mov   ecx, cr2 ; CR2 contains the virtual address
                             <1>               ; which has been caused to page fault
                             <1>               ;
00003366 C1E914              <1>  shr   ecx, 20      ; shift 20 bits right
00003369 80E1FC              <1>  and   cl, 0FCh ; mask lower 2 bits to get PDE offset
                             <1>  ;
0000336C 01CB                <1>  add   ebx, ecx ; now, EBX points to the relevant page dir entry
0000336E 8B0B                <1>  mov   ecx, [ebx] ; physical (base) address of the page table
00003370 F6C101              <1>  test  cl, 1  ; check bit 0 is set (1) or not (0).
00003373 740B                <1>  jz    short pfh_set_pde ; Page directory entry is not valid,
                             <1>                      ; set/validate page directory entry
00003375 6681E100F0          <1>  and   cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
0000337A 89CB                <1>  mov   ebx, ecx ; Physical address of the page table
0000337C 89C1                <1>  mov   ecx, eax ; new page address (physical)
0000337E EB16                <1>  jmp   short pfh_get_pte
                             <1> pfh_set_pde:
                             <1>  ;; NOTE: Page directories and page tables never be swapped out!
                             <1>  ;;     (So, we know this PDE is empty or invalid)
                             <1>  ;
00003380 08D0                <1>  or    al, dl       ; lower 3 bits are used as U/S, R/W, P flags
00003382 8903                <1>  mov   [ebx], eax ; Let's put the new page directory entry here !
00003384 30C0                <1>  xor   al, al       ; clear lower (3..8) bits
00003386 89C3                <1>  mov   ebx, eax
00003388 E8AAFCFFFF          <1>  call  allocate_page      ; (allocate a new page)
0000338D 7241                <1>  jc    short pfh_im_err   ; 'insufficient memory' error
                             <1> pfh_spde_1:
                             <1>  ; EAX = Physical (base) address of the allocated (new) page
0000338F 89C1                <1>  mov   ecx, eax
00003391 E81BFDFFFF          <1>  call  clear_page ; Clear page content
                             <1> pfh_get_pte:
00003396 0F20D0              <1>  mov   eax, cr2 ; virtual address
                             <1>               ; which has been caused to page fault
00003399 89C7                <1>  mov   edi, eax ; 20/07/2015
0000339B C1E80C              <1>  shr   eax, 12      ; shift 12 bit right to get
                             <1>               ; higher 20 bits of the page fault address
0000339E 25FF030000          <1>  and   eax, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
000033A3 C1E002              <1>  shl   eax, 2     ; shift 2 bits left to get PTE offset
000033A6 01C3                <1>  add   ebx, eax ; now, EBX points to the relevant page table entry
000033A8 8B03                <1>  mov   eax, [ebx] ; get previous value of pte
                             <1>        ; bit 0 of EAX is always 0 (otherwise we would not be here)
000033AA 21C0                <1>  and   eax, eax
000033AC 7410                <1>  jz    short pfh_gpte_1
                             <1>  ; 20/07/2015
000033AE 87D9                <1>  xchg  ebx, ecx ; new page address (physical)
000033B0 55                  <1>  push  ebp ; 20/07/2015
000033B1 0F20D5              <1>  mov   ebp, cr2
                             <1>        ; ECX = physical address of the page table entry
                             <1>        ; EBX = Memory page address (physical!)
                             <1>        ; EAX = Swap disk (offset) address
                             <1>        ; EBP = virtual address (page fault address)
000033B4 E8B7000000          <1>  call  swap_in
000033B9 5D                  <1>  pop   ebp
000033BA 7210                <1>  jc      short pfh_err_retn
000033BC 87CB                <1>  xchg  ecx, ebx
                             <1>        ; EBX = physical address of the page table entry
                             <1>        ; ECX = new page
                             <1> pfh_gpte_1:
000033BE 08D1                <1>  or    cl, dl      ; lower 3 bits are used as U/S, R/W, P flags
000033C0 890B                <1>  mov   [ebx], ecx ; Let's put the new page table entry here !
                             <1> pfh_cpp_ok:
                             <1>  ; 20/07/2015
000033C2 0F20D3              <1>  mov   ebx, cr2
000033C5 E82A020000          <1>  call  add_to_swap_queue
                             <1>  ;
                             <1>  ; The new PTE (which contains the new page) will be added to
                             <1>  ; the swap queue, here.
                             <1>  ; (Later, if memory will become insufficient,
                             <1>  ; one page will be swapped out which is at the head of
                             <1>  ; the swap queue by using FIFO and access check methods.)
                             <1>  ;
000033CA 31C0                <1>  xor   eax, eax  ; 0
                             <1>  ;
                             <1> pfh_err_retn:
000033CC 59                  <1>  pop   ecx
000033CD 5A                  <1>  pop   edx
000033CE 5B                  <1>  pop   ebx
000033CF C3                  <1>  retn
                             <1>
                             <1> pfh_im_err:
000033D0 B8E1000000          <1>  mov   eax, ERR_MAJOR_PF + ERR_MINOR_IM ; Error code in AX
                             <1>              ; Major (Primary) Error: Page Fault
```

```
                        <1>              ; Minor (Secondary) Error: Insufficient Memory !
000033D5 EBF5           <1>  jmp   short pfh_err_retn
                        <1>
                        <1>
                        <1> pfh_p_err: ; 09/03/2015
                        <1> pfh_pv_err:
                        <1>  ; Page fault was caused by a protection-violation
000033D7 B8E3000000     <1>  mov   eax, ERR_MAJOR_PF + ERR_MINOR_PV ; Error code in AX
                        <1>              ; Major (Primary) Error: Page Fault
                        <1>              ; Minor (Secondary) Error: Protection violation !
000033DC F9             <1>  stc
000033DD EBED           <1>  jmp   short pfh_err_retn
                        <1>
                        <1> copy_page:
                        <1>  ; 22/09/2015
                        <1>  ; 21/09/2015
                        <1>  ; 19/09/2015
                        <1>  ; 07/09/2015
                        <1>  ; 31/08/2015
                        <1>  ; 20/07/2015
                        <1>  ; 05/05/2015
                        <1>  ; 03/05/2015
                        <1>  ; 18/04/2015
                        <1>  ; 12/04/2015
                        <1>  ; 30/10/2014
                        <1>  ; 18/10/2014 (Retro UNIX 386 v1 - beginning)
                        <1>  ;
                        <1>  ; INPUT ->
                        <1>  ;     EBX = Virtual (linear) address of source page
                        <1>  ;          (Page fault address)
                        <1>  ; OUTPUT ->
                        <1>  ;     EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
                        <1>  ;     (corresponding PAGE TABLE ENTRY is mapped/set)
                        <1>  ;     EAX = 0 (CF = 1)
                        <1>  ;           if there is not a free page to be allocated
                        <1>  ;     (page content of the source page will be copied
                        <1>  ;     onto the target/new page)
                        <1>  ;
                        <1>  ; Modified Registers -> ecx, ebx (except EAX)
                        <1>  ;
000033DF 56             <1>  push  esi
000033E0 57             <1>  push  edi
                        <1>  ;push ebx
                        <1>  ;push ecx
000033E1 31F6           <1>  xor   esi, esi
000033E3 C1EB0C         <1>  shr   ebx, 12 ; shift 12 bits right to get PDE & PTE numbers
000033E6 89D9           <1>  mov   ecx, ebx ; save page fault address (as 12 bit shifted)
000033E8 C1EB08         <1>  shr   ebx, 8      ; shift 8 bits right and then
000033EB 80E3FC         <1>  and   bl, 0FCh ; mask lower 2 bits to get PDE offset
000033EE 89DF           <1>  mov   edi, ebx ; save it for the parent of current process
000033F0 031D[D3740000] <1>  add   ebx, [u.pgdir] ; EBX points to the relevant page dir entry
000033F6 8B03           <1>  mov   eax, [ebx] ; physical (base) address of the page table
000033F8 662500F0       <1>  and   ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
000033FC 89CB           <1>  mov   ebx, ecx   ; (restore higher 20 bits of page fault address)
000033FE 81E3FF030000   <1>  and   ebx, 3FFh  ; mask PDE# bits, the result is PTE# (0 to 1023)
00003404 66C1E302       <1>  shl   bx, 2    ; shift 2 bits left to get PTE offset
00003408 01C3           <1>  add   ebx, eax   ; EBX points to the relevant page table entry
                        <1>  ; 07/09/2015
0000340A 66F7030002     <1>         test    word [ebx], PTE_DUPLICATED ; (Does current process share this
                        <1>                           ; read only page as a child process?)
0000340F 7509           <1>  jnz   short cpp_0 ; yes
00003411 8B0B           <1>  mov   ecx, [ebx] ; PTE value
00003413 6681E100F0     <1>  and   cx, PTE_A_CLEAR ; 0F000h  ; clear page attributes
00003418 EB32           <1>  jmp   short cpp_1
                        <1> cpp_0:
0000341A 89FE           <1>  mov   esi, edi
0000341C 0335[D7740000] <1>  add   esi, [u.ppgdir] ; the parent's page directory entry
00003422 8B06           <1>  mov   eax, [esi] ; physical (base) address of the page table
00003424 662500F0       <1>  and   ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
00003428 89CE           <1>  mov   esi, ecx   ; (restore higher 20 bits of page fault address)
0000342A 81E6FF030000   <1>  and   esi, 3FFh  ; mask PDE# bits, the result is PTE# (0 to 1023)
00003430 66C1E602       <1>  shl   si, 2    ; shift 2 bits left to get PTE offset
00003434 01C6           <1>  add   esi, eax   ; EDX points to the relevant page table entry
00003436 8B0E           <1>  mov   ecx, [esi] ; PTE value of the parent process
                        <1>  ; 21/09/2015
00003438 8B03           <1>  mov   eax, [ebx] ; PTE value of the child process
0000343A 662500F0       <1>  and   ax, PTE_A_CLEAR ; 0F000h ; clear page attributes
                        <1>  ;
0000343E F6C101         <1>  test  cl, PTE_A_PRESENT ; is it a present/valid page ?
00003441 7424           <1>  jz    short cpp_3 ; the parent's page is not same page
                        <1>  ;
00003443 6681E100F0     <1>  and   cx, PTE_A_CLEAR ; 0F000h ; clear page attributes
00003448 39C8           <1>  cmp   eax, ecx   ; Same page?
0000344A 751B           <1>  jne   short cpp_3 ; Parent page and child page are not same
                        <1>                    ; Convert child's page to writable page
                        <1> cpp_1:
0000344C E8E6FBFFFF     <1>  call  allocate_page
00003451 721A           <1>  jc    short cpp_4 ; 'insufficient memory' error
00003453 21F6           <1>  and   esi, esi    ; check ESI is valid or not
00003455 7405           <1>  jz    short cpp_2
                        <1>        ; Convert read only page to writable page
                        <1>        ;(for the parent of the current process)
```

```
                             <1>  ;and  word [esi], PTE_A_CLEAR ; 0F000h
                             <1>  ; 22/09/2015
00003457 890E                <1>  mov   [esi], ecx
00003459 800E07              <1>  or    byte [esi], PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER
                             <1>                     ; 1+2+4 = 7
                             <1> cpp_2:
0000345C 89C7                <1>  mov   edi, eax ; new page address of the child process
                             <1>  ; 07/09/2015
0000345E 89CE                <1>  mov   esi, ecx ; the page address of the parent process
00003460 B900040000          <1>  mov   ecx, PAGE_SIZE / 4
00003465 F3A5                <1>  rep   movsd ; 31/08/2015
                             <1> cpp_3:
00003467 0C07                <1>  or    al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 1+2+4 = 7
00003469 8903                <1>  mov   [ebx], eax ; Update PTE
0000346B 28C0                <1>  sub   al, al ; clear attributes
                             <1> cpp_4:
                             <1>  ;pop  ecx
                             <1>  ;pop  ebx
0000346D 5F                  <1>  pop   edi
0000346E 5E                  <1>  pop   esi
0000346F C3                  <1>  retn
                             <1>
                             <1> ;; 28/04/2015
                             <1> ;; 24/10/2014
                             <1> ;; 21/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1> ;; SWAP_PAGE_QUEUE (4096 bytes)
                             <1> ;;
                             <1> ;;   0000   0001   0002   0003   ....   1020   1021   1022   1023
                             <1> ;; +------+------+------+------+-    -+------+------+------+------+
                             <1> ;; |  pg1 |  pg2 |  pg3 |  pg4 | .... |pg1021|pg1022|pg1023|pg1024|
                             <1> ;; +------+------+------+------+-    -+------+------+------+------+
                             <1> ;;
                             <1> ;; [swpq_last] = 0 to 4096 (step 4) -> the last position on the queue
                             <1> ;;
                             <1> ;; Method:
                             <1> ;;     Swap page queue is a list of allocated pages with physical
                             <1> ;;     addresses (system mode virtual adresses = physical addresses).
                             <1> ;;     It is used for 'swap_in' and 'swap_out' procedures.
                             <1> ;;     When a new page is being allocated, swap queue is updated
                             <1> ;;     by 'swap_queue_shift' procedure, header of the queue (offset 0)
                             <1> ;;     is checked for 'accessed' flag. If the 1st page on the queue
                             <1> ;;     is 'accessed' or 'read only', it is dropped from the list;
                             <1> ;;     other pages from the 2nd to the last (in [swpq_last]) shifted
                             <1> ;;     to head then the 2nd page becomes the 1st and '[swpq_last]'
                             <1> ;;     offset value becomes it's previous offset value - 4.
                             <1> ;;     If the 1st page of the swap page queue is not 'accessed'
                             <1> ;;     the queue/list is not shifted.
                             <1> ;;     After the queue/list shift, newly allocated page is added
                             <1> ;;     to the tail of the queue at the [swpq_count*4] position.
                             <1> ;;     But, if [swpq_count] > 1023, the newly allocated page
                             <1> ;;     will not be added to the tail of swap page queue.
                             <1> ;;
                             <1> ;;     During 'swap_out' procedure, swap page queue is checked for
                             <1> ;;     the first non-accessed, writable page in the list,
                             <1> ;;     from the head to the tail. The list is shifted to left
                             <1> ;;     (to the head) till a non-accessed page will be found in the list.
                             <1> ;;     Then, this page   is swapped out (to disk) and then it is dropped
                             <1> ;;     from the list by a final swap queue shift. [swpq_count] value
                             <1> ;;     is changed. If all pages on the queue' are 'accessed',
                             <1> ;;     'insufficient memory' error will be returned ('swap_out'
                             <1> ;;     procedure will be failed)...
                             <1> ;;
                             <1> ;;     Note: If the 1st page of the queue is an 'accessed' page,
                             <1> ;;     'accessed' flag of the page will be reset (0) and that page
                             <1> ;;     (PTE) will be added to the tail of the queue after
                             <1> ;;     the check, if [swpq_count] < 1023. If [swpq_count] = 1024
                             <1> ;;     the queue will be rotated and the PTE in the head will be
                             <1> ;;     added to the tail after resetting 'accessed' bit.
                             <1> ;;
                             <1> ;;
                             <1> ;;
                             <1> ;; SWAP DISK/FILE (with 4096 bytes swapped page blocks)
                             <1> ;;
                             <1> ;;  00000000  00000004  00000008  0000000C   ...   size-8    size-4
                             <1> ;; +---------+---------+---------+---------+-- --+---------+---------+
                             <1> ;; |descriptr| page(1) | page(2) | page(3) | ... |page(n-1)| page(n) |
                             <1> ;; +---------+---------+---------+---------+-- --+---------+---------+
                             <1> ;;
                             <1> ;; [swpd_next] = the first free block address in swapped page records
                             <1> ;;            for next free block search by 'swap_out' procedure.
                             <1> ;; [swpd_size] = swap disk/file size in sectors (512 bytes)
                             <1> ;;            NOTE: max. possible swap disk size is 1024 GB
                             <1> ;;            (entire swap space must be accessed by using
                             <1> ;;            31 bit offset address)
                             <1> ;; [swpd_free] = free block (4096 bytes) count in swap disk/file space
                             <1> ;; [swpd_start] = absolute/start address of the swap disk/file
                             <1> ;;             0 for file, or beginning sector of the swap partition
                             <1> ;; [swp_drv] = logical drive description table addr. of swap disk/file
                             <1> ;;
                             <1> ;;
                             <1> ;; Method:
                             <1> ;;     When the memory (ram) becomes insufficient, page allocation
```

```
<1> ;;     procedure swaps out a page from memory to the swap disk
<1> ;;     (partition) or swap file to get a new free page at the memory.
<1> ;;     Swapping out is performed by using swap page queue.
<1> ;;
<1> ;;     Allocation block size of swap disk/file is equal to page size
<1> ;;     (4096 bytes). Swapping address (in sectors) is recorded
<1> ;;     into relevant page file entry as 31 bit physical (logical)
<1> ;;     offset address as 1 bit shifted to left for present flag (0).
<1> ;;     Swapped page address is between 1 and swap disk/file size - 4.
<1> ;;     Absolute physical (logical) address of the swapped page is
<1> ;;     calculated by adding offset value to the swap partition's
<1> ;;     start address. If the swap device (disk) is a virtual disk
<1> ;;     or it is a file, start address of the swap disk/volume is 0,
<1> ;;     and offset value is equal to absolute (physical or logical)
<1> ;;     address/position. (It has not to be ZERO if the swap partition
<1> ;;     is in a partitioned virtual hard disk.)
<1> ;;
<1> ;;     Note: Swap addresses are always specified/declared in sectors,
<1> ;;     not in bytes or   in blocks/zones/clusters (4096 bytes) as unit.
<1> ;;
<1> ;;     Swap disk/file allocation is mapped via 'Swap Allocation Table'
<1> ;;     at memory as similar to 'Memory Allocation Table'.
<1> ;;
<1> ;;     Every bit of Swap Allocation Table repsesents one swap block
<1> ;;     (equal to page size) respectively. Bit 0 of the S.A.T. byte 0
<1> ;;     is reserved for swap disk/file block 0 as descriptor block
<1> ;;     (also for compatibility with PTE). If bit value is ZERO,
<1> ;;     it means relevant (respective) block is in use, and,
<1> ;;     of course, if bit value is 1, it means relevant (respective)
<1> ;;      swap disk/file block is free.
<1> ;;     For example: bit 1 of the byte 128 repsesents block 1025
<1> ;;     (128*8+1) or sector (offset) 8200 on the swap disk or
<1> ;;     byte (offset/position) 4198400 in the swap file.
<1> ;;     4GB swap space is represented via 128KB Swap Allocation Table.
<1> ;;     Initial layout of Swap Allocation Table is as follows:
<1> ;;     ------------------------------------------------------------
<1> ;;     0111111111111111111111111 .... 11111111111111111111111111111
<1> ;;     ------------------------------------------------------------
<1> ;;     (0 is reserved block, 1s represent free blocks respectively.)
<1> ;;     (Note: Allocation cell/unit of the table is bit, not byte)
<1> ;;
<1> ;;     ..............................................................
<1> ;;
<1> ;;     'swap_out' procedure checks 'free_swap_blocks' count at first,
<1> ;;     then it searches Swap Allocation Table if free count is not
<1> ;;     zero. From begining the [swpd_next] dword value, the first bit
<1> ;;     position with value of 1 on the table is converted to swap
<1> ;;     disk/file offset address, in sectors (not 4096 bytes block).
<1> ;;     'ldrv_write' procedure is called with ldrv (logical drive
<1> ;;     number of physical swap disk or virtual swap disk)
<1> ;;     number, sector offset (not absolute sector -LBA- number),
<1> ;;     and sector count (8, 512*8 = 4096) and buffer adress
<1> ;;     (memory page). That will be a direct disk write procedure.
<1> ;;     (for preventing late memory allocation, significant waiting).
<1> ;;     If disk write procedure returns with error or free count of
<1> ;;     swap blocks is ZERO, 'swap_out' procedure will return with
<1> ;;     'insufficient memory error' (cf=1).
<1> ;;
<1> ;;     (Note: Even if free swap disk/file blocks was not zero,
<1> ;;     any disk write error will not be fixed by 'swap_out' procedure,
<1> ;;     in other words, 'swap_out' will not check the table for other
<1> ;;     free blocks after a disk write error. It will return to
<1> ;;     the caller with error (CF=1) which means swapping is failed.
<1> ;;
<1> ;;     After writing the page on to swap disk/file address/sector,
<1> ;;     'swap_out' procesure returns with that swap (offset) sector
<1> ;;     address (cf=0).
<1> ;;
<1> ;;     ..............................................................
<1> ;;
<1> ;;     'swap_in' procedure loads addressed (relevant) swap disk or
<1> ;;     file sectors at specified memory page. Then page allocation
<1> ;;     procedure updates relevant page table entry with 'present'
<1> ;;     attribute. If swap disk or file reading fails there is nothing
<1> ;;     to do, except to terminate the process which is the owner of
<1> ;;     the swapped page.
<1> ;;
<1> ;;     'swap_in' procedure sets the relevant/respective bit value
<1> ;;     in the Swap Allocation Table (as free block). 'swap_in' also
<1> ;;     updates [swpd_first] pointer if it is required.
<1> ;;
<1> ;;     ..............................................................
<1> ;;
<1> ;;     Note: If [swap_enabled] value is ZERO, that means there is not
<1> ;;     a swap disk or swap file in use... 'swap_in' and 'swap_out'
<1> ;;     procedures ans 'swap page que' procedures will not be active...
<1> ;;     'Insufficient memory' error will be returned by 'swap_out'
<1> ;;     and 'general protection fault' will be returned by 'swap_in'
<1> ;;     procedure, if it is called mistakenly (a wrong value in a PTE).
<1> ;;
<1>
<1> swap_in:
```

```
                              <1>  ; 31/08/2015
                              <1>  ; 20/07/2015
                              <1>  ; 28/04/2015
                              <1>  ; 18/04/2015
                              <1>  ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
                              <1>  ;
                              <1>  ; INPUT ->
                              <1>  ;     EBX = PHYSICAL (real/flat) ADDRESS OF THE MEMORY PAGE
                              <1>  ;     EBP = VIRTUAL (LINEAR) ADDRESS (page fault address)
                              <1>  ;     EAX = Offset Address for the swapped page on the
                              <1>  ;           swap disk or in the swap file.
                              <1>  ;
                              <1>  ; OUTPUT ->
                              <1>  ;     EAX = 0 if loading at memory has been successful
                              <1>  ;
                              <1>  ;     CF = 1 -> swap disk reading error (disk/file not present
                              <1>  ;             or sector not present or drive not ready
                              <1>  ;          EAX = Error code
                              <1>  ;          [u.error] = EAX
                              <1>  ;                  = The last error code for the process
                              <1>  ;                    (will be reset after returning to user)
                              <1>  ;
                              <1>  ; Modified Registers -> EAX
                              <1>  ;
                              <1>
00003470 833D[32810000]00     <1>         cmp     dword [swp_drv], 0
00003477 7648                 <1>  jna   short swpin_dnp_err
                              <1>
00003479 3B05[36810000]       <1>  cmp   eax, [swpd_size]
0000347F 734C                 <1>  jnb   short swpin_snp_err
                              <1>
00003481 56                   <1>  push  esi
00003482 53                   <1>  push  ebx
00003483 51                   <1>  push  ecx
00003484 8B35[32810000]       <1>  mov   esi, [swp_drv]
0000348A B908000000           <1>  mov   ecx, PAGE_SIZE / LOGIC_SECT_SIZE  ; 8 !
                              <1>        ; Note: Even if corresponding physical disk's sector
                              <1>        ; size different than 512 bytes, logical disk sector
                              <1>        ; size is 512 bytes and disk reading procedure
                              <1>        ; will be performed for reading 4096 bytes
                              <1>        ; (2*2048, 8*512).
                              <1>  ; ESI = Logical disk description table address
                              <1>  ; EBX = Memory page (buffer) address (physical!)
                              <1>  ; EAX = Sector adress (offset address, logical sector number)
                              <1>  ; ECX = Sector count ; 8 sectors
0000348F 50                   <1>  push  eax
00003490 E833020000           <1>  call  logical_disk_read
00003495 58                   <1>  pop   eax
00003496 730C                 <1>  jnc   short swpin_read_ok
                              <1>  ;
00003498 B804000000           <1>  mov   eax, SWP_DISK_READ_ERR ; drive not ready or read error
0000349D A3[CF740000]         <1>  mov   [u.error], eax
000034A2 EB19                 <1>  jmp   short swpin_retn
                              <1>  ;
                              <1> swpin_read_ok:
                              <1>  ; EAX = Offset address (logical sector number)
000034A4 E891010000           <1>  call  unlink_swap_block  ; Deallocate swap block
                              <1>  ;
                              <1>  ; EBX = Memory page (buffer) address (physical!)
                              <1>  ; 20/07/2015
000034A9 89EB                 <1>  mov   ebx, ebp ; virtual address (page fault address)
000034AB 6681E300F0           <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
000034B0 8A1D[C9740000]       <1>  mov   bl, [u.uno] ; current process number
                              <1>  ; EBX = Virtual address & process number combination
000034B6 E89E000000           <1>  call  swap_queue_shift
000034BB 29C0                 <1>  sub   eax, eax  ; 0 ; Error Code = 0  (no error)
                              <1>  ;
                              <1> swpin_retn:
000034BD 59                   <1>  pop   ecx
000034BE 5B                   <1>  pop   ebx
000034BF 5E                   <1>  pop   esi
000034C0 C3                   <1>  retn
                              <1>
                              <1> swpin_dnp_err:
000034C1 B805000000           <1>  mov   eax, SWP_DISK_NOT_PRESENT_ERR
                              <1> swpin_err_retn:
000034C6 A3[CF740000]         <1>  mov   [u.error], eax
000034CB F9                   <1>  stc
000034CC C3                   <1>  retn
                              <1>
                              <1> swpin_snp_err:
000034CD B806000000           <1>  mov   eax, SWP_SECTOR_NOT_PRESENT_ERR
000034D2 EBF2                 <1>  jmp   short swpin_err_retn
                              <1>
                              <1> swap_out:
                              <1>  ; 31/08/2015
                              <1>  ; 05/05/2015
                              <1>  ; 30/04/2015
                              <1>  ; 28/04/2015
                              <1>  ; 18/04/2015
                              <1>  ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
                              <1>  ;
```

```
                             <1>  ; INPUT ->
                             <1>  ;     none
                             <1>  ;
                             <1>  ; OUTPUT ->
                             <1>  ;     EAX = Physical page address (which is swapped out
                             <1>  ;           for allocating a new page)
                             <1>  ;     CF = 1 -> swap disk writing error (disk/file not present
                             <1>  ;             or sector not present or drive not ready
                             <1>  ;          EAX = Error code
                             <1>  ;          [u.error] = EAX
                             <1>  ;                  = The last error code for the process
                             <1>  ;                    (will be reset after returning to user)
                             <1>  ;
                             <1>  ; Modified Registers -> non (except EAX)
                             <1>  ;
000034D4 66833D[30810000]01  <1>  cmp   word [swpq_count], 1
000034DC 7274                <1>         jc      short swpout_im_err ; 'insufficient memory'
                             <1>
                             <1>         ;cmp     dword [swp_drv], 1
                             <1>  ;jc   short swpout_dnp_err ; 'swap disk/file not present'
                             <1>
000034DE 833D[3A810000]01    <1>         cmp     dword [swpd_free], 1
000034E5 7258                <1>  jc    short swpout_nfspc_err ; 'no free space on swap disk'
                             <1>
000034E7 53                  <1>  push  ebx
                             <1> swpout_1:
000034E8 31DB                <1>  xor   ebx, ebx
000034EA E86A000000          <1>  call  swap_queue_shift
000034EF 21C0                <1>  and   eax, eax    ; entry count (before shifting)
000034F1 7457                <1>  jz    short swpout_npts_err  ; There is no any PTE in
                             <1>                          ; the swap queue
000034F3 BB00E00800          <1>  mov   ebx, swap_queue          ; Addres of the head of
                             <1>                          ; the swap queue
000034F8 8B03                <1>  mov   eax, [ebx]         ; The PTE in the queue head
                             <1>
                             <1>  ;test al, PTE_A_PRESENT      ; bit 0 = 1
                             <1>  ;jz   short swpout_1          ; non-present page already
                             <1>                          ; must not be in the queue
                             <1>
                             <1>  ;test al, PTE_A_WRITE          ; bit 1 = 0
                             <1>  ;jz   short swpout_1          ; read only page (must not be
                             <1>                          ; swapped out)
                             <1>
000034FA A820                <1>  test  al, PTE_A_ACCESS       ; bit 5 = 1 (Accessed)
000034FC 75EA                <1>  jnz   short swpout_1          ; accessed page (must not be
                             <1>                          ; swapped out, at this stage)
                             <1>  ;
000034FE 662500F0            <1>  and   ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
                             <1>  ;
00003502 52                  <1>  push  edx
00003503 89DA                <1>  mov   edx, ebx           ; Page table entry address
00003505 89C3                <1>  mov   ebx, eax           ; Buffer (Page) Address
                             <1>  ;
00003507 E861010000          <1>  call  link_swap_block
0000350C 7304                <1>  jnc   short swpout_2          ; It may not be needed here
0000350E 5A                  <1>  pop   edx                ; because [swpd_free] value
0000350F 5B                  <1>  pop   ebx
00003510 EB2D                <1>  jmp   short swpout_nfspc_err ; was checked at the beginging.
                             <1> swpout_2:
00003512 56                  <1>  push  esi
00003513 51                  <1>  push  ecx
00003514 50                  <1>  push  eax ; sector address
00003515 8B35[32810000]      <1>  mov   esi, [swp_drv]
0000351B B908000000          <1>  mov   ecx, PAGE_SIZE / LOGIC_SECT_SIZE  ; 8 !
                             <1>        ; Note: Even if corresponding physical disk's sector
                             <1>        ; size different than 512 bytes, logical disk sector
                             <1>        ; size is 512 bytes and disk writing procedure
                             <1>        ; will be performed for writing 4096 bytes
                             <1>        ; (2*2048, 8*512).
                             <1>  ; ESI = Logical disk description table address
                             <1>  ; EBX = Buffer address
                             <1>  ; EAX = Sector adress (offset address, logical sector number)
                             <1>  ; ECX = Sector count ; 8 sectors
00003520 E8A4010000          <1>  call  logical_disk_write
00003525 59                  <1>  pop   ecx ; sector address
00003526 730C                <1>  jnc   short swpout_write_ok
                             <1>  ;
                             <1>  ;; call     unlink_swap_block ; this block must be left as 'in use'
                             <1> swpout_dw_err:
00003528 B808000000          <1>  mov   eax, SWP_DISK_WRITE_ERR ; drive not ready or write error
0000352D A3[CF740000]        <1>  mov   [u.error], eax
00003532 EB06                <1>  jmp   short swpout_retn
                             <1>  ;
                             <1> swpout_write_ok:
                             <1>  ; EBX = Buffer (page) address
                             <1>  ; EDX = Page Table entry address
                             <1>  ; ECX = Swap disk sector (file block) address (31 bit)
00003534 D1E1                <1>  shl   ecx, 1  ; 31 bit sector address from bit 1 to bit 31
00003536 890A                <1>  mov   [edx], ecx
                             <1>        ; bit 0 = 0 (swapped page)
00003538 89D8                <1>  mov   eax, ebx
                             <1> swpout_retn:
```

```
0000353A 59                  <1>  pop   ecx
0000353B 5E                  <1>  pop   esi
0000353C 5A                  <1>  pop   edx
0000353D 5B                  <1>  pop   ebx
0000353E C3                  <1>  retn
                             <1>
                             <1> ; Note: Swap_queue will not be updated in 'swap_out' procedure
                             <1> ;after the page is swapped out. (the PTE at the queue head
                             <1> ;-with 'non-present' attribute- will be dropped from the
                             <1> ;the queue in next 'swap_out' or in next 'swap_queue_shift'.
                             <1>
                             <1> ;swpout_dnp_err:
                             <1> ;mov   eax, SWP_DISK_NOT_PRESENT_ERR ; disk not present
                             <1> ;jmp   short swpout_err_retn
                             <1> swpout_nfspc_err:
0000353F B807000000          <1>  mov   eax, SWP_NO_FREE_SPACE_ERR ; no free space
                             <1> swpout_err_retn:
00003544 A3[CF740000]        <1>  mov   [u.error], eax
                             <1>  ;stc
00003549 C3                  <1>  retn
                             <1> swpout_npts_err:
0000354A B809000000          <1>  mov   eax, SWP_NO_PAGE_TO_SWAP_ERR
0000354F 5B                  <1>  pop   ebx
00003550 EBF2                <1>  jmp   short swpout_err_retn
                             <1> swpout_im_err:
00003552 B801000000          <1>  mov   eax, ERR_MINOR_IM ; insufficient (out of) memory
00003557 EBEB                <1>  jmp   short swpout_err_retn
                             <1>
                             <1> swap_queue_shift:
                             <1>  ; 20/07/2015
                             <1>  ; 28/04/2015
                             <1>  ; 18/04/2015
                             <1>  ; 23/10/2014 (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     EBX = Virtual (linear) address (bit 12 to 31)
                             <1>  ;           and process number combination (bit 0 to 11)
                             <1>  ;     EBX = 0 -> shift/drop from the head (offset 0)
                             <1>  ; OUTPUT ->
                             <1>  ;     If EBX input > 0
                             <1>  ;         the queue will be shifted 4 bytes (dword),
                             <1>  ;         from the tail to the head, up to entry offset
                             <1>  ;         which points to EBX input value or nothing
                             <1>  ;         to do if EBX value is not found in the queue.
                             <1>  ;         (The entry -with EBX value- will be removed
                             <1>  ;          from the queue if it is found.)
                             <1>  ;     If EBX input = 0
                             <1>  ;         the queue will be shifted 4 bytes (dword),
                             <1>  ;         from the tail to the head, if the PTE address
                             <1>  ;         in head of the queue is marked as "accessed"
                             <1>  ;         or it is marked as "non present".
                             <1>  ;         (If "accessed" flag of the PTE -in the head-
                             <1>  ;         is set -to 1-, it will be reset -to 0- and then,
                             <1>  ;         the queue will be rotated -without dropping
                             <1>  ;         the PTE from the queue-, for 4 bytes on head
                             <1>  ;         to tail direction. The PTE in the head will be
                             <1>  ;         moved in the tail, other PTEs will be shifted on
                             <1>  ;         head direction.)
                             <1>  ;
                             <1>  ;     EAX = [swpq_count] (before the shifting)
                             <1>  ;         (EAX = 0 -> next 'swap_out' stage
                             <1>  ;          is not applicable)
                             <1>  ;
                             <1>  ; Modified Registers -> EAX
                             <1>  ;
00003559 0FB705[30810000]    <1>  movzx   eax, word [swpq_count]  ; Max. 1024
00003560 6621C0              <1>  and   ax, ax
00003563 7433                <1>  jz    short swpqs_retn
00003565 57                  <1>  push  edi
00003566 56                  <1>  push  esi
00003567 53                  <1>  push  ebx
00003568 51                  <1>  push  ecx
00003569 50                  <1>  push  eax
0000356A BE00E00800          <1>  mov   esi, swap_queue
0000356F 89C1                <1>  mov   ecx, eax
00003571 09DB                <1>  or    ebx, ebx
00003573 7424                <1>  jz    short swpqs_7
                             <1> swpqs_1:
00003575 AD                  <1>  lodsd
00003576 39D8                <1>  cmp   eax, ebx
00003578 7404                <1>  je    short swpqs_2
0000357A E2F9                <1>  loop  swpqs_1
0000357C EB15                <1>  jmp   short swpqs_6
                             <1> swpqs_2:
0000357E 89F7                <1>  mov   edi, esi
00003580 83EF04              <1>  sub   edi, 4
                             <1> swpqs_3:
00003583 66FF0D[30810000]    <1>  dec   word [swpq_count]
0000358A 7403                <1>  jz    short swpqs_5
                             <1> swpqs_4:
0000358C 49                  <1>  dec   ecx
0000358D F3A5                <1>  rep   movsd ; shift up (to the head)
```

```
                            <1> swpqs_5:
0000358F 31C0               <1>  xor   eax, eax
00003591 8907               <1>  mov   [edi], eax
                            <1> swpqs_6:
00003593 58                 <1>  pop   eax
00003594 59                 <1>  pop   ecx
00003595 5B                 <1>  pop   ebx
00003596 5E                 <1>  pop   esi
00003597 5F                 <1>  pop   edi
                            <1> swpqs_retn:
00003598 C3                 <1>  retn
                            <1> swpqs_7:
00003599 89F7               <1>  mov   edi, esi ; head
0000359B AD                 <1>  lodsd
                            <1>  ; 20/07/2015
0000359C 89C3               <1>  mov   ebx, eax
0000359E 81E300F0FFFF       <1>  and   ebx, ~PAGE_OFF ; ~0FFFh
                            <1>              ; ebx = virtual address (at page boundary)
000035A4 25FF0F0000         <1>  and   eax, PAGE_OFF ; 0FFFh
                            <1>              ; ax = process number (1 to 4095)
000035A9 3A05[C9740000]     <1>  cmp   al, [u.uno]
                            <1>        ; Max. 16 (nproc) processes for Retro UNIX 386 v1
000035AF 7507               <1>  jne   short swpqs_8
000035B1 A1[D3740000]       <1>  mov   eax, [u.pgdir]
000035B6 EB16               <1>  jmp   short swpqs_9
                            <1> swpqs_8:
                            <1>  ;shl  ax, 2
000035B8 C0E002             <1>  shl   al, 2
000035BB 8B80[12720000]     <1>  mov   eax, [eax+p.upage-4]
000035C1 09C0               <1>  or    eax, eax
000035C3 74BE               <1>  jz    short swpqs_3 ; invalid upage
000035C5 83C05B             <1>  add   eax, u.pgdir - user
                            <1>               ; u.pgdir value for the process
                            <1>               ; is in [eax]
000035C8 8B00               <1>  mov   eax, [eax]
000035CA 21C0               <1>  and   eax, eax
000035CC 74B5               <1>  jz    short swpqs_3 ; invalid page directory
                            <1> swpqs_9:
000035CE 52                 <1>  push  edx
                            <1>  ; eax = page directory
                            <1>  ; ebx = virtual address
000035CF E87EFBFFFF         <1>  call  get_pte
000035D4 89D3               <1>  mov   ebx, edx ; PTE address
000035D6 5A                 <1>  pop   edx
000035D7 72AA               <1>  jc    short swpqs_3 ; empty PDE
                            <1>  ; EAX = PTE value
000035D9 A801               <1>  test  al, PTE_A_PRESENT ; bit 0 = 1
000035DB 74A6               <1>  jz    short swpqs_3 ; Drop non-present page
                            <1>                    ; from the queue (head)
000035DD A802               <1>  test  al, PTE_A_WRITE     ; bit 1 = 0
000035DF 74A2               <1>  jz    short swpqs_3 ; Drop read only page
                            <1>                    ; from the queue (head)
                            <1>  ;test al, PTE_A_ACCESS  ; bit 5 = 1 (Accessed)
                            <1>  ;jz   short swpqs_6 ; present
                            <1>                    ; non-accessed page
000035E1 0FBAF005           <1>         btr     eax, PTE_A_ACCESS_BIT ; reset 'accessed' bit
000035E5 73AC               <1>  jnc   short swpqs_6  ; non-accessed page
000035E7 8903               <1>  mov   [ebx], eax     ; save changed attribute
                            <1>  ;
                            <1>  ; Rotation (head -> tail)
000035E9 49                 <1>  dec   ecx     ; entry count -> last entry number
000035EA 74A7               <1>  jz    short swpqs_6
                            <1>        ; esi = head + 4
                            <1>        ; edi = head
000035EC 8B07               <1>  mov   eax, [edi] ; 20/07/2015
000035EE F3A5               <1>  rep   movsd  ; n = 1 to k-1, [n - 1] = [n]
000035F0 8907               <1>  mov   [edi], eax ; head -> tail ; [k] = [1]
000035F2 EB9F               <1>  jmp   short swpqs_6
                            <1>
                            <1> add_to_swap_queue:
                            <1> ; temporary - 16/09/2015
000035F4 C3                 <1> retn
                            <1>  ; 20/07/2015
                            <1>  ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
                            <1>  ;
                            <1>  ; Adds new page to swap queue
                            <1>  ; (page directories and page tables must not be added
                            <1>  ; to swap queue)
                            <1>  ;
                            <1>  ; INPUT ->
                            <1>  ;     EBX = Virtual address (for current process, [u.uno])
                            <1>  ;
                            <1>  ; OUTPUT ->
                            <1>  ;     EAX = [swpq_count]
                            <1>  ;           (after the PTE has been added)
                            <1>  ;     EAX = 0 -> Swap queue is full, (1024 entries)
                            <1>  ;           the pte could not be added.
                            <1>  ;
                            <1>  ; Modified Registers -> EAX
                            <1>  ;
000035F5 53                 <1>  push  ebx
000035F6 6681E300F0         <1>         and     bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
```

```
000035FB 8A1D[C9740000]          <1>  mov   bl, [u.uno] ; current process number
00003601 E853FFFFFF              <1>  call  swap_queue_shift ; drop from the queue if
                                 <1>                     ; it is already in the queue
                                 <1>       ; Then add it to the tail of the queue
00003606 0FB705[30810000]        <1>  movzx eax, word [swpq_count]
0000360D 663D0004                <1>  cmp   ax, 1024
00003611 7205                    <1>  jb    short atsq_1
00003613 6629C0                  <1>  sub   ax, ax
00003616 5B                      <1>  pop   ebx
00003617 C3                      <1>  retn
                                 <1> atsq_1:
00003618 56                      <1>  push  esi
00003619 BE00E00800              <1>  mov   esi, swap_queue
0000361E 6621C0                  <1>  and   ax, ax
00003621 740A                    <1>  jz    short atsq_2
00003623 66C1E002                <1>  shl   ax, 2 ; convert to offset
00003627 01C6                    <1>  add   esi, eax
00003629 66C1E802                <1>  shr   ax, 2
                                 <1> atsq_2:
0000362D 6640                    <1>  inc   ax
0000362F 891E                    <1>  mov   [esi], ebx ; Virtual address + [u.uno] combination
00003631 66A3[30810000]          <1>  mov   [swpq_count], ax
00003637 5E                      <1>  pop   esi
00003638 5B                      <1>  pop   ebx
00003639 C3                      <1>  retn
                                 <1>
                                 <1> unlink_swap_block:
                                 <1>  ; 15/09/2015
                                 <1>  ; 30/04/2015
                                 <1>  ; 18/04/2015
                                 <1>  ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
                                 <1>  ;
                                 <1>  ; INPUT ->
                                 <1>  ;     EAX = swap disk/file offset address
                                 <1>  ;           (bit 1 to bit 31)
                                 <1>  ; OUTPUT ->
                                 <1>  ;     [swpd_free] is increased
                                 <1>  ;     (corresponding SWAP DISK ALLOC. TABLE bit is SET)
                                 <1>  ;
                                 <1>  ; Modified Registers -> EAX
                                 <1>  ;
0000363A 53                      <1>  push  ebx
0000363B 52                      <1>  push  edx
                                 <1>  ;
0000363C C1E804                  <1>  shr   eax, SECTOR_SHIFT+1  ;3+1 ; shift sector address to
                                 <1>                         ; 3 bits right
                                 <1>                         ; to get swap block/page number
0000363F 89C2                    <1>  mov   edx, eax
                                 <1>  ; 15/09/2015
00003641 C1EA03                  <1>  shr   edx, 3                 ; to get offset to S.A.T.
                                 <1>                         ; (1 allocation bit = 1 page)
                                 <1>                         ; (1 allocation bytes = 8 pages)
00003644 80E2FC                  <1>  and   dl, 0FCh         ; clear lower 2 bits
                                 <1>                         ; (to get 32 bit position)
                                 <1>  ;
00003647 BB00000D00              <1>  mov   ebx, swap_alloc_table ; Swap Allocation Table address
0000364C 01D3                    <1>  add   ebx, edx
0000364E 83E01F                  <1>  and   eax, 1Fh         ; lower 5 bits only
                                 <1>                         ; (allocation bit position)
00003651 3B05[3E810000]          <1>  cmp   eax, [swpd_next]     ; is the new free block addr. lower
                                 <1>                         ; than the address in 'swpd_next' ?
                                 <1>                         ; (next/first free block value)
00003657 7305                    <1>  jnb   short uswpbl_1         ; no
00003659 A3[3E810000]            <1>  mov   [swpd_next], eax     ; yes
                                 <1> uswpbl_1:
0000365E 0FAB03                  <1>  bts   [ebx], eax       ; unlink/release/deallocate block
                                 <1>                         ; set relevant bit to 1.
                                 <1>                         ; set CF to the previous bit value
00003661 F5                      <1>  cmc                    ; complement carry flag
00003662 7206                    <1>  jc    short uswpbl_2         ; do not increase swfd_free count
                                 <1>                         ; if the block is already deallocated
                                 <1>                         ; before.
00003664 FF05[3A810000]          <1>         inc     dword [swpd_free]
                                 <1> uswpbl_2:
0000366A 5A                      <1>  pop   edx
0000366B 5B                      <1>  pop   ebx
0000366C C3                      <1>  retn
                                 <1>
                                 <1> link_swap_block:
                                 <1>  ; 01/07/2015
                                 <1>  ; 18/04/2015
                                 <1>  ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
                                 <1>  ;
                                 <1>  ; INPUT -> none
                                 <1>  ;
                                 <1>  ; OUTPUT ->
                                 <1>  ;     EAX = OFFSET ADDRESS OF THE ALLOCATED BLOCK (4096 bytes)
                                 <1>  ;           in sectors (corresponding
                                 <1>  ;           SWAP DISK ALLOCATION TABLE bit is RESET)
                                 <1>  ;
                                 <1>  ;     CF = 1 and EAX = 0
                                 <1>  ;              if there is not a free block to be allocated
```

```
                              <1>  ;
                              <1>  ; Modified Registers -> none (except EAX)
                              <1>  ;
                              <1>
                              <1>  ;mov  eax, [swpd_free]
                              <1>  ;and  eax, eax
                              <1>  ;jz   short out_of_swpspc
                              <1>  ;
0000366D 53                   <1>  push  ebx
0000366E 51                   <1>  push  ecx
                              <1>  ;
0000366F BB00000D00           <1>  mov   ebx, swap_alloc_table ; Swap Allocation Table offset
00003674 89D9                 <1>  mov   ecx, ebx
00003676 031D[3E810000]       <1>  add   ebx, [swpd_next] ; Free block searching starts from here
                              <1>                   ; next_free_swap_block >> 5
0000367C 030D[42810000]       <1>  add   ecx, [swpd_last] ; Free block searching ends here
                              <1>                   ; (total_swap_blocks - 1) >> 5
                              <1> lswbl_scan:
00003682 39CB                 <1>  cmp   ebx, ecx
00003684 770A                 <1>  ja    short lswbl_notfound
                              <1>  ;
00003686 0FBC03               <1>  bsf   eax, [ebx] ; Scans source operand for first bit set (1).
                              <1>                ; Clears ZF if a bit is found set (1) and
                              <1>                ; loads the destination with an index to
                              <1>                ; first set bit. (0 -> 31)
                              <1>                ; Sets ZF to 1 if no bits are found set.
                              <1>  ; 01/07/2015
00003689 751C                 <1>  jnz   short lswbl_found ; ZF = 0 -> a free block has been found
                              <1>              ;
                              <1>              ; NOTE:  a Swap Disk Allocation Table bit
                              <1>              ;      with value of 1 means
                              <1>              ;      the corresponding page is free
                              <1>              ;      (Retro UNIX 386 v1 feaure only!)
0000368B 83C304               <1>  add   ebx, 4
                              <1>              ; We return back for searching next page block
                              <1>              ; NOTE: [swpd_free] is not ZERO; so,
                              <1>              ;     we always will find at least 1 free block here.
0000368E EBF2                 <1>  jmp         short lswbl_scan
                              <1>  ;
                              <1> lswbl_notfound:
00003690 81E900000D00         <1>  sub   ecx, swap_alloc_table
00003696 890D[3E810000]       <1>  mov   [swpd_next], ecx ; next/first free page = last page
                              <1>                   ; (unlink_swap_block procedure will change it)
0000369C 31C0                 <1>  xor   eax, eax
0000369E A3[3A810000]         <1>  mov   [swpd_free], eax
000036A3 F9                   <1>  stc
                              <1> lswbl_ok:
000036A4 59                   <1>  pop   ecx
000036A5 5B                   <1>  pop   ebx
000036A6 C3                   <1>  retn
                              <1>  ;
                              <1> ;out_of_swpspc:
                              <1> ; stc
                              <1> ; retn
                              <1>
                              <1> lswbl_found:
000036A7 89D9                 <1>  mov   ecx, ebx
000036A9 81E900000D00         <1>  sub   ecx, swap_alloc_table
000036AF 890D[3E810000]       <1>  mov   [swpd_next], ecx ; Set first free block searching start
                              <1>                   ; address/offset (to the next)
000036B5 FF0D[3A810000]       <1>        dec     dword [swpd_free] ; 1 block has been allocated (X = X-1)
                              <1>  ;
000036BB 0FB303               <1>  btr   [ebx], eax   ; The destination bit indexed by the source value
                              <1>                   ; is copied into the Carry Flag and then cleared
                              <1>                   ; in the destination.
                              <1>                   ;
                              <1>                   ; Reset the bit which is corresponding to the
                              <1>                   ; (just) allocated block.
000036BE C1E105               <1>  shl   ecx, 5            ; (block offset * 32) + block index
000036C1 01C8                 <1>  add   eax, ecx     ; = block number
000036C3 C1E003               <1>  shl   eax, SECTOR_SHIFT ; 3, sector (offset) address of the block
                              <1>                   ; 1 block =  8 sectors
                              <1>  ;
                              <1>  ; EAX = offset address of swap disk/file sector (beginning of the block)
                              <1>  ;
                              <1>  ; NOTE: The relevant page table entry will be updated
                              <1>  ;       according to this EAX value...
                              <1>  ;
000036C6 EBDC                 <1>  jmp   short lswbl_ok
                              <1>
                              <1> logical_disk_read:
                              <1>  ; 20/07/2015
                              <1>  ; 09/03/2015 (temporary code here)
                              <1>  ;
                              <1>  ; INPUT ->
                              <1>  ;     ESI = Logical disk description table address
                              <1>  ;     EBX = Memory page (buffer) address (physical!)
                              <1>  ;     EAX = Sector adress (offset address, logical sector number)
                              <1>  ;     ECX = Sector count
                              <1>  ;
                              <1>  ;
000036C8 C3                   <1>  retn
```

```
                             <1>
                             <1> logical_disk_write:
                             <1>  ; 20/07/2015
                             <1>  ; 09/03/2015 (temporary code here)
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     ESI = Logical disk description table address
                             <1>  ;     EBX = Memory page (buffer) address (physical!)
                             <1>  ;     EAX = Sector adress (offset address, logical sector number)
                             <1>  ;     ECX = Sector count
                             <1>  ;
000036C9 C3                  <1>  retn
                             <1>
                             <1> get_physical_addr:
                             <1>  ; 18/10/2015
                             <1>  ; 29/07/2015
                             <1>  ; 20/07/2015
                             <1>  ; 04/06/2015
                             <1>  ; 20/05/2015
                             <1>  ; 28/04/2015
                             <1>  ; 18/04/2015
                             <1>  ; Get physical address
                             <1>  ;     (allocates a new page for user if it is not present)
                             <1>  ;
                             <1>  ; (This subroutine is needed for mapping user's virtual
                             <1>  ; (buffer) address to physical address (of the buffer).)
                             <1>  ; ('sys write', 'sys read' system calls...)
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     EBX = virtual address
                             <1>  ;     u.pgdir = page directory (physical) address
                             <1>  ;
                             <1>  ; OUTPUT ->
                             <1>  ;     EAX = physical address
                             <1>  ;     EBX = linear address
                             <1>  ;     EDX = physical address of the page frame
                             <1>  ;           (with attribute bits)
                             <1>  ;     ECX = byte count within the page frame
                             <1>  ;
                             <1>  ; Modified Registers -> EAX, EBX, ECX, EDX
                             <1>  ;
000036CA 81C300004000        <1>  add   ebx, CORE ; 18/10/2015
                             <1>  ;
000036D0 A1[D3740000]        <1>  mov   eax, [u.pgdir]
000036D5 E878FAFFFF          <1>  call  get_pte
                             <1>        ; EDX = Page table entry address (if CF=0)
                             <1>          ;       Page directory entry address (if CF=1)
                             <1>        ;       (Bit 0 value is 0 if PT is not present)
                             <1>        ; EAX = Page table entry value (page address)
                             <1>        ;     CF = 1 -> PDE not present or invalid ?
000036DA 731C                <1>  jnc   short gpa_1
                             <1>  ;
000036DC E856F9FFFF          <1>  call  allocate_page
000036E1 725B                <1>  jc    short gpa_im_err  ; 'insufficient memory' error
                             <1> gpa_0:
000036E3 E8C9F9FFFF          <1>  call  clear_page
                             <1>  ; EAX = Physical (base) address of the allocated (new) page
000036E8 0C07                <1>  or    al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER ; 4+2+1 = 7
                             <1>                 ; lower 3 bits are used as U/S, R/W, P flags
                             <1>                 ; (user, writable, present page)
000036EA 8902                <1>  mov   [edx], eax ; Let's put the new page directory entry here !
000036EC A1[D3740000]        <1>  mov   eax, [u.pgdir]
000036F1 E85CFAFFFF          <1>  call  get_pte
000036F6 7246                <1>  jc    short gpa_im_err ; 'insufficient memory' error
                             <1> gpa_1:
                             <1>  ; EAX = PTE value, EDX = PTE address
000036F8 A801                <1>  test  al, PTE_A_PRESENT
000036FA 751A                <1>  jnz   short gpa_3
000036FC 09C0                <1>  or    eax, eax
000036FE 7430                <1>  jz    short gpa_4  ; Allocate a new page
                             <1>  ; 20/07/2015
00003700 55                  <1>  push  ebp
00003701 89DD                <1>  mov   ebp, ebx ; virtual (linear) address
                             <1>  ; reload swapped page
00003703 E83C000000          <1>  call  reload_page ; 28/04/2015
00003708 5D                  <1>  pop   ebp
00003709 7224                <1>  jc    short gpa_retn
                             <1> gpa_2:
                             <1>  ; 20/07/2015
                             <1>  ; 20/05/2015
                             <1>  ; add this page to swap queue
0000370B 50                  <1>  push  eax
                             <1>  ; EBX = virtual address
0000370C E8E3FEFFFF          <1>  call  add_to_swap_queue
00003711 58                  <1>  pop   eax
                             <1>        ; PTE address in EDX
                             <1>        ; virtual address in EBX
                             <1>  ; EAX = memory page address
00003712 0C07                <1>  or    al, PTE_A_PRESENT + PTE_A_USER + PTE_A_WRITE
                             <1>                      ; present flag, bit 0 = 1
                             <1>                      ; user flag, bit 2 = 1
                             <1>                      ; writable flag, bit 1 = 1
```

```
00003714 8902                <1>  mov   [edx], eax  ; Update PTE value
                             <1> gpa_3:
                             <1>  ; 18/10/2015
00003716 89D9                <1>  mov   ecx, ebx
00003718 81E1FF0F0000        <1>  and   ecx, PAGE_OFF
0000371E 89C2                <1>  mov   edx, eax
00003720 662500F0            <1>  and   ax, PTE_A_CLEAR
00003724 01C8                <1>  add   eax, ecx
00003726 F7D9                <1>  neg   ecx ; 1 -> -1 (0FFFFFFFFh), 4095 (0FFFh) -> -4095
00003728 81C100100000        <1>  add   ecx, PAGE_SIZE
0000372E F8                  <1>  clc
                             <1> gpa_retn:
0000372F C3                  <1>  retn
                             <1> gpa_4:
00003730 E802F9FFFF          <1>  call  allocate_page
00003735 7207                <1>  jc    short gpa_im_err ; 'insufficient memory' error
00003737 E875F9FFFF          <1>  call  clear_page
0000373C EBCD                <1>  jmp   short gpa_2
                             <1>
                             <1> gpa_im_err:
0000373E B801000000          <1>  mov   eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
                             <1>                      ; Major error = 0 (No protection fault)
00003743 C3                  <1>  retn
                             <1>
                             <1> reload_page:
                             <1>  ; 20/07/2015
                             <1>  ; 28/04/2015 (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; Reload (Restore) swapped page at memory
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     EBP = Virtual (linear) memory address
                             <1>  ;     EAX = PTE value (swap disk sector address)
                             <1>  ;     (Swap disk sector address = bit 1 to bit 31 of EAX)
                             <1>  ; OUTPUT ->
                             <1>  ;     EAX = PHYSICAL (real/flat) ADDRESS OF RELOADED PAGE
                             <1>  ;
                             <1>  ;     CF = 1 and EAX = error code
                             <1>  ;
                             <1>  ; Modified Registers -> none (except EAX)
                             <1>  ;
00003744 D1E8                <1>  shr   eax, 1   ; Convert PTE value to swap disk address
00003746 53                  <1>  push  ebx      ;
00003747 89C3                <1>  mov   ebx, eax ; Swap disk (offset) address
00003749 E8E9F8FFFF          <1>  call  allocate_page
0000374E 720C                <1>  jc    short rlp_im_err
00003750 93                  <1>  xchg  eax, ebx
                             <1>  ; EBX = Physical memory (page) address
                             <1>  ; EAX = Swap disk (offset) address
                             <1>  ; EBP = Virtual (linear) memory address
00003751 E81AFDFFFF          <1>  call  swap_in
00003756 720B                <1>  jc    short rlp_swp_err  ; (swap disk/file read error)
00003758 89D8                <1>  mov   eax, ebx
                             <1> rlp_retn:
0000375A 5B                  <1>  pop   ebx
0000375B C3                  <1>  retn
                             <1>
                             <1> rlp_im_err:
0000375C B801000000          <1>  mov   eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
                             <1>                      ; Major error = 0 (No protection fault)
00003761 EBF7                <1>  jmp   short rlp_retn
                             <1>
                             <1> rlp_swp_err:
00003763 B804000000          <1>  mov   eax, SWP_DISK_READ_ERR ; Swap disk read error !
00003768 EBF0                <1>  jmp   short rlp_retn
                             <1>
                             <1>
                             <1> copy_page_dir:
                             <1>  ; 19/09/2015
                             <1>  ; temporary - 07/09/2015
                             <1>  ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
                             <1>  ;
                             <1>  ; INPUT ->
                             <1>  ;     [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
                             <1>  ;               page directory.
                             <1>  ; OUTPUT ->
                             <1>  ;     EAX =  PHYSICAL (real/flat) ADDRESS of the child's
                             <1>  ;            page directory.
                             <1>  ;     (New page directory with new page table entries.)
                             <1>  ;     (New page tables with read only copies of the parent's
                             <1>  ;     pages.)
                             <1>  ;     EAX = 0 -> Error (CF = 1)
                             <1>  ;
                             <1>  ; Modified Registers -> none (except EAX)
                             <1>  ;
0000376A E8C8F8FFFF          <1>  call  allocate_page
0000376F 723E                <1>  jc    short cpd_err
                             <1>  ;
00003771 55                  <1>  push  ebp ; 20/07/2015
00003772 56                  <1>  push  esi
00003773 57                  <1>  push  edi
00003774 53                  <1>  push  ebx
```

```
00003775 51                   <1>  push  ecx
00003776 8B35[D3740000]       <1>  mov   esi, [u.pgdir]
0000377C 89C7                 <1>  mov   edi, eax
0000377E 50                   <1>  push  eax ; save child's page directory address
                              <1>  ; copy PDE 0 from the parent's page dir to the child's page dir
                              <1>  ; (use same system space for all user page tables)
0000377F A5                   <1>  movsd
00003780 BD00004000           <1>  mov   ebp, 1024*4096 ; pass the 1st 4MB (system space)
00003785 B9FF030000           <1>  mov   ecx, (PAGE_SIZE / 4) - 1 ; 1023
                              <1> cpd_0:
0000378A AD                   <1>  lodsd
                              <1>  ;or   eax, eax
                              <1>         ;jnz     short cpd_1
0000378B A801                 <1>  test  al, PDE_A_PRESENT ;  bit 0 =  1
0000378D 7508                 <1>  jnz   short cpd_1
                              <1>  ; (virtual address at the end of the page table)
0000378F 81C500004000         <1>  add   ebp, 1024*4096 ; page size * PTE count
00003795 EB0F                 <1>  jmp   short cpd_2
                              <1> cpd_1:
00003797 662500F0             <1>  and   ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
0000379B 89C3                 <1>  mov   ebx, eax
                              <1>  ; EBX = Parent's page table address
0000379D E81F000000           <1>  call  copy_page_table
000037A2 720C                 <1>  jc    short cpd_p_err
                              <1>  ; EAX = Child's page table address
000037A4 0C07                 <1>  or    al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
                              <1>               ; set bit 0, bit 1 and bit 2 to 1
                              <1>               ; (present, writable, user)
                              <1> cpd_2:
000037A6 AB                   <1>  stosd
000037A7 E2E1                 <1>  loop  cpd_0
                              <1>  ;
000037A9 58                   <1>  pop   eax  ; restore child's page directory address
                              <1> cpd_3:
000037AA 59                   <1>  pop   ecx
000037AB 5B                   <1>  pop   ebx
000037AC 5F                   <1>  pop   edi
000037AD 5E                   <1>  pop   esi
000037AE 5D                   <1>  pop   ebp
                              <1> cpd_err:
000037AF C3                   <1>  retn
                              <1> cpd_p_err:
                              <1>  ; release the allocated pages missing (recover free space)
000037B0 58                   <1>  pop   eax  ; the new page directory address (physical)
000037B1 8B1D[D3740000]       <1>  mov   ebx, [u.pgdir] ; parent's page directory address
000037B7 E8B4F9FFFF           <1>  call  deallocate_page_dir
000037BC 29C0                 <1>  sub   eax, eax ; 0
000037BE F9                   <1>  stc
000037BF EBE9                 <1>  jmp   short cpd_3
                              <1>
                              <1> copy_page_table:
                              <1>  ; 19/09/2015
                              <1>  ; temporary - 07/09/2015
                              <1>  ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
                              <1>  ;
                              <1>  ; INPUT ->
                              <1>  ;     EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
                              <1>  ;     EBP = page table entry index (from 'copy_page_dir')
                              <1>  ; OUTPUT ->
                              <1>  ;     EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
                              <1>  ;     EBP = (recent) page table index (for 'add_to_swap_queue')
                              <1>  ;     CF = 1 -> error
                              <1>  ;
                              <1>  ; Modified Registers -> EBP (except EAX)
                              <1>  ;
000037C1 E871F8FFFF           <1>  call  allocate_page
000037C6 725A                 <1>  jc    short cpt_err
                              <1>  ;
000037C8 50                   <1>  push  eax ; *
                              <1>  ;push       ebx
000037C9 56                   <1>  push  esi
000037CA 57                   <1>  push  edi
000037CB 52                   <1>  push  edx
000037CC 51                   <1>  push  ecx
                              <1>  ;
000037CD 89DE                 <1>  mov   esi, ebx
000037CF 89C7                 <1>  mov   edi, eax
000037D1 89C2                 <1>  mov   edx, eax
000037D3 81C200100000         <1>  add   edx, PAGE_SIZE
                              <1> cpt_0:
000037D9 AD                   <1>  lodsd
000037DA A801                 <1>  test  al, PTE_A_PRESENT ;  bit 0 =  1
000037DC 750B                 <1>  jnz   short cpt_1
000037DE 21C0                 <1>  and   eax, eax
000037E0 7430                 <1>  jz    short cpt_2
                              <1>  ; ebp = virtual (linear) address of the memory page
000037E2 E85DFFFFFF           <1>  call  reload_page ; 28/04/2015
000037E7 7234                 <1>  jc    short cpt_p_err
                              <1> cpt_1:
000037E9 662500F0             <1>  and   ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
000037ED 89C1                 <1>  mov   ecx, eax
                              <1>  ; Allocate a new page for the child process
```

```
000037EF E843F8FFFF          <1>  call  allocate_page
000037F4 7227                <1>  jc    short cpt_p_err
000037F6 57                  <1>  push  edi
000037F7 56                  <1>  push  esi
000037F8 89CE                <1>  mov   esi, ecx
000037FA 89C7                <1>  mov   edi, eax
000037FC B900040000          <1>  mov   ecx, PAGE_SIZE/4
00003801 F3A5                <1>  rep   movsd ; copy page (4096 bytes)
00003803 5E                  <1>  pop   esi
00003804 5F                  <1>  pop   edi
                             <1>  ;
00003805 53                  <1>  push  ebx
00003806 50                  <1>  push  eax
00003807 89EB                <1>  mov   ebx, ebp
                             <1>  ; ebx = virtual address of the memory page
00003809 E8E6FDFFFF          <1>  call  add_to_swap_queue
0000380E 58                  <1>  pop   eax
0000380F 5B                  <1>  pop   ebx
                             <1>  ;
                             <1>  ;or   ax, PTE_A_USER+PTE_A_PRESENT
00003810 0C07                <1>  or    al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
                             <1> cpt_2:
00003812 AB                  <1>  stosd  ; EDI points to child's PTE
                             <1>  ;
00003813 81C500100000        <1>  add   ebp, 4096 ; 20/07/2015 (next page)
                             <1>  ;
00003819 39D7                <1>  cmp   edi, edx
0000381B 72BC                <1>  jb    short cpt_0
                             <1> cpt_p_err:
0000381D 59                  <1>  pop   ecx
0000381E 5A                  <1>  pop   edx
0000381F 5F                  <1>  pop   edi
00003820 5E                  <1>  pop   esi
                             <1>  ;pop  ebx
00003821 58                  <1>  pop   eax ; *
                             <1> cpt_err:
00003822 C3                  <1>  retn
                             <1>
                             <1>
                             <1> ; /// End Of MEMORY MANAGEMENT FUNCTIONS ///
                             <1>
                             <1> ;; Data:
                             <1>
                             <1> ; 09/03/2015
                             <1> ;swpq_count: dw 0 ; count of pages on the swap que
                             <1> ;swp_drv:    dd 0 ; logical drive description table address of the swap drive/disk
                             <1> ;swpd_size:  dd 0 ; size of swap drive/disk (volume) in sectors (512 bytes).

                             <1> ;swpd_free:  dd 0 ; free page blocks (4096 bytes) on swap disk/drive (logical)
                             <1> ;swpd_next:  dd 0 ; next free page block
                             <1> ;swpd_last:  dd 0 ; last swap page block
                                 %include 'sysdefs.inc' ; 09/03/2015
                             <1> ; Retro UNIX 386 v1 Kernel - SYSDEFS.INC
                             <1> ; Last Modification: 09/12/2015
                             <1> ;
                             <1> ; ///////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS //////////////
                             <1> ; (Modified from
                             <1> ; Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)
                             <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))
                             <1> ;      UNIX.ASM (MASM 6.11) --> SYSDEFS.INC (NASM 2.11)
                             <1> ; ---------------------------------------------------------------------------
                             <1> ;
                             <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                             <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                             <1> ; <Bell Laboratories (17/3/1972)>
                             <1> ; <Preliminary Release of UNIX Implementation Document>
                             <1> ;
                             <1> ; ***************************************************************************
                             <1>
                             <1> nproc  equ   16  ; number of processes
                             <1> nfiles equ   50
                             <1> ntty   equ     8   ; 8+1 -> 8 (10/05/2013)
                             <1> nbuf   equ   4   ; 6 ;; 21/08/2015 - 'namei' buffer problem when nbuf > 4
                             <1>        ; NOTE: If fd0 super block buffer addres is beyond of the 1st
                             <1>        ; 32K, DMA r/w routine or someting else causes a jump to
                             <1>        ; kernel panic routine (in 'alloc' routine, in u5.s)
                             <1>        ; because of invalid buffer content (r/w error).
                             <1>        ; When all buffers are set before the end of the 1st 32k,
                             <1>        ; there is no problem!? (14/11/2015)
                             <1>
                             <1> ;csgmntequ   2000h ; 26/05/2013 (segment of process 1)
                             <1> ;core   equ   0         ; 19/04/2013
                             <1> ;ecore equ   32768 - 64  ; 04/06/2013 (24/05/2013)
                             <1>  ; (if total size of argument list and arguments is 128 bytes)
                             <1>  ; maximum executable file size = 32768-(64+40+128-6) = 32530 bytes
                             <1>  ; maximum stack size = 40 bytes (+6 bytes for 'IRET' at 32570)
                             <1>  ; initial value of user's stack pointer = 32768-64-128-2 = 32574
                             <1>  ;     (sp=32768-args_space-2 at the beginning of execution)
                             <1>  ; argument list offset = 32768-64-128 = 32576 (if it is 128 bytes)
                             <1>  ; 'u' structure offset (for the '/core' dump file) = 32704
                             <1>  ; '/core' dump file size = 32768 bytes
                             <1>
```

```
<1> ; 08/03/2014
<1> ;sdsegmnt equ6C0h  ; 256*16 bytes (swap data segment size for 16 processes)

<1> ; 19/04/2013 Retro UNIX 8086 v1 feaure only !
<1> ;;sdsegmnt equ     740h  ; swap data segment (for user structures and registers)
<1>
<1> ; 30/08/2013
<1> time_count equ 4 ; 10 --> 4 01/02/2014
<1>
<1> ; 05/02/2014
<1> ; process status
<1> ;SFREE equ 0
<1> ;SRUN  equ 1
<1> ;SWAIT equ 2
<1> ;SZOMB equ 3
<1> ;SSLEEPequ 4 ; Retro UNIX 8086 V1 extension (for sleep and wakeup)
<1>
<1> ; 09/03/2015
<1> userdata equ 80000h ; user structure data address for current user ; temporary
<1> swap_queue equ 90000h - 2000h ; swap queue address ; temporary
<1> swap_alloc_table equ 0D0000h  ;  swap allocation table address ; temporary
<1>
<1> ; 17/09/2015
<1> ESPACE equ 48 ; [u.usp] (at 'sysent') - [u.sp] value for error return
<1>
<1> ; 21/09/2015 (36)
<1> ; 01/07/2015 (35)
<1> ; 14/07/2013 (0-34)
<1> ; UNIX v1 system calls
<1> _rele  equ 0
<1> _exit  equ 1
<1> _fork  equ 2
<1> _read  equ 3
<1> _write equ 4
<1> _open  equ 5
<1> _close equ 6
<1> _wait  equ 7
<1> _creat equ 8
<1> _link  equ 9
<1> _unlinkequ 10
<1> _exec  equ 11
<1> _chdir equ 12
<1> _time  equ 13
<1> _mkdir equ 14
<1> _chmod equ 15
<1> _chown equ 16
<1> _break equ 17
<1> _stat  equ 18
<1> _seek  equ 19
<1> _tell  equ 20
<1> _mount equ 21
<1> _umountequ 22
<1> _setuidequ 23
<1> _getuidequ 24
<1> _stime equ 25
<1> _quit  equ 26
<1> _intr  equ 27
<1> _fstat equ 28
<1> _emt   equ 29
<1> _mdate equ 30
<1> _stty  equ 31
<1> _gtty  equ 32
<1> _ilginsequ 33
<1> _sleep equ 34 ; Retro UNIX 8086 v1 feature only !
<1> _msg   equ 35 ; Retro UNIX 386 v1 feature only !
<1> _geterrequ 36 ; Retro UNIX 386 v1 feature only !
<1>
<1> %macro sys 1-4
<1>     ; 03/09/2015
<1>     ; 13/04/2015
<1>     ; Retro UNIX 386 v1 system call.
<1>     %if %0 >= 2
<1>         mov ebx, %2
<1>         %if %0 >= 3
<1>             mov ecx, %3
<1>             %if %0 = 4
<1>                mov edx, %4
<1>             %endif
<1>         %endif
<1>     %endif
<1>     mov eax, %1
<1>     int 30h
<1> %endmacro
<1>
<1> ; 13/05/2015 - ERROR CODES
<1> ERR_FILE_NOT_OPEN  equ 10 ; 'file not open !' error
<1> ERR_FILE_ACCESS    equ 11 ; 'permission denied !' error
<1> ; 14/05/2015
<1> ERR_DIR_ACCESS     equ 11 ; 'permission denied !' error
<1> ERR_FILE_NOT_FOUND equ 12 ; 'file not found !' error
<1> ERR_TOO_MANY_FILES equ 13 ; 'too many open files !' error
<1> ERR_DIR_EXISTS     equ 14 ; 'directory already exists !' error
```

```
                             <1> ; 16/05/2015
                             <1> ERR_DRV_NOT_RDY    equ 15 ; 'drive not ready !' error
                             <1> ; 18/05/2015
                             <1> ERR_DEV_NOT_RDY    equ 15 ; 'device not ready !' error
                             <1> ERR_DEV_ACCESS     equ 11 ; 'permission denied !' error
                             <1> ERR_DEV_NOT_OPEN   equ 10 ; 'device not open !' error
                             <1> ; 07/06/2015
                             <1> ERR_FILE_EOF    equ 16 ; 'end of file !' error
                             <1> ERR_DEV_VOL_SIZE   equ 16 ; 'out of volume' error
                             <1> ; 09/06/2015
                             <1> ERR_DRV_READ    equ 17 ; 'disk read error !'
                             <1> ERR_DRV_WRITE   equ 18 ; 'disk write error !'
                             <1> ; 16/06/2015
                             <1> ERR_NOT_DIR     equ 19 ; 'not a (valid) directory !' error
                             <1> ERR_FILE_SIZE   equ 20 ; 'file size error !'
                             <1> ; 22/06/2015
                             <1> ERR_NOT_SUPERUSER  equ 11 ; 'permission denied !' error
                             <1> ERR_NOT_OWNER      equ 11 ; 'permission denied !' error
                             <1> ERR_NOT_FILE       equ 11 ; 'permission denied !' error
                             <1> ; 23/06/2015
                             <1> ERR_FILE_EXISTS    equ 14 ; 'file already exists !' error
                             <1> ERR_DRV_NOT_SAME   equ 21 ; 'not same drive !' error
                             <1> ERR_DIR_NOT_FOUND  equ 12 ; 'directory not found !' error
                             <1> ERR_NOT_EXECUTABLE equ 22 ; 'not executable file !' error
                             <1> ; 27/06/2015
                             <1> ERR_INV_PARAMETER  equ 23 ; 'invalid parameter !' error
                             <1> ERR_INV_DEV_NAME   equ 24 ; 'invalid device name !' error
                             <1> ; 29/06/2015
                             <1> ERR_TIME_OUT    equ 25 ; 'time out !' error
                             <1> ERR_DEV_NOT_RESP   equ 25 ; 'device not responding !' error
                             <1>
                             <1> ; 26/08/2015
                             <1> ; 24/07/2015
                             <1> ; 24/06/2015
                             <1> MAX_ARG_LEN     equ 256 ; max. length of sys exec arguments
                             <1> ; 01/07/2015
                             <1> MAX_MSG_LEN     equ 255 ; max. msg length for 'sysmsg'
                             <1> ;
                                 %include 'u0.s'        ; 15/03/2015
                             <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS0.INC
                             <1> ; Last Modification: 21/11/2015
                             <1> ; -----------------------------------------------------------------------------
                             <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
                             <1> ; (v0.1 - Beginning: 11/07/2012)
                             <1> ;
                             <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                             <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                             <1> ; <Bell Laboratories (17/3/1972)>
                             <1> ; <Preliminary Release of UNIX Implementation Document>
                             <1> ;
                             <1> ; Retro UNIX 8086 v1 - U0.ASM (28/07/2014) //// UNIX v1 -> u0.s
                             <1> ;
                             <1> ; *****************************************************************************
                             <1>
                             <1> sys_init:
                             <1>  ; 18/10/2015
                             <1>  ; 28/08/2015
                             <1>  ; 24/08/2015
                             <1>  ; 14/08/2015
                             <1>  ; 24/07/2015
                             <1>  ; 02/07/2015
                             <1>  ; 01/07/2015
                             <1>  ; 23/06/2015
                             <1>  ; 15/04/2015
                             <1>  ; 13/04/2015
                             <1>  ; 11/03/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 28/07/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ;call ldrv_init ; Logical drive description tables initialization
                             <1>  ;
                             <1>  ; 14/02/2014
                             <1>  ; 14/07/2013
00003823 66B82900            <1>  mov   ax, 41
00003827 66A3[70740000]      <1>  mov   [rootdir], ax
0000382D 66A3[84740000]      <1>  mov   [u.cdir], ax
00003833 2401                <1>  and   al, 1 ; 15/04/2015
00003835 A2[C9740000]        <1>  mov   [u.uno], al
0000383A 66A3[6E740000]      <1>  mov   [mpid], ax
00003840 66A3[76710000]      <1>  mov   [p.pid], ax
00003846 A2[06720000]        <1>  mov   [p.stat], al ; SRUN, 05/02/2014
                             <1>  ;
0000384B B004                <1>  mov   al, time_count ; 30/08/2013
0000384D A2[BC740000]        <1>  mov   [u.quant], al ; 14/07/2013
                             <1>  ; 02/07/2015
00003852 A1[A8700000]        <1>  mov   eax, [k_page_dir]
                             <1>  ;sub  eax, eax
00003857 A3[D3740000]        <1>  mov   [u.pgdir], eax ; reset
                             <1>  ; 18/10/2015
                             <1>  ;mov  [u.ppgdir], eax ; 0
                             <1>         ;
0000385C E856030000          <1>  call  epoch
00003861 A3[B47E0000]        <1>  mov   [s.time], eax ; 13/03/2015
```

```
                             <1>  ; 17/07/2013
00003866 E8B0060000          <1>  call  bf_init ; buffer initialization
                             <1>  ; 23/06/2015
0000386B E8C7F7FFFF          <1>  call  allocate_page
                             <1>  ;;jc  error
00003870 0F829C000000        <1>         jc     panic   ; jc short panic (01/07/2015)
00003876 A3[CA740000]        <1>  mov   [u.upage], eax ; user structure page
0000387B A3[16720000]        <1>  mov   [p.upage], eax
                             <1>  ;
00003880 E82CF8FFFF          <1>  call  clear_page
                             <1>  ;
                             <1>  ; 14/08/2015
00003885 FA                  <1>  cli
                             <1>  ; 14/03/2015
                             <1>  ; 17/01/2014
00003886 E8DE010000          <1>  call  sp_init ; serial port initialization
                             <1>  ; 14/08/2015
0000388B FB                  <1>  sti
                             <1>  ;
                             <1>  ; 30/06/2015
                             <1>  ;mov  esi, kernel_init_ok_msg
                             <1>  ;call       print_msg
                             <1>  ;
0000388C 30DB                <1>  xor   bl, bl ; video page 0
                             <1> vp_clr_nxt:  ; clear video pages (reset cursor positions)
0000388E E8FE2D0000          <1>  call  vp_clr  ; 17/07/2013
00003893 FEC3                <1>  inc   bl
00003895 80FB08              <1>  cmp   bl, 8
00003898 72F4                <1>  jb    short vp_clr_nxt
                             <1>  ;
                             <1>  ; 24/07/2015
                             <1>  ;push    KDATA
                             <1>        ;push    esp
                             <1>  ;mov  [tss.esp0], esp
                             <1>         ;mov     word [tss.ss0], KDATA
                             <1>  ;
                             <1>  ; 24/08/2015
                             <1>  ;; temporary (01/07/2015)
0000389A C605[BC740000]04    <1>  mov   byte [u.quant], time_count ; 4
                             <1>                    ; it is not needed here !
                             <1>  ;;inc byte [u.kcall] ; 'the caller is kernel' sign
000038A1 FE0D[77740000]      <1>  dec   byte [sysflg] ; FFh = ready for system call
                             <1>                   ; 0 = executing a system call
                             <1>  ;;sys       _msg, kernel_init_ok_msg, 255, 0
                             <1>  ;
                             <1>  ;;; 06/08/2015
                             <1>  ;;;call     getch ; wait for a key stroke
                             <1>  ;;mov       ecx, 0FFFFFFFh
                             <1> ;;sys_init_msg_wait:
                             <1> ;;     push  ecx
                             <1> ;;     mov   al, 1
                             <1> ;;     mov   ah, [ptty] ; active (current) video page
                             <1> ;;     call  getc_n
                             <1> ;;     pop   ecx
                             <1> ;;     jnz   short sys_init_msg_ok
                             <1> ;;     loop  sys_init_msg_wait
                             <1>  ;
                             <1> ;;sys_init_msg_ok:
                             <1>  ; 28/08/2015 (initial settings for the 1st 'rswap')
000038A7 6A10                <1>  push  KDATA ; ss
000038A9 54                  <1>  push  esp
000038AA 9C                  <1>  pushfd
000038AB 6A08                <1>  push  KCODE ; cs
000038AD 68[E1380000]        <1>  push  init_exec ; eip
000038B2 8925[78740000]      <1>  mov   [u.sp], esp
000038B8 1E                  <1>  push  ds
000038B9 06                  <1>  push  es
000038BA 0FA0                <1>  push  fs
000038BC 0FA8                <1>  push  gs
000038BE 60                  <1>  pushad
000038BF 8925[7C740000]      <1>  mov   [u.usp], esp
000038C5 E8481B0000          <1>  call  wswap ; save current user (u) structure, user registers
                             <1>              ; and interrupt return components (for IRET)
000038CA 61                  <1>  popad
000038CB 6658                <1>  pop   ax ; gs
000038CD 6658                <1>  pop   ax ; fs
000038CF 6658                <1>  pop   ax ; es
000038D1 6658                <1>  pop   ax ; ds
000038D3 58                  <1>  pop   eax ; eip (init_exec)
000038D4 6658                <1>  pop   ax ; cs (KCODE)
000038D6 58                  <1>  pop   eax ; E-FLAGS
000038D7 58                  <1>  pop   eax ; esp
000038D8 6658                <1>  pop   ax ; ss (KDATA)
                             <1>  ;
000038DA 31C0                <1>  xor   eax, eax ; 0
000038DC A3[D7740000]        <1>  mov   [u.ppgdir], eax ; reset (to zero) for '/etc/init'
                             <1>  ;
                             <1>  ; 02/07/2015
                             <1>  ; [u.pgdir ] = [k_page_dir]
                             <1>  ; [u.ppgdir] = 0 (page dir of the parent process)
                             <1>  ;      (The caller is os kernel sign for 'sysexec')
                             <1> init_exec:
```

```
                             <1>  ; 13/03/2013
                             <1>  ; 24/07/2013
000038E1 BB[08390000]        <1>  mov   ebx, init_file
000038E6 B9[00390000]        <1>  mov   ecx, init_argp
                             <1>  ; EBX contains 'etc/init' asciiz file name address
                             <1>  ; ECX contains address of argument list pointer
                             <1>  ;
                             <1>  ;dec  byte [sysflg] ; FFh = ready for system call
                             <1>                      ; 0 = executing a system call
                             <1>  sys   _exec  ; execute file
                             <2>
                             <2>
                             <2>
                             <2>  %if %0 >= 2
                             <2>  mov ebx, %2
                             <2>  %if %0 >= 3
                             <2>  mov ecx, %3
                             <2>  %if %0 = 4
                             <2>  mov edx, %4
                             <2>  %endif
                             <2>  %endif
                             <2>  %endif
000038EB B80B000000          <2>  mov eax, %1
000038F0 CD30                <2>  int 30h
000038F2 731E                <1>  jnc   short panic
                             <1>  ;
000038F4 BE[586D0000]        <1>  mov   esi, etc_init_err_msg
000038F9 E837000000          <1>  call  print_msg
000038FE EB1C                <1>  jmp   short key_to_reboot
                             <1>
                             <1> ;align 4
                             <1> init_argp:
00003900 [08390000]00000000  <1>  dd    init_file, 0  ; 23/06/2015 (dw -> dd)
                             <1> init_file:
                             <1>  ; 24/08/2015
00003908 2F6574632F696E6974- <1>  db    '/etc/init', 0
00003911 00                  <1>
                             <1> panic:
                             <1>  ; 13/03/2015 (Retro UNIX 386 v1)
                             <1>  ; 07/03/2014 (Retro UNIX 8086 v1)
00003912 BE[3D6D0000]        <1>  mov   esi, panic_msg
00003917 E819000000          <1>  call  print_msg
                             <1> key_to_reboot:
                             <1>  ; 15/11/2015
0000391C E8A72B0000          <1>  call  getch
                             <1>        ; wait for a character from the current tty
                             <1>  ;
00003921 B00A                <1>  mov   al, 0Ah
00003923 8A1D[D6700000]      <1>  mov   bl, [ptty] ; [active_page]
00003929 B407                <1>  mov   ah, 07h ; Black background,
                             <1>              ; light gray forecolor
0000392B E8EDDBFFFF          <1>  call  write_tty
00003930 E96DD8FFFF          <1>  jmp   cpu_reset
                             <1>
                             <1> print_msg:
                             <1>  ; 01/07/2015
                             <1>  ; 13/03/2015 (Retro UNIX 386 v1)
                             <1>  ; 07/03/2014 (Retro UNIX 8086 v1)
                             <1>  ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
                             <1>  ;
                             <1>  ;
00003935 AC                  <1>  lodsb
                             <1> pmsg1:
00003936 56                  <1>  push  esi
00003937 0FB61D[D6700000]    <1>  movzx ebx, byte [ptty]
0000393E B407                <1>  mov   ah, 07h ; Black background, light gray forecolor
00003940 E8D8DBFFFF          <1>  call  write_tty
00003945 5E                  <1>  pop   esi
00003946 AC                  <1>  lodsb
00003947 20C0                <1>  and   al, al
00003949 75EB                <1>  jnz   short pmsg1
0000394B C3                  <1>  retn
                             <1>
                             <1> ctrlbrk:
                             <1>  ; 12/11/2015
                             <1>  ; 13/03/2015 (Retro UNIX 386 v1)
                             <1>  ; 06/12/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; INT 1Bh (control+break) handler
                             <1>  ;
                             <1>        ; Retro Unix 8086 v1 feature only!
                             <1>        ;
0000394C 66833D[BE740000]00  <1>  cmp   word [u.intr], 0
00003954 7645                <1>  jna   short cbrk4
                             <1> cbrk0:
                             <1>  ; 12/11/2015
                             <1>  ; 06/12/2013
00003956 66833D[C0740000]00  <1>  cmp   word [u.quit], 0
0000395E 743B                <1>  jz    short cbrk4
                             <1>  ;
                             <1>  ; 20/09/2013
00003960 6650                <1>  push  ax
```

```
00003962 A0[D6700000]        <1>  mov   al, [ptty]
                             <1>  ;
                             <1>  ; 12/11/2015
                             <1>  ;
                             <1>  ; ctrl+break (EOT, CTRL+D) from serial port
                             <1>  ; or ctrl+break from console (pseudo) tty
                             <1>  ; (!redirection!)
                             <1>  ;
00003967 3C08                <1>  cmp   al, 8 ; serial port tty nums > 7
00003969 7211                <1>         jb      short cbrk1 ; console (pseudo) tty
                             <1>  ;
                             <1>  ; Serial port interrupt handler sets [ptty]
                             <1>  ; to the port's tty number (as temporary).
                             <1>  ;
                             <1>  ; If active process is using a stdin or
                             <1>  ; stdout redirection (by the shell),
                             <1>         ; console tty keyboard must be available
                             <1>  ; to terminate running process,
                             <1>  ; in order to prevent a deadlock.
                             <1>  ;
0000396B 52                  <1>  push  edx
0000396C 0FB615[C9740000]    <1>  movzx edx, byte [u.uno]
00003973 3A82[D5710000]      <1>  cmp     al, [edx+p.ttyc-1] ; console tty (rw)
00003979 5A                  <1>  pop   edx
0000397A 7412                <1>  je    short cbrk2
                             <1> cbrk1:
0000397C FEC0                <1>  inc   al  ; [u.ttyp] : 1 based tty number
                             <1>  ; 06/12/2013
0000397E 3A05[B0740000]      <1>  cmp   al, [u.ttyp] ; recent open tty (r)
00003984 7408                <1>  je    short cbrk2
00003986 3A05[B1740000]      <1>         cmp     al, [u.ttyp+1] ; recent open tty (w)
0000398C 750B                <1>  jne   short cbrk3
                             <1> cbrk2:
                             <1>  ;; 06/12/2013
                             <1>  ;mov  ax, [u.quit]
                             <1>  ;and  ax, ax
                             <1>  ;jz   short cbrk3
                             <1>  ;
0000398E 6631C0              <1>  xor   ax, ax ; 0
00003991 6648                <1>  dec   ax
                             <1>  ; 0FFFFh = 'ctrl+brk' keystroke
00003993 66A3[C0740000]      <1>  mov   [u.quit], ax
                             <1> cbrk3:
00003999 6658                <1>  pop   ax
                             <1> cbrk4:
0000399B C3                  <1>  retn
                             <1>
                             <1> com2_int:
                             <1>  ; 07/11/2015
                             <1>  ; 24/10/2015
                             <1>  ; 23/10/2015
                             <1>  ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 28/07/2014 (Retro UNIX 8086 v1)
                             <1>  ; < serial port 2 interrupt handler >
                             <1>  ;
0000399C 890424              <1>  mov   [esp], eax ; overwrite call return address
                             <1>  ;push eax
0000399F 66B80900            <1>  mov   ax, 9
000039A3 EB07                <1>  jmp   short comm_int
                             <1> com1_int:
                             <1>  ; 07/11/2015
                             <1>  ; 24/10/2015
000039A5 890424              <1>  mov   [esp], eax ; overwrite call return address
                             <1>  ; 23/10/2015
                             <1>  ;push eax
000039A8 66B80800            <1>  mov   ax, 8
                             <1> comm_int:
                             <1>  ; 20/11/2015
                             <1>  ; 18/11/2015
                             <1>  ; 17/11/2015
                             <1>  ; 16/11/2015
                             <1>  ; 09/11/2015
                             <1>  ; 08/11/2015
                             <1>  ; 07/11/2015
                             <1>  ; 06/11/2015 (serial4.asm, 'serial')
                             <1>  ; 01/11/2015
                             <1>  ; 26/10/2015
                             <1>  ; 23/10/2015
000039AC 53                  <1>  push  ebx
000039AD 56                  <1>  push  esi
000039AE 57                  <1>  push  edi
000039AF 1E                  <1>  push  ds
000039B0 06                  <1>  push  es
                             <1>  ; 18/11/2015
000039B1 0F20DB              <1>  mov   ebx, cr3
000039B4 53                  <1>  push  ebx ; ****
                             <1>  ;
000039B5 51                  <1>  push  ecx ; ***
000039B6 52                  <1>  push  edx ; **
                             <1>  ;
000039B7 BB10000000          <1>  mov   ebx, KDATA
000039BC 8EDB                <1>  mov   ds, bx
```

```
000039BE 8EC3                  <1>  mov   es, bx
                               <1>  ;
000039C0 8B0D[A8700000]        <1>  mov   ecx, [k_page_dir]
000039C6 0F22D9                <1>  mov   cr3, ecx
                               <1>  ; 20/11/2015
                               <1>  ; Interrupt identification register
000039C9 66BAFA02              <1>  mov   dx, 2FAh ; COM2
                               <1>  ;
000039CD 3C08                  <1>  cmp   al, 8
000039CF 7702                  <1>  ja    short com_i0
                               <1>  ;
                               <1>  ; 20/11/2015
                               <1>  ; 17/11/2015
                               <1>  ; 16/11/2015
                               <1>  ; 15/11/2015
                               <1>  ; 24/10/2015
                               <1>  ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)
                               <1>  ; 28/07/2014 (Retro UNIX 8086 v1)
                               <1>  ; < serial port 1 interrupt handler >
                               <1>  ;
000039D1 FEC6                  <1>  inc   dh ; 3FAh ; COM1 Interrupt id. register
                               <1> com_i0:
                               <1>  ;push eax ; *
                               <1>  ; 07/11/2015
000039D3 A2[16710000]          <1>  mov   byte [ccomport], al
                               <1>  ; 09/11/2015
000039D8 0FB7D8                <1>  movzx ebx, ax ; 8 or 9
                               <1>  ; 17/11/2015
                               <1>  ; reset request for response status
000039DB 88A3[0C710000]        <1>  mov   [ebx+req_resp-8], ah ; 0
                               <1>  ;
                               <1>  ; 20/11/2015
000039E1 EC                    <1>  in    al, dx            ; read interrupt id. register
000039E2 EB00                  <1>  JMP   $+2         ; I/O DELAY
000039E4 2404                  <1>  and   al, 4       ; received data available?
000039E6 7470                  <1>  jz    short com_eoi     ; (transmit. holding reg. empty)
                               <1>  ;
                               <1>  ; 20/11/2015
000039E8 80EA02                <1>  sub   dl, 3FAh-3F8h     ; data register (3F8h, 2F8h)
000039EB EC                    <1>  in    al, dx      ; read character
                               <1>  ;JMP  $+2         ; I/O DELAY
                               <1>  ; 08/11/2015
                               <1>  ; 07/11/2015
000039EC 89DE                  <1>  mov   esi, ebx
000039EE 89DF                  <1>  mov   edi, ebx
000039F0 81C6[10710000]        <1>  add   esi, rchar - 8 ; points to last received char
000039F6 81C7[12710000]        <1>  add   edi, schar - 8 ; points to last sent char
000039FC 8806                  <1>  mov   [esi], al ; received char (current char)
                               <1>  ; query
000039FE 20C0                  <1>  and   al, al
00003A00 7527                  <1>  jnz   short com_i2
                               <1>        ; response
                               <1>  ; 17/11/2015
                               <1>  ; set request for response status
00003A02 FE83[0C710000]        <1>         inc     byte [ebx+req_resp-8] ; 1
                               <1>  ;
00003A08 6683C205              <1>  add   dx, 3FDh-3F8h     ; (3FDh, 2FDh)
00003A0C EC                    <1>  in    al, dx            ; read line status register
00003A0D EB00                  <1>  JMP   $+2         ; I/O DELAY
00003A0F 2420                  <1>  and   al, 20h           ; transmitter holding reg. empty?
00003A11 7445                  <1>  jz    short com_eoi     ; no
00003A13 B0FF                  <1>  mov   al, 0FFh    ; response
00003A15 6683EA05              <1>  sub   dx, 3FDh-3F8h     ; data port (3F8h, 2F8h)
00003A19 EE                    <1>  out   dx, al            ; send on serial port
                               <1>  ; 17/11/2015
00003A1A 803F00                <1>  cmp   byte [edi], 0   ; query ? (schar)
00003A1D 7502                  <1>  jne   short com_i1    ; no
00003A1F 8807                  <1>  mov   [edi], al   ; 0FFh (responded)
                               <1> com_i1:
                               <1>  ; 17/11/2015
                               <1>  ; reset request for response status (again)
00003A21 FE8B[0C710000]        <1>         dec     byte [ebx+req_resp-8] ; 0
00003A27 EB2F                  <1>  jmp   short com_eoi
                               <1> com_i2:
                               <1>  ; 08/11/2015
00003A29 3CFF                  <1>  cmp   al, 0FFh    ; (response ?)
00003A2B 7417                  <1>  je    short com_i3      ; (check for response signal)
                               <1>  ; 07/11/2015
00003A2D 3C04                  <1>  cmp   al, 04h     ; EOT
00003A2F 751C                  <1>  jne   short com_i4
                               <1>  ; EOT = 04h (End of Transmit) - 'CTRL + D'
                               <1>  ;(an EOT char is supposed as a ctrl+brk from the terminal)
                               <1>  ; 08/11/2015
                               <1>        ; ptty -> tty 0 to 7 (pseudo screens)
00003A31 861D[D6700000]        <1>  xchg  bl, [ptty]  ; tty number (8 or 9)
00003A37 E810FFFFFF            <1>  call  ctrlbrk
00003A3C 861D[D6700000]        <1>  xchg  [ptty], bl ; (restore ptty value and BL value)
                               <1>  ;mov  al, 04h ; EOT
                               <1>  ; 08/11/2015
00003A42 EB09                  <1>  jmp   short com_i4
                               <1> com_i3:
                               <1>  ; 08/11/2015
```

```
                              <1>  ; If 0FFh has been received just after a query
                              <1>  ; (schar, ZERO), it is a response signal.
                              <1>  ; 17/11/2015
00003A44 803F00               <1>         cmp     byte [edi], 0 ; query ? (schar)
00003A47 7704                 <1>  ja    short com_i4 ; no
                              <1>  ; reset query status (schar)
00003A49 8807                 <1>  mov   [edi], al ; 0FFh
00003A4B FEC0                 <1>  inc   al ; 0
                              <1> com_i4:
                              <1>  ; 27/07/2014
                              <1>  ; 09/07/2014
00003A4D D0E3                 <1>  shl   bl, 1
00003A4F 81C3[D8700000]       <1>  add   ebx, ttychr
                              <1>  ; 23/07/2014 (always overwrite)
                              <1>  ;;cmp word [ebx], 0
                              <1>  ;;ja  short com_eoi
                              <1>  ;
00003A55 668903               <1>  mov   [ebx], ax   ; Save ascii code
                              <1>                   ; scan code = 0
                              <1> com_eoi:
                              <1>  ;mov  al, 20h
                              <1>  ;out  20h, al        ; end of interrupt
                              <1>  ;
                              <1>  ; 07/11/2015
                              <1>        ;pop  eax ; *
00003A58 A0[16710000]         <1>  mov   al, byte [ccomport] ; current COM port
                              <1>   ; al = tty number (8 or 9)
00003A5D E88B1A0000           <1>         call wakeup
                              <1> com_iret:
                              <1>  ; 23/10/2015
00003A62 5A                   <1>  pop   edx ; **
00003A63 59                   <1>  pop   ecx ; ***
                              <1>  ; 18/11/2015
                              <1>  ;pop  eax ; ****
                              <1>  ;mov  cr3, eax
                              <1>  ;jmp  iiret
00003A64 E94BD0FFFF           <1>  jmp   iiretp
                              <1>
                              <1> ;iiretp: ; 01/09/2015
                              <1> ;; 28/08/2015
                              <1> ;pop   eax ; (*) page directory
                              <1> ;mov   cr3, eax
                              <1> ;iiret:
                              <1> ;; 22/08/2014
                              <1> ;mov   al, 20h ; END OF INTERRUPT COMMAND TO 8259
                              <1> ;out   20h, al     ; 8259 PORT
                              <1> ;;
                              <1> ;pop   es
                              <1> ;pop   ds
                              <1> ;pop   edi
                              <1> ;pop   esi
                              <1> ;pop   ebx ; 29/08/2014
                              <1> ;pop   eax
                              <1> ;iretd
                              <1>
                              <1> sp_init:
                              <1>  ; 07/11/2015
                              <1>  ; 29/10/2015
                              <1>  ; 26/10/2015
                              <1>  ; 23/10/2015
                              <1>  ; 29/06/2015
                              <1>  ; 14/03/2015 (Retro UNIX 386 v1 - 115200 baud)
                              <1>  ; 28/07/2014 (Retro UNIX 8086 v1 - 9600 baud)
                              <1>  ; Initialization of Serial Port Communication Parameters
                              <1>  ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
                              <1>  ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
                              <1>  ;
                              <1>  ; ((Modified registers: EAX, ECX, EDX, EBX))
                              <1>  ;
                              <1>  ; INPUT:  (29/06/2015)
                              <1>  ;     AL = 0 for COM1
                              <1>  ;          1 for COM2
                              <1>  ;     AH = Communication parameters
                              <1>  ;
                              <1>  ;  (*) Communication parameters (except BAUD RATE):
                              <1>  ;     Bit   4     3     2     1     0
                              <1>  ;           -PARITY--   STOP BIT  -WORD LENGTH-
                              <1>  ;  this one -->  00 = none    0 = 1 bit  11 = 8 bits
                              <1>  ;           01 = odd     1 = 2 bits 10 = 7 bits
                              <1>  ;           11 = even
                              <1>  ;  Baud rate setting bits: (29/06/2015)
                              <1>  ;           Retro UNIX 386 v1 feature only !
                              <1>  ;     Bit   7    6    5  | Baud rate
                              <1>  ;           ------------------------
                              <1>  ;     value 0    0    0  | Default (Divisor = 1)
                              <1>  ;           0    0    1  | 9600 (12)
                              <1>  ;           0    1    0  | 19200 (6)
                              <1>  ;           0    1        1  | 38400 (3)
                              <1>  ;           1    0        0  | 14400 (8)
                              <1>  ;           1    0        1  | 28800 (4)
                              <1>  ;           1    1    0  | 57600 (2)
                              <1>  ;           1    1    1  | 115200 (1)
```

```
                            <1>
                            <1>  ; References:
                            <1>  ; (1) IBM PC-XT Model 286 BIOS Source Code
                            <1>  ;     RS232.ASM --- 10/06/1985 COMMUNICATIONS BIOS (RS232)
                            <1>  ; (2) Award BIOS 1999 - ATORGS.ASM
                            <1>  ; (3) http://wiki.osdev.org/Serial_Ports
                            <1>  ;
                            <1>  ; Set communication parameters for COM1 (= 03h)
                            <1>  ;
00003A69 BB[12710000]       <1>  mov   ebx, com1p        ; COM1 parameters
00003A6E 66BAF803           <1>  mov   dx, 3F8h          ; COM1
                            <1>   ; 29/10/2015
00003A72 66B90103           <1>  mov   cx, 301h  ; divisor = 1 (115200 baud)
00003A76 E86F000000         <1>  call  sp_i3 ; call A4
00003A7B A880               <1>  test  al, 80h
00003A7D 7410               <1>  jz    short sp_i0 ; OK..
                            <1>        ; Error !
                            <1>  ;mov  dx, 3F8h
00003A7F 80EA05             <1>  sub   dl, 5 ; 3FDh -> 3F8h
00003A82 66B90E03           <1>  mov   cx, 30Eh  ; divisor = 12 (9600 baud)
00003A86 E85F000000         <1>  call  sp_i3 ; call A4
00003A8B A880               <1>  test  al, 80h
00003A8D 7508               <1>  jnz   short sp_i1
                            <1> sp_i0:
                            <1>         ; (Note: Serial port interrupts will be disabled here...)
                            <1>         ; (INT 14h initialization code disables interrupts.)
                            <1>  ;
00003A8F C603E3             <1>  mov   byte [ebx], 0E3h ; 11100011b
00003A92 E8DC000000         <1>  call  sp_i5 ; 29/06/2015
                            <1> sp_i1:
00003A97 43                 <1>  inc   ebx
00003A98 66BAF802           <1>  mov   dx, 2F8h          ; COM2
                            <1>   ; 29/10/2015
00003A9C 66B90103           <1>  mov   cx, 301h  ; divisor = 1 (115200 baud)
00003AA0 E845000000         <1>  call  sp_i3 ; call A4
00003AA5 A880               <1>  test  al, 80h
00003AA7 7410               <1>  jz    short sp_i2 ; OK..
                            <1>        ; Error !
                            <1>  ;mov  dx, 2F8h
00003AA9 80EA05             <1>  sub   dl, 5 ; 2FDh -> 2F8h
00003AAC 66B90E03           <1>  mov   cx, 30Eh  ; divisor = 12 (9600 baud)
00003AB0 E835000000         <1>  call  sp_i3 ; call A4
00003AB5 A880               <1>  test  al, 80h
00003AB7 7530               <1>  jnz   short sp_i7
                            <1> sp_i2:
00003AB9 C603E3             <1>  mov   byte [ebx], 0E3h ; 11100011b
                            <1> sp_i6:
                            <1>  ;; COM2 - enabling IRQ 3
                            <1>  ; 07/11/2015
                            <1>  ; 26/10/2015
00003ABC 9C                 <1>  pushf
00003ABD FA                 <1>  cli
                            <1>  ;
00003ABE 66BAFC02           <1>  mov   dx, 2FCh          ; modem control register
00003AC2 EC                 <1>  in    al, dx                  ; read register
00003AC3 EB00               <1>  JMP   $+2               ; I/O DELAY
00003AC5 0C08               <1>  or    al, 8             ; enable bit 3 (OUT2)
00003AC7 EE                 <1>  out   dx, al            ; write back to register
00003AC8 EB00               <1>  JMP   $+2               ; I/O DELAY
00003ACA 66BAF902           <1>  mov   dx, 2F9h          ; interrupt enable register
00003ACE EC                 <1>  in    al, dx            ; read register
00003ACF EB00               <1>  JMP   $+2               ; I/O DELAY
                            <1>  ;or   al, 1             ; receiver data interrupt enable and
00003AD1 0C03               <1>  or    al, 3             ; transmitter empty interrupt enable
00003AD3 EE                 <1>  out   dx, al                  ; write back to register
00003AD4 EB00               <1>  JMP   $+2               ; I/O DELAY
00003AD6 E421               <1>  in    al, 21h           ; read interrupt mask register
00003AD8 EB00               <1>  JMP   $+2               ; I/O DELAY
00003ADA 24F7               <1>  and   al, 0F7h          ; enable IRQ 3 (COM2)
00003ADC E621               <1>  out   21h, al           ; write back to register
                            <1>  ;
                            <1>  ; 23/10/2015
00003ADE B8[9C390000]       <1>  mov   eax, com2_int
00003AE3 A3[7B3F0000]       <1>  mov   [com2_irq3], eax
                            <1>  ; 26/10/2015
00003AE8 9D                 <1>  popf
                            <1> sp_i7:
00003AE9 C3                 <1>  retn
                            <1>
                            <1> sp_i3:
                            <1> ;A4:   ;-----      INITIALIZE THE COMMUNICATIONS PORT
                            <1>  ; 28/10/2015
00003AEA FEC2               <1>  inc   dl    ; 3F9h (2F9h)     ; 3F9h, COM1 Interrupt enable register
00003AEC B000               <1>  mov   al, 0
00003AEE EE                 <1>  out   dx, al                  ; disable serial port interrupt
00003AEF EB00               <1>  JMP   $+2               ; I/O DELAY
00003AF1 80C202             <1>  add   dl, 2       ; 3FBh (2FBh)     ; COM1 Line control register (3FBh)
00003AF4 B080               <1>  mov   al, 80h
00003AF6 EE                 <1>  out   dx, al                  ; SET DLAB=1 ; divisor latch access bit
                            <1>  ;-----      SET BAUD RATE DIVISOR
                            <1>  ; 26/10/2015
00003AF7 80EA03             <1>  sub   dl, 3   ; 3F8h (2F8h)   ; register for least significant byte
```

```
                              <1>                          ; of the divisor value
00003AFA 88C8                 <1>  mov   al, cl      ; 1
00003AFC EE                   <1>  out   dx, al                  ; 1 = 115200 baud (Retro UNIX 386 v1)
                              <1>                          ; 2 = 57600 baud
                              <1>                          ; 3 = 38400 baud
                              <1>                          ; 6 = 19200 baud
                              <1>                          ; 12 = 9600 baud (Retro UNIX 8086 v1)
00003AFD EB00                 <1>  JMP   $+2               ; I/O DELAY
00003AFF 28C0                 <1>  sub   al, al
00003B01 FEC2                 <1>  inc   dl      ; 3F9h (2F9h)   ; register for most significant byte
                              <1>                          ; of the divisor value
00003B03 EE                   <1>  out   dx, al ; 0
00003B04 EB00                 <1>  JMP   $+2               ; I/O DELAY
                              <1>  ;
00003B06 88E8                 <1>  mov   al, ch ; 3        ; 8 data bits, 1 stop bit, no parity
                              <1>  ;and  al, 1Fh ; Bits 0,1,2,3,4
00003B08 80C202               <1>  add   dl, 2 ; 3FBh (2FBh)     ; Line control register
00003B0B EE                   <1>  out   dx, al
00003B0C EB00                 <1>  JMP   $+2               ; I/O DELAY
                              <1>  ; 29/10/2015
00003B0E FECA                 <1>  dec   dl    ; 3FAh (2FAh)     ; FIFO Control register (16550/16750)
00003B10 30C0                 <1>  xor   al, al                  ; 0
00003B12 EE                   <1>  out   dx, al                  ; Disable FIFOs (reset to 8250 mode)
00003B13 EB00                 <1>  JMP   $+2
                              <1> sp_i4:
                              <1> ;A18:  ;-----      COMM PORT STATUS ROUTINE
                              <1>  ; 29/06/2015 (line status after modem status)
00003B15 80C204               <1>  add   dl, 4 ; 3FEh (2FEh)     ; Modem status register
                              <1> sp_i4s:
00003B18 EC                   <1>  in    al, dx                  ; GET MODEM CONTROL STATUS
00003B19 EB00                 <1>  JMP   $+2               ; I/O DELAY
00003B1B 88C4                 <1>  mov   ah, al                  ; PUT IN (AH) FOR RETURN
00003B1D FECA                 <1>  dec   dl    ; 3FDh (2FDh)     ; POINT TO LINE STATUS REGISTER
                              <1>                          ; dx = 3FDh for COM1, 2FDh for COM2
00003B1F EC                   <1>  in    al, dx                  ; GET LINE CONTROL STATUS
                              <1>  ; AL = Line status, AH = Modem status
00003B20 C3                   <1>  retn
                              <1>
                              <1> sp_status:
                              <1>  ; 29/06/2015
                              <1>  ; 27/06/2015 (Retro UNIX 386 v1)
                              <1>  ; Get serial port status
00003B21 66BAFE03             <1>  mov   dx, 3FEh          ; Modem status register (COM1)
00003B25 28C6                 <1>  sub   dh, al                  ; dh = 2 for COM2 (al = 1)
                              <1>                          ; dx = 2FEh for COM2
00003B27 EBEF                 <1>  jmp   short sp_i4s
                              <1>
                              <1> sp_setp: ; Set serial port communication parameters
                              <1>  ; 07/11/2015
                              <1>  ; 29/10/2015
                              <1>  ; 29/06/2015
                              <1>  ; Retro UNIX 386 v1 feature only !
                              <1>  ;
                              <1>  ; INPUT:
                              <1>  ;     AL = 0 for COM1
                              <1>  ;          1 for COM2
                              <1>  ;     AH = Communication parameters (*)
                              <1>  ; OUTPUT:
                              <1>  ;     CL = Line status
                              <1>  ;     CH = Modem status
                              <1>  ;   If cf = 1 -> Error code in [u.error]
                              <1>  ;           'invalid parameter !'
                              <1>  ;                 or
                              <1>  ;           'device not ready !' error
                              <1>  ;
                              <1>  ;  (*) Communication parameters (except BAUD RATE):
                              <1>  ;     Bit   4     3     2     1     0
                              <1>  ;           -PARITY--   STOP BIT  -WORD LENGTH-
                              <1>  ;  this one -->  00 = none    0 = 1 bit  11 = 8 bits
                              <1>  ;           01 = odd     1 = 2 bits 10 = 7 bits
                              <1>  ;           11 = even
                              <1>  ;  Baud rate setting bits: (29/06/2015)
                              <1>  ;           Retro UNIX 386 v1 feature only !
                              <1>  ;     Bit   7    6    5  | Baud rate
                              <1>  ;           ------------------------
                              <1>  ;     value 0    0    0  | Default (Divisor = 1)
                              <1>  ;           0    0    1  | 9600 (12)
                              <1>  ;           0    1    0  | 19200 (6)
                              <1>  ;           0    1       1  | 38400 (3)
                              <1>  ;           1    0       0  | 14400 (8)
                              <1>  ;           1    0       1  | 28800 (4)
                              <1>  ;           1    1    0  | 57600 (2)
                              <1>  ;           1    1    1  | 115200 (1)
                              <1>  ;
                              <1>  ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
                              <1>  ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
                              <1>  ;
                              <1>  ; ((Modified registers: EAX, ECX, EDX, EBX))
                              <1>  ;
00003B29 66BAF803             <1>  mov   dx, 3F8h
00003B2D BB[12710000]         <1>  mov   ebx, com1p ; COM1 control byte offset
00003B32 3C01                 <1>  cmp   al, 1
```

```
00003B34 776B                <1>  ja    short sp_invp_err
00003B36 7203                <1>  jb    short sp_setp1 ;  COM1 (AL = 0)
00003B38 FECE                <1>  dec   dh ; 2F8h
00003B3A 43                  <1>  inc   ebx ; COM2 control byte offset
                             <1> sp_setp1:
                             <1>  ; 29/10/2015
00003B3B 8823                <1>  mov   [ebx], ah
00003B3D 0FB6CC              <1>  movzx       ecx, ah
00003B40 C0E905              <1>  shr   cl, 5 ; -> baud rate index
00003B43 80E41F              <1>  and   ah, 1Fh ; communication parameters except baud rate
00003B46 8A81[B03B0000]      <1>  mov   al, [ecx+b_div_tbl]
00003B4C 6689C1              <1>  mov   cx, ax
00003B4F E896FFFFFF          <1>  call  sp_i3
00003B54 6689C1              <1>  mov   cx, ax ; CL = Line status, CH = Modem status
00003B57 A880                <1>  test  al, 80h
00003B59 740F                <1>  jz    short sp_setp2
00003B5B C603E3              <1>         mov     byte [ebx], 0E3h ; Reset to initial value (11100011b)
                             <1> stp_dnr_err:
00003B5E C705[CF740000]0F00- <1>  mov   dword [u.error], ERR_DEV_NOT_RDY ; 'device not ready !'
00003B66 0000                <1>
                             <1>  ; CL = Line status, CH = Modem status
00003B68 F9                  <1>  stc
00003B69 C3                  <1>  retn
                             <1> sp_setp2:
00003B6A 80FE02              <1>  cmp   dh, 2 ; COM2 (2F?h)
00003B6D 0F8649FFFFFF        <1>         jna     sp_i6
                             <1>                 ; COM1 (3F?h)
                             <1> sp_i5:
                             <1>  ; 07/11/2015
                             <1>  ; 26/10/2015
                             <1>  ; 29/06/2015
                             <1>  ;
                             <1>  ;; COM1 - enabling IRQ 4
00003B73 9C                  <1>  pushf
00003B74 FA                  <1>  cli
00003B75 66BAFC03            <1>  mov   dx, 3FCh          ; modem control register
00003B79 EC                  <1>  in    al, dx                  ; read register
00003B7A EB00                <1>  JMP   $+2               ; I/O DELAY
00003B7C 0C08                <1>  or    al, 8             ; enable bit 3 (OUT2)
00003B7E EE                  <1>  out   dx, al            ; write back to register
00003B7F EB00                <1>  JMP   $+2               ; I/O DELAY
00003B81 66BAF903            <1>  mov   dx, 3F9h          ; interrupt enable register
00003B85 EC                  <1>  in    al, dx            ; read register
00003B86 EB00                <1>  JMP   $+2               ; I/O DELAY
                             <1>  ;or   al, 1             ; receiver data interrupt enable and
00003B88 0C03                <1>  or    al, 3             ; transmitter empty interrupt enable
00003B8A EE                  <1>  out   dx, al                  ; write back to register
00003B8B EB00                <1>  JMP   $+2               ; I/O DELAY
00003B8D E421                <1>  in    al, 21h           ; read interrupt mask register
00003B8F EB00                <1>  JMP   $+2               ; I/O DELAY
00003B91 24EF                <1>  and   al, 0EFh          ; enable IRQ 4 (COM1)
00003B93 E621                <1>  out   21h, al           ; write back to register
                             <1>  ;
                             <1>  ; 23/10/2015
00003B95 B8[A5390000]        <1>  mov   eax, com1_int
00003B9A A3[773F0000]        <1>  mov   [com1_irq4], eax
                             <1>  ; 26/10/2015
00003B9F 9D                  <1>  popf
00003BA0 C3                  <1>  retn
                             <1>
                             <1> sp_invp_err:
00003BA1 C705[CF740000]1700- <1>  mov   dword [u.error], ERR_INV_PARAMETER ; 'invalid parameter !'
00003BA9 0000                <1>
00003BAB 31C9                <1>  xor   ecx, ecx
00003BAD 49                  <1>  dec   ecx ; 0FFFFh
00003BAE F9                  <1>  stc
00003BAF C3                  <1>  retn
                             <1>
                             <1> ; 29/10/2015
                             <1> b_div_tbl: ; Baud rate divisor table (115200/divisor)
00003BB0 010C0603080401      <1>  db 1, 12, 6, 3, 8, 4, 1
                             <1>
                             <1> ; Retro UNIX 8086 v1 - UNIX.ASM (01/09/2014)
                             <1> epoch:
                             <1>  ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
                             <1>  ; 09/04/2013 (Retro UNIX 8086 v1 - UNIX.ASM)
                             <1>  ; 'epoch' procedure prototype:
                             <1>  ;                 UNIXCOPY.ASM, 10/03/2013
                             <1>  ; 14/11/2012
                             <1>  ; unixboot.asm (boot file configuration)
                             <1>  ; version of "epoch" procedure in "unixproc.asm"
                             <1>  ; 21/7/2012
                             <1>  ; 15/7/2012
                             <1>  ; 14/7/2012
                             <1>  ; Erdogan Tan - RETRO UNIX v0.1
                             <1>  ; compute current date and time as UNIX Epoch/Time
                             <1>  ; UNIX Epoch: seconds since 1/1/1970 00:00:00
                             <1>  ;
                             <1>         ;  ((Modified registers: EAX, EDX, ECX, EBX))
                             <1>  ;
00003BB7 E81D010000          <1>  call  get_rtc_time            ; Return Current Time
00003BBC 86E9                <1>         xchg ch,cl
```

```
00003BBE 66890D[B26D0000]    <1>         mov  [hour], cx
00003BC5 86F2                <1>         xchg dh,dl
00003BC7 668915[B66D0000]    <1>         mov  [second], dx
                             <1>  ;
00003BCE E837010000          <1>         call get_rtc_date            ; Return Current Date
00003BD3 86E9                <1>         xchg ch,cl
00003BD5 66890D[AC6D0000]    <1>         mov  [year], cx
00003BDC 86F2                <1>         xchg dh,dl
00003BDE 668915[AE6D0000]    <1>         mov  [month], dx
                             <1>  ;
00003BE5 66B93030            <1>  mov   cx, 3030h
                             <1>  ;
00003BE9 A0[B26D0000]        <1>  mov   al, [hour] ; Hour
                             <1>              ; AL <= BCD number)
00003BEE D410                <1>         db   0D4h,10h          ; Undocumented inst. AAM
                             <1>                          ; AH = AL / 10h
                             <1>                          ; AL = AL MOD 10h
00003BF0 D50A                <1>         aad  ; AX= AH*10+AL
00003BF2 A2[B26D0000]        <1>  mov   [hour], al
00003BF7 A0[B36D0000]        <1>  mov   al, [hour+1] ; Minute
                             <1>              ; AL <= BCD number)
00003BFC D410                <1>         db   0D4h,10h          ; Undocumented inst. AAM
                             <1>                          ; AH = AL / 10h
                             <1>                          ; AL = AL MOD 10h
00003BFE D50A                <1>         aad  ; AX= AH*10+AL
00003C00 A2[B46D0000]        <1>  mov   [minute], al
00003C05 A0[B66D0000]        <1>  mov   al, [second] ; Second
                             <1>              ; AL <= BCD number)
00003C0A D410                <1>         db   0D4h,10h          ; Undocumented inst. AAM
                             <1>                          ; AH = AL / 10h
                             <1>                          ; AL = AL MOD 10h
00003C0C D50A                <1>         aad  ; AX= AH*10+AL
00003C0E A2[B66D0000]        <1>  mov   [second], al
00003C13 66A1[AC6D0000]      <1>  mov   ax, [year] ; Year (century)
00003C19 6650                <1>         push ax
                             <1>        ; AL <= BCD number)
00003C1B D410                <1>         db   0D4h,10h          ; Undocumented inst. AAM
                             <1>                          ; AH = AL / 10h
                             <1>                          ; AL = AL MOD 10h
00003C1D D50A                <1>         aad  ; AX= AH*10+AL
00003C1F B464                <1>  mov   ah, 100
00003C21 F6E4                <1>  mul   ah
00003C23 66A3[AC6D0000]      <1>  mov   [year], ax
00003C29 6658                <1>  pop   ax
00003C2B 88E0                <1>  mov   al, ah
                             <1>              ; AL <= BCD number)
00003C2D D410                <1>         db   0D4h,10h          ; Undocumented inst. AAM
                             <1>                          ; AH = AL / 10h
                             <1>                          ; AL = AL MOD 10h
00003C2F D50A                <1>         aad  ; AX= AH*10+AL
00003C31 660105[AC6D0000]    <1>  add   [year], ax
00003C38 A0[AE6D0000]        <1>  mov   al, [month] ; Month
                             <1>              ; AL <= BCD number)
00003C3D D410                <1>         db   0D4h,10h          ; Undocumented inst. AAM
                             <1>                          ; AH = AL / 10h
                             <1>                          ; AL = AL MOD 10h
00003C3F D50A                <1>         aad  ; AX= AH*10+AL
00003C41 A2[AE6D0000]        <1>  mov   [month], al
00003C46 A0[AF6D0000]        <1>         mov     al, [month+1]        ; Day
                             <1>              ; AL <= BCD number)
00003C4B D410                <1>         db   0D4h,10h          ; Undocumented inst. AAM
                             <1>                          ; AH = AL / 10h
                             <1>                          ; AL = AL MOD 10h
00003C4D D50A                <1>         aad  ; AX= AH*10+AL
00003C4F A2[B06D0000]        <1>         mov     [day], al
                             <1>
                             <1> convert_to_epoch:
                             <1>  ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit modification)
                             <1>  ; 09/04/2013 (retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; ((Modified registers: EAX, EDX, EBX))
                             <1>  ;
                             <1>  ; Derived from DALLAS Semiconductor
                             <1>  ; Application Note 31 (DS1602/DS1603)
                             <1>  ; 6 May 1998
00003C54 29C0                <1>  sub   eax, eax
00003C56 66A1[AC6D0000]      <1>  mov   ax, [year]
00003C5C 662DB207            <1>  sub   ax, 1970
00003C60 BA6D010000          <1>  mov   edx, 365
00003C65 F7E2                <1>  mul   edx
00003C67 31DB                <1>  xor   ebx, ebx
00003C69 8A1D[AE6D0000]      <1>  mov   bl, [month]
00003C6F FECB                <1>  dec   bl
00003C71 D0E3                <1>  shl   bl, 1
                             <1>  ;sub  edx, edx
00003C73 668B93[B86D0000]    <1>  mov   dx, [EBX+DMonth]
00003C7A 8A1D[B06D0000]      <1>         mov     bl, [day]
00003C80 FECB                <1>  dec   bl
00003C82 01D0                <1>  add   eax, edx
00003C84 01D8                <1>  add   eax, ebx
                             <1>              ; EAX = days since 1/1/1970
00003C86 668B15[AC6D0000]    <1>  mov   dx, [year]
```

```
00003C8D 6681EAB107          <1>  sub   dx, 1969
00003C92 66D1EA              <1>  shr   dx, 1
00003C95 66D1EA              <1>  shr   dx, 1
                             <1>        ; (year-1969)/4
00003C98 01D0                <1>  add   eax, edx
                             <1>              ; + leap days since 1/1/1970
00003C9A 803D[AE6D0000]02    <1>  cmp   byte [month], 2   ; if past february
00003CA1 7610                <1>  jna   short cte1
00003CA3 668B15[AC6D0000]    <1>  mov   dx, [year]
00003CAA 6683E203            <1>  and   dx, 3 ; year mod 4
00003CAE 7503                <1>  jnz   short cte1
                             <1>              ; and if leap year
00003CB0 83C001              <1>  add   eax, 1      ; add this year's leap day (february 29)
                             <1> cte1:              ; compute seconds since 1/1/1970
00003CB3 BA18000000          <1>  mov   edx, 24
00003CB8 F7E2                <1>  mul   edx
00003CBA 8A15[B26D0000]      <1>  mov   dl, [hour]
00003CC0 01D0                <1>  add   eax, edx
                             <1>        ; EAX = hours since 1/1/1970 00:00:00
                             <1>  ;mov  ebx, 60
00003CC2 B33C                <1>  mov   bl, 60
00003CC4 F7E3                <1>  mul   ebx
00003CC6 8A15[B46D0000]      <1>  mov   dl, [minute]
00003CCC 01D0                <1>  add   eax, edx
                             <1>        ; EAX = minutes since 1/1/1970 00:00:00
                             <1>  ;mov  ebx, 60
00003CCE F7E3                <1>  mul   ebx
00003CD0 8A15[B66D0000]      <1>  mov   dl, [second]
00003CD6 01D0                <1>  add   eax, edx
                             <1>        ; EAX -> seconds since 1/1/1970 00:00:00
00003CD8 C3                  <1>  retn
                             <1>
                             <1> get_rtc_time:
                             <1>  ; 15/03/2015
                             <1>  ; Derived from IBM PC-XT Model 286 BIOS Source Code
                             <1>  ; BIOS2.ASM ---- 10/06/1985 BIOS INTERRUPT ROUTINES
                             <1>  ; INT 1Ah
                             <1>  ; (AH) = 02H  READ THE REAL TIME CLOCK AND RETURN WITH,    :
                             <1>  ;       (CH) = HOURS IN BCD (00-23)            :
                             <1>  ;       (CL) = MINUTES IN BCD (00-59)                :
                             <1>  ;       (DH) = SECONDS IN BCD (00-59)                :
                             <1>  ;       (DL) = DAYLIGHT SAVINGS ENABLE (00-01).      :
                             <1>  ;
                             <1> RTC_20:                        ; GET RTC TIME
00003CD9 FA                  <1>  cli
00003CDA E870CFFFFF          <1>  CALL  UPD_IPR           ; CHECK FOR UPDATE IN PROCESS
00003CDF 7227                <1>  JC    short RTC_29            ; EXIT IF ERROR (CY= 1)
                             <1>
00003CE1 B000                <1>  MOV   AL,CMOS_SECONDS   ; SET ADDRESS OF SECONDS
00003CE3 E84FCFFFFF          <1>  CALL  CMOS_READ         ; GET SECONDS
00003CE8 88C6                <1>  MOV   DH,AL             ; SAVE
00003CEA B00B                <1>  MOV   AL,CMOS_REG_B           ; ADDRESS ALARM REGISTER
00003CEC E846CFFFFF          <1>  CALL  CMOS_READ         ; READ CURRENT VALUE OF DSE BIT
00003CF1 2401                <1>  AND   AL,00000001B            ; MASK FOR VALID DSE BIT
00003CF3 88C2                <1>  MOV   DL,AL             ; SET [DL] TO ZERO FOR NO DSE BIT
00003CF5 B002                <1>  MOV   AL,CMOS_MINUTES   ; SET ADDRESS OF MINUTES
00003CF7 E83BCFFFFF          <1>  CALL  CMOS_READ         ; GET MINUTES
00003CFC 88C1                <1>  MOV   CL,AL             ; SAVE
00003CFE B004                <1>  MOV   AL,CMOS_HOURS           ; SET ADDRESS OF HOURS
00003D00 E832CFFFFF          <1>  CALL  CMOS_READ         ; GET HOURS
00003D05 88C5                <1>  MOV   CH,AL             ; SAVE
00003D07 F8                  <1>  CLC                     ; SET CY= 0
                             <1> RTC_29:
00003D08 FB                  <1>  sti
00003D09 C3                  <1>  RETn                    ; RETURN WITH RESULT IN CARRY FLAG
                             <1>
                             <1> get_rtc_date:
                             <1>  ; 15/03/2015
                             <1>  ; Derived from IBM PC-XT Model 286 BIOS Source Code
                             <1>  ; BIOS2.ASM ---- 10/06/1985 BIOS INTERRUPT ROUTINES
                             <1>  ; INT 1Ah
                             <1>  ; (AH) = 04H  READ THE DATE FROM THE REAL TIME CLOCK AND RETURN WITH,:
                             <1>  ;      (CH) = CENTURY IN BCD (19 OR 20)                :
                             <1>  ;      (CL) = YEAR IN BCD (00-99)                      :
                             <1>  ;      (DH) = MONTH IN BCD (01-12)                     :
                             <1>  ;      (DL) = DAY IN BCD (01-31).
                             <1>  ;
                             <1> RTC_40:                        ; GET RTC DATE
00003D0A FA                  <1>  cli
00003D0B E83FCFFFFF          <1>  CALL  UPD_IPR           ; CHECK FOR UPDATE IN PROCESS
00003D10 7225                <1>  JC    short RTC_49            ; EXIT IF ERROR (CY= 1)
                             <1>
00003D12 B007                <1>  MOV   AL,CMOS_DAY_MONTH ; ADDRESS DAY OF MONTH
00003D14 E81ECFFFFF          <1>  CALL  CMOS_READ         ; READ DAY OF MONTH
00003D19 88C2                <1>  MOV   DL,AL             ; SAVE
00003D1B B008                <1>  MOV   AL,CMOS_MONTH           ; ADDRESS MONTH
00003D1D E815CFFFFF          <1>  CALL  CMOS_READ         ; READ MONTH
00003D22 88C6                <1>  MOV   DH,AL             ; SAVE
00003D24 B009                <1>  MOV   AL,CMOS_YEAR            ; ADDRESS YEAR
00003D26 E80CCFFFFF          <1>  CALL  CMOS_READ         ; READ YEAR
00003D2B 88C1                <1>  MOV   CL,AL             ; SAVE
00003D2D B032                <1>  MOV   AL,CMOS_CENTURY   ; ADDRESS CENTURY LOCATION
```

```
00003D2F E803CFFFFF          <1>  CALL  CMOS_READ         ; GET CENTURY BYTE
00003D34 88C5                <1>  MOV   CH,AL             ; SAVE
00003D36 F8                  <1>  CLC                     ; SET CY=0
                             <1> RTC_49:
00003D37 FB                  <1>  sti
00003D38 C3                  <1>  RETn                    ; RETURN WITH RESULTS IN CARRY FLAG
                             <1> 
                             <1> set_date_time:
                             <1> convert_from_epoch:
                             <1>  ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
                             <1>  ; 20/06/2013 (Retro UNIX 8086 v1)
                             <1>  ; 'convert_from_epoch' procedure prototype:
                             <1>  ;                 UNIXCOPY.ASM, 10/03/2013
                             <1>  ;
                             <1>  ; ((Modified registers: EAX, EDX, ECX, EBX))
                             <1>  ;
                             <1>  ; Derived from DALLAS Semiconductor
                             <1>  ; Application Note 31 (DS1602/DS1603)
                             <1>  ; 6 May 1998
                             <1>  ;
                             <1>  ; INPUT:
                             <1>  ; EAX = Unix (Epoch) Time
                             <1>  ;
00003D39 31D2                <1>  xor   edx, edx
00003D3B B93C000000          <1>  mov   ecx, 60
00003D40 F7F1                <1>  div   ecx
                             <1>  ;mov  [imin], eax   ; whole minutes
                             <1>                 ; since 1/1/1970
00003D42 668915[B66D0000]    <1>  mov   [second], dx  ; leftover seconds
00003D49 29D2                <1>  sub   edx, edx
00003D4B F7F1                <1>  div   ecx
                             <1>  ;mov  [ihrs], eax   ; whole hours
                             <1>  ;                   ; since 1/1/1970
00003D4D 668915[B46D0000]    <1>  mov   [minute], dx  ; leftover minutes
00003D54 31D2                <1>  xor   edx, edx
                             <1>  ;mov  cx, 24
00003D56 B118                <1>  mov   cl, 24
00003D58 F7F1                <1>  div   ecx
                             <1>  ;mov  [iday], ax   ; whole days
                             <1>                    ; since 1/1/1970
00003D5A 668915[B26D0000]    <1>  mov   [hour], dx   ; leftover hours
00003D61 05DB020000          <1>  add   eax, 365+366 ; whole day since
                             <1>                    ; 1/1/1968
                             <1>  ;mov  [iday], ax
00003D66 50                  <1>  push  eax
00003D67 29D2                <1>  sub   edx, edx
00003D69 B9B5050000          <1>  mov   ecx, (4*365)+1 ; 4 years = 1461 days
00003D6E F7F1                <1>  div   ecx
00003D70 59                  <1>  pop   ecx
                             <1>  ;mov  [lday], ax   ; count of quadyrs (4 years)
00003D71 6652                <1>  push  dx
                             <1>  ;mov  [qday], dx   ; days since quadyr began
00003D73 6683FA3C            <1>  cmp   dx, 31 + 29  ; if past feb 29 then
00003D77 F5                  <1>  cmc              ; add this quadyr's leap day
00003D78 83D000              <1>  adc   eax, 0           ; to # of qadyrs (leap days)
                             <1>  ;mov  [lday], ax   ; since 1968
                             <1>  ;mov  cx, [iday]
00003D7B 91                  <1>  xchg  ecx, eax     ; ECX = lday, EAX = iday
00003D7C 29C8                <1>  sub   eax, ecx     ; iday - lday
00003D7E B96D010000          <1>  mov   ecx, 365
00003D83 31D2                <1>  xor   edx, edx
                             <1>  ; EAX = iday-lday, EDX = 0
00003D85 F7F1                <1>  div   ecx
                             <1>  ;mov  [iyrs], ax   ; whole years since 1968
                             <1>  ;jday = iday - (iyrs*365) - lday
                             <1>  ;mov [jday], dx      ; days since 1/1 of current year
                             <1>  ;add  eax, 1968
00003D87 6605B007            <1>  add   ax, 1968     ; compute year
00003D8B 66A3[AC6D0000]      <1>  mov   [year], ax
00003D91 6689D1              <1>  mov   cx, dx
                             <1>  ;mov  dx, [qday]
00003D94 665A                <1>  pop   dx
00003D96 6681FA6D01          <1>  cmp   dx, 365          ; if qday <= 365 and qday >= 60
00003D9B 7709                <1>  ja    short cfe1   ; jday = jday +1
00003D9D 6683FA3C            <1>  cmp   dx, 60       ; if past 2/29 and leap year then
00003DA1 F5                  <1>        cmc              ; add a leap day to the # of whole
00003DA2 6683D100            <1>  adc   cx, 0        ; days since 1/1 of current year
                             <1> cfe1:
                             <1>  ;mov  [jday], cx
00003DA6 66BB0C00            <1>  mov   bx, 12       ; estimate month
00003DAA 66BA6E01            <1>  mov   dx, 366      ; mday, max. days since 1/1 is 365
00003DAE 6683E003            <1>  and   ax, 11b      ; year mod 4 (and dx, 3)
                             <1> cfe2:  ; Month calculation  ; 0 to 11  (11 to 0)
00003DB2 6639D1              <1>  cmp   cx, dx       ; mday = # of days passed from 1/1
00003DB5 731D                <1>  jnb   short cfe3
00003DB7 664B                <1>  dec   bx           ; month = month - 1
00003DB9 66D1E3              <1>  shl   bx, 1
00003DBC 668B93[B86D0000]    <1>  mov   dx, [EBX+DMonth] ; # elapsed days at 1st of month
00003DC3 66D1EB              <1>  shr   bx, 1        ; bx = month - 1 (0 to 11)
00003DC6 6683FB01            <1>  cmp   bx, 1        ; if month > 2 and year mod 4  = 0
00003DCA 76E6                <1>  jna   short cfe2   ; then mday = mday + 1
00003DCC 08C0                <1>  or    al, al       ; if past 2/29 and leap year then
```

```
00003DCE 75E2                <1>  jnz   short cfe2   ; add leap day (to mday)
00003DD0 6642                <1>  inc   dx           ; mday = mday + 1
00003DD2 EBDE                <1>  jmp   short cfe2
                             <1> cfe3:
00003DD4 6643                <1>  inc   bx         ; -> bx = month, 1 to 12
00003DD6 66891D[AE6D0000]    <1>  mov   [month], bx
00003DDD 6629D1              <1>  sub   cx, dx           ; day = jday - mday + 1
00003DE0 6641                <1>  inc   cx
00003DE2 66890D[B06D0000]    <1>  mov   [day], cx
                             <1>
                             <1>  ; eax, ebx, ecx, edx is changed at return
                             <1>  ; output ->
                             <1>  ; [year], [month], [day], [hour], [minute], [second]
                             <1>
                             <1>  ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
                             <1>  ; 20/06/2013 (Retro UNIX 8086 v1)
                             <1> set_date:
00003DE9 A0[AD6D0000]        <1>         mov     al, [year+1]
00003DEE D40A                <1>  aam   ; ah = al / 10, al = al mod 10
00003DF0 D510                <1>  db    0D5h,10h     ; Undocumented inst. AAD
                             <1>                   ; AL = AH * 10h + AL
00003DF2 88C5                <1>  mov   ch, al ; century (BCD)
00003DF4 A0[AC6D0000]        <1>  mov   al, [year]
00003DF9 D40A                <1>  aam   ; ah = al / 10, al = al mod 10
00003DFB D510                <1>  db    0D5h,10h     ; Undocumented inst. AAD
                             <1>                   ; AL = AH * 10h + AL
00003DFD 88C1                <1>  mov   cl, al ; year (BCD)
00003DFF A0[AE6D0000]        <1>         mov  al, [month]
00003E04 D40A                <1>  aam   ; ah = al / 10, al = al mod 10
00003E06 D510                <1>  db    0D5h,10h     ; Undocumented inst. AAD
                             <1>                   ; AL = AH * 10h + AL
00003E08 88C6                <1>  mov   dh, al ; month (BCD)
00003E0A A0[B06D0000]        <1>  mov   al, [day]
00003E0F D40A                <1>  aam   ; ah = al / 10, al = al mod 10
00003E11 D510                <1>  db    0D5h,10h     ; Undocumented inst. AAD
                             <1>                   ; AL = AH * 10h + AL
00003E13 88C6                <1>  mov   dh, al ; day (BCD)
                             <1>  ; Set real-time clock date
00003E15 E879000000          <1>  call  set_rtc_date
                             <1> set_time:
                             <1>         ; Read real-time clock time
                             <1>  ; (get day light saving time bit status)
00003E1A FA                  <1>  cli
00003E1B E82FCEFFFF          <1>  CALL  UPD_IPR           ; CHECK FOR UPDATE IN PROCESS
                             <1>  ; cf = 1 -> al = 0
00003E20 7207                <1>         jc      short stime1
00003E22 B00B                <1>  MOV   AL,CMOS_REG_B           ; ADDRESS ALARM REGISTER
00003E24 E80ECEFFFF          <1>  CALL  CMOS_READ         ; READ CURRENT VALUE OF DSE BIT
                             <1> stime1:
00003E29 FB                  <1>  sti
00003E2A 2401                <1>  AND   AL,00000001B            ; MASK FOR VALID DSE BIT
00003E2C 88C2                <1>  MOV   DL,AL             ; SET [DL] TO ZERO FOR NO DSE BIT
                             <1>  ; DL = 1 or 0 (day light saving time)
                             <1>  ;
00003E2E A0[B26D0000]        <1>  mov   al, [hour]
00003E33 D40A                <1>  aam   ; ah = al / 10, al = al mod 10
00003E35 D510                <1>  db    0D5h,10h     ; Undocumented inst. AAD
                             <1>                   ; AL = AH * 10h + AL
00003E37 88C5                <1>  mov   ch, al ; hour (BCD)
00003E39 A0[B46D0000]        <1>         mov     al, [minute]
00003E3E D40A                <1>  aam   ; ah = al / 10, al = al mod 10
00003E40 D510                <1>  db    0D5h,10h     ; Undocumented inst. AAD
                             <1>                   ; AL = AH * 10h + AL
00003E42 88C1                <1>  mov   cl, al       ; minute (BCD)
00003E44 A0[B66D0000]        <1>         mov     al, [second]
00003E49 D40A                <1>  aam   ; ah = al / 10, al = al mod 10
00003E4B D510                <1>  db    0D5h,10h     ; Undocumented inst. AAD
                             <1>                   ; AL = AH * 10h + AL
00003E4D 88C6                <1>  mov   dh, al           ; second (BCD)
                             <1>  ; Set real-time clock time
                             <1>  ; call      set_rtc_time
                             <1> set_rtc_time:
                             <1>  ; 15/04/2015 (257, POSTEQU.INC -> H EQU 256, X EQU H+1)
                             <1>  ; 15/03/2015
                             <1>  ; Derived from IBM PC-XT Model 286 BIOS Source Code
                             <1>  ; BIOS2.ASM ---- 10/06/1985 BIOS INTERRUPT ROUTINES
                             <1>  ; INT 1Ah
                             <1>  ; (AH) = 03H  SET THE REAL TIME CLOCK USING,               :
                             <1>  ;      (CH) = HOURS IN BCD (00-23)                           :
                             <1>  ;      (CL) = MINUTES IN BCD (00-59)                               :
                             <1>  ;      (DH) = SECONDS IN BCD (00-59)                               :
                             <1>  ;      (DL) = 01 IF DAYLIGHT SAVINGS ENABLE OPTION, ELSE 00.    :
                             <1>  ;                                                :
                             <1>  ;  NOTE: (DL)= 00 IF DAYLIGHT SAVINGS TIME ENABLE IS NOT ENABLED. :
                             <1>  ;        (DL)= 01 ENABLES TWO SPECIAL UPDATES THE LAST SUNDAY IN  :
                             <1>  ;         APRIL   (1:59:59 --> 3:00:00 AM) AND THE LAST SUNDAY IN :
                             <1>  ;         OCTOBER (1:59:59 --> 1:00:00 AM) THE FIRST TIME.   :
                             <1>  ;
                             <1> RTC_30:                        ; SET RTC TIME
00003E4F FA                  <1>  cli
00003E50 E8FACDFFFF          <1>  CALL  UPD_IPR           ; CHECK FOR UPDATE IN PROCESS
00003E55 7305                <1>  JNC   short RTC_35            ; GO AROUND IF CLOCK OPERATING
```

```
00003E57 E886000000          <1>  CALL  RTC_STA          ; ELSE TRY INITIALIZING CLOCK
                             <1> RTC_35:
00003E5C 88F4                <1>  MOV   AH,DH            ; GET TIME BYTE - SECONDS
00003E5E B000                <1>  MOV   AL,CMOS_SECONDS  ; ADDRESS SECONDS
00003E60 E89E000000          <1>  CALL  CMOS_WRITE       ; UPDATE SECONDS
00003E65 88CC                <1>  MOV   AH,CL            ; GET TIME BYTE - MINUTES
00003E67 B002                <1>  MOV   AL,CMOS_MINUTES  ; ADDRESS MINUTES
00003E69 E895000000          <1>  CALL  CMOS_WRITE       ; UPDATE MINUTES
00003E6E 88EC                <1>  MOV   AH,CH            ; GET TIME BYTE - HOURS
00003E70 B004                <1>  MOV   AL,CMOS_HOURS          ; ADDRESS HOURS
00003E72 E88C000000          <1>  CALL  CMOS_WRITE       ; UPDATE ADDRESS
                             <1>  ;MOV  AX,X*CMOS_REG_B  ; ADDRESS ALARM REGISTER
00003E77 66B80B0B            <1>  MOV   AX,257*CMOS_REG_B      ;
00003E7B E8B7CDFFFF          <1>  CALL  CMOS_READ        ; READ CURRENT TIME
00003E80 2462                <1>  AND   AL,01100010B           ; MASK FOR VALID BIT POSITIONS
00003E82 0C02                <1>  OR    AL,00000010B           ; TURN ON 24 HOUR MODE
00003E84 80E201              <1>  AND   DL,00000001B           ; USE ONLY THE DSE BIT
00003E87 08D0                <1>  OR    AL,DL            ; GET DAY LIGHT SAVINGS TIME BIT (OSE)
00003E89 86E0                <1>  XCHG  AH,AL            ; PLACE IN WORK REGISTER AND GET ADDRESS
00003E8B E873000000          <1>  CALL  CMOS_WRITE       ; SET NEW ALARM BITS
00003E90 F8                  <1>  CLC                    ; SET CY= 0
00003E91 FB                  <1>  sti
00003E92 C3                  <1>  RETn                   ; RETURN WITH CY= 0
                             <1>
                             <1> set_rtc_date:
                             <1>  ; 15/04/2015 (257, POSTEQU.INC -> H EQU 256, X EQU H+1)
                             <1>  ; 15/03/2015
                             <1>  ; Derived from IBM PC-XT Model 286 BIOS Source Code
                             <1>  ; BIOS2.ASM ---- 10/06/1985 BIOS INTERRUPT ROUTINES
                             <1>  ; INT 1Ah
                             <1>  ; (AH) = 05H  SET THE DATE INTO THE REAL TIME CLOCK USING, :
                             <1>  ;     (CH) = CENTURY IN BCD (19 OR 20)                 :
                             <1>  ;     (CL) = YEAR IN BCD (00-99)                 :
                             <1>  ;     (DH) = MONTH IN BCD (01-12)                :
                             <1>  ;     (DL) = DAY IN BCD (01-31).
                             <1>  ;
                             <1> RTC_50:                        ; SET RTC DATE
00003E93 FA                  <1>  cli
00003E94 E8B6CDFFFF          <1>  CALL  UPD_IPR          ; CHECK FOR UPDATE IN PROCESS
00003E99 7305                <1>  JNC   short RTC_55           ; GO AROUND IF NO ERROR
00003E9B E842000000          <1>  CALL  RTC_STA          ; ELSE INITIALIZE CLOCK
                             <1> RTC_55:
00003EA0 66B80600            <1>  MOV   AX,CMOS_DAY_WEEK ; ADDRESS OF DAY OF WEEK BYTE
00003EA4 E85A000000          <1>  CALL  CMOS_WRITE       ; LOAD ZEROS TO DAY OF WEEK
00003EA9 88D4                <1>  MOV   AH,DL            ; GET DAY OF MONTH BYTE
00003EAB B007                <1>  MOV   AL,CMOS_DAY_MONTH ; ADDRESS DAY OF MONTH BYTE
00003EAD E851000000          <1>  CALL  CMOS_WRITE       ; WRITE OF DAY OF MONTH REGISTER
00003EB2 88F4                <1>  MOV   AH,DH            ; GET MONTH
00003EB4 B008                <1>  MOV   AL,CMOS_MONTH          ; ADDRESS MONTH BYTE
00003EB6 E848000000          <1>  CALL  CMOS_WRITE       ; WRITE MONTH REGISTER
00003EBB 88CC                <1>  MOV   AH,CL            ; GET YEAR BYTE
00003EBD B009                <1>  MOV   AL,CMOS_YEAR           ; ADDRESS YEAR REGISTER
00003EBF E83F000000          <1>  CALL  CMOS_WRITE       ; WRITE YEAR REGISTER
00003EC4 88EC                <1>  MOV   AH,CH            ; GET CENTURY BYTE
00003EC6 B032                <1>  MOV   AL,CMOS_CENTURY  ; ADDRESS CENTURY BYTE
00003EC8 E836000000          <1>  CALL  CMOS_WRITE       ; WRITE CENTURY LOCATION
                             <1>  ;MOV  AX,X*CMOS_REG_B  ; ADDRESS ALARM REGISTER
00003ECD 66B80B0B            <1>  MOV   AX,257*CMOS_REG_B      ;
00003ED1 E861CDFFFF          <1>  CALL  CMOS_READ        ; READ CURRENT SETTINGS
00003ED6 247F                <1>  AND   AL,07FH          ; CLEAR 'SET BIT'
00003ED8 86E0                <1>  XCHG  AH,AL            ; MOVE TO WORK REGISTER
00003EDA E824000000          <1>  CALL  CMOS_WRITE       ; AND START CLOCK UPDATING
00003EDF F8                  <1>  CLC                    ; SET CY= 0
00003EE0 FB                  <1>  sti
00003EE1 C3                  <1>  RETn                   ; RETURN CY=0
                             <1>
                             <1>  ; 15/03/2015
                             <1> RTC_STA:                       ; INITIALIZE REAL TIME CLOCK
00003EE2 B426                <1>  mov   ah, 26h
00003EE4 B00A                <1>  mov   al, CMOS_REG_A         ; ADDRESS REGISTER A AND LOAD DATA MASK
00003EE6 E818000000          <1>  CALL  CMOS_WRITE       ; INITIALIZE STATUS REGISTER A
00003EEB B482                <1>  mov   ah, 82h
00003EED B00B                <1>  mov   al, CMOS_REG_B         ; SET "SET BIT" FOR CLOCK INITIALIZATION
00003EEF E80F000000          <1>  CALL  CMOS_WRITE       ; AND 24 HOUR MODE TO REGISTER B
00003EF4 B00C                <1>  MOV   AL,CMOS_REG_C          ; ADDRESS REGISTER C
00003EF6 E83CCDFFFF          <1>  CALL  CMOS_READ        ; READ REGISTER C TO INITIALIZE
00003EFB B00D                <1>  MOV   AL,CMOS_REG_D          ; ADDRESS REGISTER D
00003EFD E835CDFFFF          <1>  CALL  CMOS_READ        ; READ REGISTER D TO INITIALIZE
00003F02 C3                  <1>  RETn
                             <1>
                             <1>  ; 15/03/2015
                             <1>  ; IBM PC/XT Model 286 BIOS source code ----- 10/06/85 (test4.asm)
                             <1> CMOS_WRITE:              ; WRITE (AH) TO LOCATION (AL)
00003F03 9C                  <1>  pushf             ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
                             <1>  ;push ax          ; SAVE WORK REGISTER VALUES
00003F04 D0C0                <1>  rol   al, 1       ; MOVE NMI BIT TO LOW POSITION
00003F06 F9                  <1>  stc               ; FORCE NMI BIT ON IN CARRY FLAG
00003F07 D0D8                <1>  rcr   al, 1       ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
00003F09 FA                  <1>  cli               ; DISABLE INTERRUPTS
00003F0A E670                <1>  out   CMOS_PORT, al     ; ADDRESS LOCATION AND DISABLE NMI
00003F0C 88E0                <1>  mov   al, ah           ; GET THE DATA BYTE TO WRITE
00003F0E E671                <1>  out   CMOS_DATA, al     ; PLACE IN REQUESTED CMOS LOCATION
```

```
00003F10 B01E                <1>  mov   al, CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
00003F12 D0D8                <1>  rcr   al, 1       ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
00003F14 E670                <1>  out   CMOS_PORT, al     ; SET DEFAULT TO READ ONLY REGISTER
00003F16 90                  <1>  nop               ; I/O DELAY
00003F17 E471                <1>  in    al, CMOS_DATA     ; OPEN STANDBY LATCH
                             <1>  ;pop  ax          ; RESTORE WORK REGISTERS
00003F19 9D                  <1>  popf
00003F1A C3                  <1>  RETn
                             <1>
                             <1> bf_init:
                             <1>  ; 14/08/2015
                             <1>  ; 02/07/2015
                             <1>  ; 01/07/2015
                             <1>  ; 15/04/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; Buffer (pointer) initialization !
                             <1>  ;
                             <1>  ; 17/07/2013 - 24/07/2013
                             <1>  ; Retro UNIX 8086 v1 (U9.ASM)
                             <1>  ; (Retro UNIX 8086 v1 feature only !)
                             <1>  ;
00003F1B BF[4A740000]        <1>  mov   edi, bufp
00003F20 B8[207D0000]        <1>  mov   eax, buffer + (nbuf*520)
00003F25 29D2                <1>  sub   edx, edx
00003F27 FECA                <1>  dec   dl
00003F29 31C9                <1>  xor   ecx, ecx
00003F2B 49                  <1>  dec   ecx
                             <1> bi0:
00003F2C 2D08020000          <1>  sub   eax, 520 ; 8 header + 512 data
00003F31 AB                  <1>  stosd
00003F32 89C6                <1>  mov   esi, eax
00003F34 8916                <1>  mov   [esi], edx ; 000000FFh
                             <1>                  ; Not a valid device sign
00003F36 894E04              <1>  mov   [esi+4], ecx ; 0FFFFFFFFh
                             <1>              ; Not a valid block number sign
00003F39 3D[00750000]        <1>  cmp   eax, buffer
00003F3E 77EC                <1>  ja    short bi0
00003F40 B8[207D0000]        <1>  mov   eax, sb0
00003F45 AB                  <1>  stosd
00003F46 B8[287F0000]        <1>  mov   eax, sb1
00003F4B AB                  <1>  stosd
00003F4C 89C6                <1>  mov   esi, eax ; offset sb1
00003F4E 8916                <1>  mov   [esi], edx ; 000000FFh
                             <1>                  ; Not a valid device sign
00003F50 894E04              <1>  mov   [esi+4], ecx ; 0FFFFFFFFh
                             <1>              ; Not a valid block number sign
                             <1>  ; 14/08/2015
                             <1>  ;call       rdev_init
                             <1>  ;retn
                             <1>
                             <1> rdev_init: ; root device, super block buffer initialization
                             <1>  ; 14/08/2015
                             <1>  ; Retro UNIX 386 v1 feature only !
                             <1>  ;
                             <1>  ; NOTE: Disk partitions (file systems), logical
                             <1>  ; drive initialization, partition's start sector etc.
                             <1>  ; will be coded here, later in 'ldrv_init'
                             <1>
00003F53 0FB605[D46D0000]    <1>  movzx eax, byte [boot_drv]
                             <1> rdi_0:
00003F5A 3C80                <1>  cmp   al, 80h
00003F5C 7202                <1>  jb    short rdi_1
00003F5E 2C7E                <1>  sub   al, 7Eh ; 80h = 2 (hd0), 81h = 3 (hd1)
                             <1> rdi_1:
00003F60 A2[68740000]        <1>  mov   [rdev], al
00003F65 BB[207D0000]        <1>         mov  ebx, sb0 ; super block buffer
00003F6A 8903                <1>  mov   [ebx], eax
00003F6C B001                <1>  mov   al, 1 ; eax = 1
00003F6E 894304              <1>  mov   [ebx+4], eax ; super block address on disk
00003F71 E82A240000          <1>  call  diskio
00003F76 C3                  <1>  retn
                             <1>
                             <1> ; 23/10/2015
                             <1> com1_irq4:
00003F77 [7F3F0000]          <1>  dd dummy_retn
                             <1> com2_irq3:
00003F7B [7F3F0000]          <1>  dd dummy_retn
                             <1>
                             <1> dummy_retn:
00003F7F C3                  <1>  retn
                                 %include 'u1.s'        ; 10/05/2015
                             <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS1.INC
                             <1> ; Last Modification: 23/11/2015
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
                             <1> ; (v0.1 - Beginning: 11/07/2012)
                             <1> ;
                             <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                             <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                             <1> ; <Bell Laboratories (17/3/1972)>
                             <1> ; <Preliminary Release of UNIX Implementation Document>
                             <1> ;
                             <1> ; Retro UNIX 8086 v1 - U1.ASM (12/07/2014) //// UNIX v1 -> u1.s
                             <1> ;
                             <1> ; ****************************************************************************
                             <1>
                             <1> unkni: ; / used for all system calls
                             <1> sysent: ; < enter to system call >
                             <1>   ;19/10/2015
                             <1>  ; 21/09/2015
                             <1>  ; 01/07/2015
                             <1>  ; 19/05/2015
                             <1>  ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 10/04/2013 - 18/01/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'unkni' or 'sysent' is sytem entry from various traps.
                             <1>  ; The trap type is determined and an indirect jump is made to
                             <1>  ; the appropriate system call handler. If there is a trap inside
                             <1>  ; the system a jump to panic is made. All user registers are saved
                             <1>  ; and u.sp points to the end of the users stack. The sys (trap)
                             <1>  ; instructor is decoded to get the the system code part (see
                             <1>  ; trap instruction in the PDP-11 handbook) and from this
                             <1>  ; the indirect jump address is calculated. If a bad system call is
                             <1>  ; made, i.e., the limits of the jump table are exceeded, 'badsys'
                             <1>  ; is called. If the call is legitimate control passes to the
                             <1>  ; appropriate system routine.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     Through a trap caused by any sys call outside the system.
                             <1>  ; Arguments:
                             <1>  ;     Arguments of particular system call.
                             <1>  ; ..............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       System call number is in EAX register.
                             <1>  ;
                             <1>  ;       Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
                             <1>  ;     registers depending of function details.
                             <1>        ;
                             <1>  ; 16/04/2015
00003F80 368925[78740000]    <1>         mov     [ss:u.sp], esp ; Kernel stack points to return address
                             <1>  ; save user registers
00003F87 1E                  <1>  push  ds
00003F88 06                  <1>  push  es
00003F89 0FA0                <1>  push  fs
00003F8B 0FA8                <1>  push  gs
00003F8D 60                  <1>  pushad  ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
                             <1>  ;
                             <1>  ; ESPACE = esp - [ss:u.sp] ; 4*12 = 48 ; 17/09/2015
                             <1>  ;     (ESPACE is size of space in kernel stack
                             <1>  ;     for saving/restoring user registers.)
                             <1>  ;
00003F8E 50                  <1>  push  eax ; 01/07/2015
00003F8F 66B81000            <1>  mov     ax, KDATA
00003F93 8ED8                <1>         mov     ds, ax
00003F95 8EC0                <1>         mov     es, ax
00003F97 8EE0                <1>         mov     fs, ax
00003F99 8EE8                <1>         mov     gs, ax
00003F9B A1[A8700000]        <1>  mov   eax, [k_page_dir]
00003FA0 0F22D8              <1>  mov   cr3, eax
00003FA3 58                  <1>  pop   eax ; 01/07/2015
                             <1>  ; 19/10/2015
00003FA4 FC                  <1>  cld
                             <1>  ;
00003FA5 FE05[77740000]      <1>  inc   byte [sysflg]
                             <1>        ; incb sysflg / indicate a system routine is in progress
00003FAB FB                  <1>         sti   ; 18/01/2014
00003FAC 0F8560F9FFFF        <1>  jnz     panic ; 24/05/2013
                             <1>        ; beq 1f
                             <1>        ; jmp panic ; / called if trap inside system
                             <1> ;1:
                             <1>  ; 16/04/2015
00003FB2 A3[80740000]        <1>  mov   [u.r0], eax
00003FB7 8925[7C740000]      <1>  mov   [u.usp], esp ; kernel stack points to user's registers
                             <1>  ;
                             <1>        ; mov $s.syst+2,clockp
                             <1>        ; mov r0,-(sp) / save user registers
                             <1>        ; mov sp,u.r0 / pointer to bottom of users stack
                             <1>                ; / in u.r0
                             <1>        ; mov r1,-(sp)
                             <1>        ; mov r2,-(sp)
                             <1>        ; mov r3,-(sp)
                             <1>        ; mov r4,-(sp)
                             <1>        ; mov r5,-(sp)
                             <1>        ; mov ac,-(sp) / "accumulator" register for extended
                             <1>                    ; / arithmetic unit
                             <1>        ; mov mq,-(sp) / "multiplier quotient" register for the
                             <1>                    ; / extended arithmetic unit
                             <1>        ; mov sc,-(sp) / "step count" register for the extended
                             <1>                    ; / arithmetic unit
                             <1>        ; mov sp,u.sp / u.sp points to top of users stack
                             <1>        ; mov 18.(sp),r0 / store pc in r0
                             <1>        ; mov -(r0),r0 / sys inst in r0      10400xxx
                             <1>        ; sub $sys,r0 / get xxx code
00003FBD C1E002              <1>  shl   eax, 2
```

```
                            <1>        ; asl r0 / multiply by 2 to jump indirect in bytes
00003FC0 3D94000000         <1>  cmp   eax, end_of_syscalls - syscalls
                            <1>        ; cmp r0,$2f-1f / limit of table (35) exceeded
                            <1>  ;jnb  short badsys
                            <1>        ; bhis badsys / yes, bad system call
00003FC5 F5                 <1>  cmc
00003FC6 9C                 <1>  pushf
00003FC7 50                 <1>  push  eax
00003FC8 8B2D[78740000]     <1>  mov   ebp, [u.sp] ; Kernel stack at the beginning of sys call
00003FCE B0FE               <1>  mov   al, 0FEh ; 11111110b
00003FD0 1400               <1>  adc   al, 0 ; al = al + cf
00003FD2 204508             <1>  and   [ebp+8], al ; flags (reset carry flag)
                            <1>        ; bic $341,20.(sp) / set users processor priority to 0
                            <1>                        ; / and clear carry bit
00003FD5 5D                 <1>  pop   ebp ; eax
00003FD6 9D                 <1>  popf
00003FD7 0F8248010000       <1>         jc      badsys
00003FDD A1[80740000]       <1>  mov   eax, [u.r0]
                            <1>  ; system call registers: EAX, EDX, ECX, EBX, ESI, EDI
00003FE2 FFA5[E83F0000]     <1>  jmp   dword [ebp+syscalls]
                            <1>        ; jmp *1f(r0) / jump indirect thru table of addresses
                            <1>                       ; / to proper system routine.
                            <1> syscalls: ; 1:
                            <1>  ; 21/09/2015
                            <1>  ; 01/07/2015
                            <1>  ; 16/04/2015 (32 bit address modification)
00003FE8 [EF400000]         <1>  dd sysrele  ; / 0
00003FEC [95410000]         <1>  dd sysexit  ; / 1
00003FF0 [BA420000]         <1>  dd sysfork  ; / 2
00003FF4 [CD430000]         <1>  dd sysread  ; / 3
00003FF8 [E8430000]         <1>  dd syswrite       ; / 4
00003FFC [52440000]         <1>  dd sysopen  ; / 5
00004000 [82450000]         <1>  dd sysclose       ; / 6
00004004 [3C420000]         <1>  dd syswait  ; / 7
00004008 [02450000]         <1>  dd syscreat       ; / 8
0000400C [A9480000]         <1>  dd syslink  ; / 9
00004010 [6B490000]         <1>  dd sysunlink      ; / 10
00004014 [3B4A0000]         <1>  dd sysexec  ; / 11
00004018 [A2500000]         <1>  dd syschdir       ; / 12
0000401C [86510000]         <1>  dd systime  ; / 13
00004020 [39450000]         <1>  dd sysmkdir       ; / 14
00004024 [F4500000]         <1>  dd syschmod       ; / 15
00004028 [56510000]         <1>  dd syschown       ; / 16
0000402C [B9510000]         <1>  dd sysbreak       ; / 17
00004030 [134E0000]         <1>  dd sysstat  ; / 18
00004034 [7E520000]         <1>  dd sysseek  ; / 19
00004038 [90520000]         <1>  dd systell  ; / 20
0000403C [8C5D0000]         <1>  dd sysmount       ; / 21
00004040 [3E5E0000]         <1>  dd sysumount      ; / 22
00004044 [0D530000]         <1>  dd syssetuid      ; / 23
00004048 [3E530000]         <1>  dd sysgetuid      ; / 24
0000404C [95510000]         <1>  dd sysstime       ; / 25
00004050 [01530000]         <1>  dd sysquit  ; / 26
00004054 [F5520000]         <1>  dd sysintr  ; / 27
00004058 [EF4D0000]         <1>  dd sysfstat       ; / 28
0000405C [9E450000]         <1>  dd sysemt   ; / 29
00004060 [CC450000]         <1>  dd sysmdate       ; / 30
00004064 [17460000]         <1>  dd sysstty  ; / 31
00004068 [96470000]         <1>  dd sysgtty  ; / 32
0000406C [C7450000]         <1>  dd sysilgins      ; / 33
00004070 [7A660000]         <1>  dd syssleep       ; 34 ; Retro UNIX 8086 v1 feature only !
                            <1>                  ; 11/06/2014
00004074 [A9660000]         <1>  dd sysmsg   ; 35 ; Retro UNIX 386 v1 feature only !
                            <1>                  ; 01/07/2015
00004078 [80670000]         <1>  dd sysgeterr      ; 36 ; Retro UNIX 386 v1 feature only !
                            <1>                  ; 21/09/2015 - get last error number
                            <1> end_of_syscalls:
                            <1>
                            <1> error:
                            <1>  ; 17/09/2015
                            <1>  ; 03/09/2015
                            <1>  ; 01/09/2015
                            <1>  ; 09/06/2015
                            <1>  ; 13/05/2015
                            <1>  ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
                            <1>  ; 10/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)
                            <1>  ;
                            <1>  ; 'error' merely sets the error bit off the processor status (c-bit)
                            <1>  ; then falls right into the 'sysret', 'sysrele' return sequence.
                            <1>  ;
                            <1>  ; INPUTS -> none
                            <1>  ; OUTPUTS ->
                            <1>  ;     processor status - carry (c) bit is set (means error)
                            <1>  ;
                            <1>  ; 26/05/2013 (Stack pointer must be reset here!
                            <1>  ;           Because, jumps to error procedure
                            <1>  ;           disrupts push-pop nesting balance)
                            <1>  ;
0000407C 8B2D[78740000]     <1>  mov   ebp, [u.sp] ; interrupt (system call) return (iretd) address
00004082 804D0801           <1>  or    byte [ebp+8], 1  ; set carry bit of flags register
                            <1>                        ; (system call will return with cf = 1)
                            <1>        ; bis $1,20.(r1) / set c bit in processor status word below
```

```
                             <1>                       ; / users stack
                             <1>  ; 17/09/2015
00004086 83ED30              <1>  sub   ebp, ESPACE ; 48 ; total size of stack frame ('sysdefs.inc')
                             <1>                     ; for saving/restoring user registers
                             <1>  ;cmp  ebp, [u.usp]
                             <1>  ;je   short err0
00004089 892D[7C740000]      <1>  mov   [u.usp], ebp
                             <1> ;err0:
                             <1>  ; 01/09/2015
0000408F 8B25[7C740000]      <1>  mov   esp, [u.usp]          ; Retro Unix 8086 v1 modification!
                             <1>                        ; 10/04/2013
                             <1>                        ; (If an I/O error occurs during disk I/O,
                             <1>                        ; related procedures will jump to 'error'
                             <1>                        ; procedure directly without returning to
                             <1>                        ; the caller procedure. So, stack pointer
                             <1>                                     ; must be restored here.)
                             <1>  ; 13/05/2015
                             <1>  ; NOTE: (The last) error code is in 'u.error', it can be retrieved by
                             <1>  ;     'get last error' system call later.
                             <1>
                             <1>  ; 03/09/2015 - 09/06/2015 - 07/08/2013
00004095 C605[E1740000]00    <1>  mov   byte [u.kcall], 0 ; namei_r, mkdir_w reset
                             <1>
                             <1> sysret: ; < return from system call>
                             <1>  ; 10/09/2015
                             <1>  ; 29/07/2015
                             <1>  ; 25/06/2015
                             <1>  ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 10/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysret' first checks to see if process is about to be
                             <1>  ; terminated (u.bsys). If it is, 'sysexit' is called.
                             <1>  ; If not, following happens:
                             <1>  ;     1) The user's stack pointer is restored.
                             <1>  ;     2) r1=0 and 'iget' is called to see if last mentioned
                             <1>  ;        i-node has been modified. If it has, it is written out
                             <1>  ;        via 'ppoke'.
                             <1>  ;     3) If the super block has been modified, it is written out
                             <1>  ;        via 'ppoke'.
                             <1>  ;     4) If the dismountable file system's super block has been
                             <1>  ;        modified, it is written out to the specified device
                             <1>  ;        via 'ppoke'.
                             <1>  ;     5) A check is made if user's time quantum (uquant) ran out
                             <1>  ;        during his execution. If so, 'tswap' is called to give
                             <1>  ;        another user a chance to run.
                             <1>  ;     6) 'sysret' now goes into 'sysrele'.
                             <1>  ;         (See 'sysrele' for conclusion.)
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     jump table or 'br sysret'
                             <1>  ; Arguments:
                             <1>  ;     -
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; ((AX=r1 for 'iget' input))
                             <1>  ;
0000409C 6631C0              <1>  xor   ax, ax ; 04/05/2013
                             <1> sysret0: ; 29/07/2015 (eax = 0, jump from sysexec)
0000409F FEC0                <1>  inc   al ; 04/05/2013
000040A1 3805[C8740000]      <1>  cmp   [u.bsys], al ; 1
                             <1>        ; tstb u.bsys / is a process about to be terminated because
000040A7 0F83E8000000        <1>         jnb     sysexit ; 04/05/2013
                             <1>        ; bne sysexit / of an error? yes, go to sysexit
                             <1>  ;mov  esp, [u.usp] ; 24/05/2013 (that is not needed here)
                             <1>        ; mov u.sp,sp / no point stack to users stack
000040AD FEC8                <1>  dec   al ; mov ax, 0
                             <1>        ; clr r1 / zero r1 to check last mentioned i-node
000040AF E866160000          <1>  call  iget
                             <1>        ; jsr r0,iget / if last mentioned i-node has been modified
                             <1>                    ; / it is written out
000040B4 6631C0              <1>  xor   ax, ax ; 0
000040B7 3805[75740000]      <1>  cmp   [smod], al ; 0
                             <1>        ; tstb      smod / has the super block been modified
000040BD 7614                <1>  jna   short sysret1
                             <1>        ; beq 1f / no, 1f
000040BF A2[75740000]        <1>  mov   [smod], al ; 0
                             <1>        ; clrb smod / yes, clear smod
000040C4 BB[207D0000]        <1>  mov   ebx, sb0 ;; 07/08//2013
000040C9 66810B0002          <1>        or    word [ebx], 200h ;;
                             <1>  ;or   word [sb0], 200h ; write bit, bit 9
                             <1>        ; bis $1000,sb0 / set write bit in I/O queue for super block
                             <1>                          ; / output
                             <1>  ; AX = 0
000040CE E8B2210000          <1>  call  poke ; 07/08/2013
                             <1>  ; call      ppoke
                             <1>  ; AX = 0
                             <1>        ; jsr r0,ppoke / write out modified super block to disk
                             <1> sysret1: ;1:
000040D3 3805[76740000]      <1>  cmp   [mmod], al ; 0
                             <1>        ; tstb      mmod / has the super block for the dismountable file
                             <1>                   ; / system
000040D9 7614                <1>  jna   short sysrel0
```

```
                             <1>        ; beq 1f / been modified?  no, 1f
000040DB A2[76740000]        <1>  mov   [mmod], al ; 0
                             <1>        ; clrb      mmod / yes, clear mmod
                             <1>         ;mov    ax, [mntd]
                             <1>         ;;mov   al, [mdev] ; 26/04/2013
000040E0 BB[287F0000]        <1>  mov   ebx, sb1 ;; 07/08//2013
                             <1>         ;;mov[ebx], al
                             <1>  ;mov    [sb1], al
                             <1>        ; movb      mntd,sb1 / set the I/O queue
000040E5 66810B0002          <1>  or    word [ebx], 200h
                             <1>  ;or   word [sb1], 200h ; write bit, bit 9
                             <1>        ; bis $1000,sb1 / set write bit in I/O queue for detached sb
000040EA E896210000          <1>  call  poke ; 07/08/2013
                             <1>  ;call ppoke
                             <1>        ; jsr r0,ppoke / write it out to its device
                             <1>         ;xor    al, al ; 26/04/2013
                             <1> ;1:
                             <1>        ; tstb uquant / is the time quantum 0?
                             <1>        ; bne 1f / no, don't swap it out
                             <1>
                             <1> sysrele: ; < release >
                             <1>  ; 14/10/2015
                             <1>  ; 01/09/2015
                             <1>  ; 24/07/2015
                             <1>  ; 14/05/2015
                             <1>  ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 10/04/2013 - 07/03/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysrele' first calls 'tswap' if the time quantum for a user is
                             <1>  ;  zero (see 'sysret'). It then restores the user's registers and
                             <1>  ; turns off the system flag. It then checked to see if there is
                             <1>  ; an interrupt from the user by calling 'isintr'. If there is,
                             <1>  ; the output gets flashed (see isintr) and interrupt action is
                             <1>  ; taken by a branch to 'intract'. If there is no interrupt from
                             <1>  ; the user, a rti is made.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     Fall through a 'bne' in 'sysret' & ?
                             <1>  ; Arguments:
                             <1>  ;     -
                             <1>  ; ............................................................
                             <1>  ;
                             <1>  ; 23/02/2014 (swapret)
                             <1>  ; 22/09/2013
                             <1> sysrel0: ;1:
000040EF 803D[BC740000]00    <1>  cmp   byte [u.quant], 0 ; 16/05/2013
                             <1>        ; tstb uquant / is the time quantum 0?
000040F6 7705                <1>         ja      short swapret
                             <1>        ; bne 1f / no, don't swap it out
                             <1> sysrelease: ; 07/12/2013 (jump from 'clock')
000040F8 E893120000          <1>  call  tswap
                             <1>        ; jsr r0,tswap / yes, swap it out
                             <1> ;
                             <1> ; Retro Unix 8086 v1 feature: return from 'swap' to 'swapret' address.
                             <1> swapret: ;1:
                             <1>  ; 10/09/2015
                             <1>  ; 01/09/2015
                             <1>  ; 14/05/2015
                             <1>  ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit, pm modifications)
                             <1>  ; 26/05/2013 (Retro UNIX 8086 v1)
                             <1>  ; cli
                             <1>  ; 24/07/2015
                             <1>  ;
                             <1>  ;; 'esp' must be already equal to '[u.usp]' here !
                             <1>  ;; mov      esp, [u.usp]
                             <1>
                             <1>  ; 22/09/2013
000040FD E878140000          <1>  call  isintr
                             <1>  ; 20/10/2013
00004102 7405                <1>  jz    short sysrel1
00004104 E875000000          <1>  call  intract
                             <1>        ; jsr r0,isintr / is there an interrupt from the user
                             <1>        ;     br intract / yes, output gets flushed, take interrupt
                             <1>                       ; / action
                             <1> sysrel1:
00004109 FA                  <1>  cli ; 14/10/2015
0000410A FE0D[77740000]      <1>  dec   byte [sysflg]
                             <1>        ; decb sysflg / turn system flag off
00004110 A1[D3740000]        <1>  mov     eax, [u.pgdir]
00004115 0F22D8              <1>  mov   cr3, eax  ; 1st PDE points to Kernel Page Table 0 (1st 4 MB)
                             <1>                 ; (others are different than kernel page tables)
                             <1>  ; 10/09/2015
00004118 61                  <1>  popad ; edi, esi, ebp, temp (icrement esp by 4), ebx, edx, ecx, eax
                             <1>        ; mov (sp)+,sc / restore user registers
                             <1>        ; mov (sp)+,mq
                             <1>        ; mov (sp)+,ac
                             <1>        ; mov (sp)+,r5
                             <1>        ; mov (sp)+,r4
                             <1>        ; mov (sp)+,r3
                             <1>        ; mov (sp)+,r2
                             <1>  ;
00004119 A1[80740000]        <1>  mov   eax, [u.r0]  ; ((return value in EAX))
```

```
0000411E 0FA9                 <1>  pop   gs
00004120 0FA1                 <1>  pop   fs
00004122 07                   <1>  pop   es
00004123 1F                   <1>  pop   ds
00004124 CF                   <1>  iretd
                              <1>        ; rti / no, return from interrupt
                              <1>
                              <1> badsys:
                              <1>  ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; (Major Modification: 'core' dumping procedure in
                              <1>         ;       original UNIX v1 and Retro UNIX 8086 v1
                              <1>  ;     has been changed to print 'Invalid System Call !'
                              <1>  ;     message on the user's console tty.)
                              <1>  ; (EIP, EAX values will be shown on screen with error message)
                              <1>  ; (EIP = Return address just after the system call -INT 30h-)
                              <1>  ; (EAX = Function number)
                              <1>  ;
00004125 FE05[C8740000]       <1>  inc   byte [u.bsys]
                              <1>  ;
0000412B 8B1D[78740000]       <1>  mov   ebx, [u.sp] ; esp at the beginning of 'sysent'
00004131 8B03                 <1>  mov   eax, [ebx] ; EIP (return address, not 'INT 30h' address)
00004133 E8E7D7FFFF           <1>  call  dwordtohex
00004138 8915[A06D0000]       <1>  mov   [bsys_msg_eip], edx
0000413E A3[A46D0000]         <1>  mov   [bsys_msg_eip+4], eax
00004143 A1[80740000]         <1>  mov   eax, [u.r0]
00004148 E8D2D7FFFF           <1>  call  dwordtohex
0000414D 8915[906D0000]       <1>  mov   [bsys_msg_eax], edx
00004153 A3[946D0000]         <1>  mov   [bsys_msg_eax+4], eax
00004158 31C0                 <1>  xor   eax, eax
0000415A C705[A0740000]-      <1>         mov     dword [u.base], badsys_msg ; "Invalid System call !"
00004160 [716D0000]           <1>
00004164 8B1D[90740000]       <1>  mov   ebx, [u.fofp]
0000416A 8903                 <1>  mov   [ebx], eax
                              <1>  ;mov  eax, 1 ; inode number of console tty (for user)
0000416C 40                   <1>  inc   eax
0000416D C705[A4740000]3B00- <1>  mov   dword [u.count], BSYS_M_SIZE
00004175 0000                 <1>
                              <1>        ; writei
                              <1>        ; INPUTS ->
                              <1>        ;    r1 - inode number
                              <1>        ;    u.count - byte count to be written
                              <1>        ;    u.base - points to user buffer
                              <1>        ;    u.fofp - points to word with current file offset
                              <1>        ; OUTPUTS ->
                              <1>        ;    u.count - cleared
                              <1>        ;    u.nread - accumulates total bytes passed back
                              <1>        ;
                              <1>        ; ((Modified registers: EDX, EBX, ECX, ESI, EDI, EBP))
00004177 E874190000           <1>  call  writei
                              <1>  ;mov  eax, 1
0000417C EB17                 <1>  jmp   sysexit
                              <1>
                              <1>        ; incb u.bsys / turn on the user's bad-system flag
                              <1>        ; mov $3f,u.namep / point u.namep to "core\0\0"
                              <1>        ; jsr r0,namei / get the i-number for the core image file
                              <1>        ; br 1f / error
                              <1>        ; neg r1 / negate the i-number to open the core image file
                              <1>               ; / for writing
                              <1>        ; jsr r0,iopen / open the core image file
                              <1>        ; jsr r0,itrunc / free all associated blocks
                              <1>        ; br 2f
                              <1> ;1:
                              <1>        ; mov $17,r1 / put i-node mode (17) in r1
                              <1>        ; jsr r0,maknod / make an i-node
                              <1>        ; mov u.dirbuf,r1 / put i-node number in r1
                              <1> ;2:
                              <1>        ; mov $core,u.base / move address core to u.base
                              <1>        ; mov $ecore-core,u.count / put the byte count in u.count
                              <1>        ; mov $u.off,u.fofp / more user offset to u.fofp
                              <1>        ; clr u.off / clear user offset
                              <1>        ; jsr r0,writei / write out the core image to the user
                              <1>        ; mov $user,u.base / pt. u.base to user
                              <1>        ; mov $64.,u.count / u.count = 64
                              <1>        ; jsr r0,writei / write out all the user parameters
                              <1>        ; neg r1 / make i-number positive
                              <1>        ; jsr r0,iclose / close the core image file
                              <1>        ; br sysexit /
                              <1> ;3:
                              <1>        ; <core\0\0>
                              <1>
                              <1> intract: ; / interrupt action
                              <1>  ; 14/10/2015
                              <1>  ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 09/05/2013 - 07/12/2013 (Retro UNIX 8086 v1)
                              <1>  ;
                              <1>  ; Retro UNIX 8086 v1 modification !
                              <1>  ; (Process/task switching and quit routine by using
                              <1>  ; Retro UNIX 8086 v1 keyboard interrupt output.))
                              <1>  ;
                              <1>  ; input -> 'u.quit'  (also value of 'u.intr' > 0)
                              <1>  ; output -> If value of 'u.quit' = FFFFh ('ctrl+brk' sign)
                              <1>  ;           'intract' will jump to 'sysexit'.
```

```
                             <1>  ;         Intract will return to the caller
                             <1>  ;           if value of 'u.quit' <> FFFFh.
                             <1>  ; 14/10/2015
0000417E FB                  <1>  sti
                             <1>  ; 07/12/2013
0000417F 66FF05[C0740000]    <1>  inc   word [u.quit]
00004186 7408                <1>  jz    short intrct0 ; FFFFh -> 0
00004188 66FF0D[C0740000]    <1>  dec   word [u.quit]
                             <1>  ; 16/04/2015
0000418F C3                  <1>  retn
                             <1> intrct0:
00004190 58                  <1>  pop   eax ; call intract -> retn
                             <1>  ;
00004191 31C0                <1>  xor   eax, eax
00004193 FEC0                <1>  inc   al  ; mov ax, 1
                             <1> ;;;
                             <1>  ; UNIX v1 original 'intract' routine...
                             <1>  ; / interrupt action
                             <1>        ;cmp *(sp),$rti / are you in a clock interrupt?
                             <1>        ; bne 1f / no, 1f
                             <1>        ; cmp (sp)+,(sp)+ / pop clock pointer
                             <1>  ; 1: / now in user area
                             <1>        ; mov r1,-(sp) / save r1
                             <1>        ; mov u.ttyp,r1
                             <1>              ; / pointer to tty buffer in control-to r1
                             <1>        ; cmpb 6(r1),$177
                             <1>              ; / is the interrupt char equal to "del"
                             <1>        ; beq 1f / yes, 1f
                             <1>        ; clrb 6(r1)
                             <1>                ; / no, clear the byte
                             <1>              ; / (must be a quit character)
                             <1>        ; mov (sp)+,r1 / restore r1
                             <1>        ; clr u.quit / clear quit flag
                             <1>        ; bis $20,2(sp)
                             <1>              ; / set trace for quit (sets t bit of
                             <1>              ; / ps-trace trap)
                             <1>        ; rti   ;  / return from interrupt
                             <1>  ; 1: / interrupt char = del
                             <1>        ; clrb 6(r1) / clear the interrupt byte
                             <1>                 ; / in the buffer
                             <1>        ; mov (sp)+,r1 / restore r1
                             <1>        ; cmp u.intr,$core / should control be
                             <1>                    ; / transferred to loc core?
                             <1>        ; blo 1f
                             <1>        ; jmp *u.intr / user to do rti yes,
                             <1>                    ; / transfer to loc core
                             <1>  ; 1:
                             <1>        ; sys 1 / exit
                             <1>
                             <1> sysexit: ; <terminate process>
                             <1>  ; 01/09/2015
                             <1>  ; 31/08/2015
                             <1>  ; 14/05/2015
                             <1>  ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 19/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysexit' terminates a process. First each file that
                             <1>  ; the process has opened is closed by 'flose'. The process
                             <1>  ; status is then set to unused. The 'p.pid' table is then
                             <1>  ; searched to find children of the dying process. If any of
                             <1>  ; children are zombies (died by not waited for), they are
                             <1>  ; set free. The 'p.pid' table is then searched to find the
                             <1>  ; dying process's parent. When the parent is found, it is
                             <1>  ; checked to see if it is free or it is a zombie. If it is
                             <1>  ; one of these, the dying process just dies. If it is waiting
                             <1>  ; for a child process to die, it notified that it doesn't
                             <1>  ; have to wait anymore by setting it's status from 2 to 1
                             <1>  ; (waiting to active). It is awakened and put on runq by
                             <1>  ; 'putlu'. The dying process enters a zombie state in which
                             <1>  ; it will never be run again but stays around until a 'wait'
                             <1>  ; is completed by it's parent process. If the parent is not
                             <1>  ; found, process just dies. This means 'swap' is called with
                             <1>  ; 'u.uno=0'. What this does is the 'wswap' is not called
                             <1>  ; to write out the process and 'rswap' reads the new process
                             <1>  ; over the one that dies..i.e., the dying process is
                             <1>  ; overwritten and destroyed.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysexit or conditional branch.
                             <1>  ; Arguments:
                             <1>  ;     -
                             <1>  ; ..............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       System call number (=1) is in EAX register.
                             <1>  ;
                             <1>  ;       Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
                             <1>  ;       registers depending of function details.
                             <1>  ;
                             <1>  ; ('swap' procedure is mostly different than original UNIX v1.)
                             <1>  ;
                             <1> ; / terminate process
                             <1>  ; AX = 1
00004195 6648                <1>  dec   ax ; 0
00004197 66A3[BE740000]      <1>  mov   [u.intr], ax ; 0
                             <1>        ; clr u.intr / clear interrupt control word
                             <1>        ; clr r1 / clear r1
                             <1>  ; AX = 0
                             <1> sysexit_1: ; 1:
                             <1>  ; AX = File descriptor
                             <1>        ; / r1 has file descriptor (index to u.fp list)
                             <1>        ; / Search the whole list
0000419D E8070D0000          <1>  call  fclose
                             <1>        ; jsr r0,fclose / close all files the process opened
                             <1>  ;; ignore error return
                             <1>        ; br .+2 / ignore error return
                             <1>  ;inc  ax
000041A2 FEC0                <1>  inc   al
                             <1>        ; inc r1 / increment file descriptor
                             <1>  ;cmp  ax, 10
000041A4 3C0A                <1>  cmp   al, 10
                             <1>        ; cmp r1,$10. / end of u.fp list?
000041A6 72F5                <1>  jb    short sysexit_1
                             <1>        ; blt 1b / no, go back
000041A8 0FB61D[C9740000]    <1>  movzx ebx, byte [u.uno] ; 01/09/2015
                             <1>        ; movb      u.uno,r1 / yes, move dying process's number to r1
000041AF 88A3[05720000]      <1>  mov   [ebx+p.stat-1], ah ; 0, SFREE, 05/02/2014
                             <1>        ; clrb p.stat-1(r1) / free the process
                             <1>  ;shl  bx, 1
000041B5 D0E3                <1>  shl   bl, 1
                             <1>        ; asl r1 / use r1 for index into the below tables
000041B7 668B8B[74710000]    <1>  mov   cx, [ebx+p.pid-2]
                             <1>        ; mov p.pid-2(r1),r3 / move dying process's name to r3
000041BE 668B93[94710000]    <1>  mov   dx, [ebx+p.ppid-2]
                             <1>        ; mov p.ppid-2(r1),r4 / move its parents name to r4
                             <1>  ; xor       bx, bx ; 0
000041C5 30DB                <1>  xor   bl, bl ; 0
                             <1>        ; clr r2
000041C7 31F6                <1>  xor   esi, esi ; 0
                             <1>        ; clr r5 / initialize reg
                             <1> sysexit_2: ; 1:
                             <1>          ; / find children of this dying process,
                             <1>        ; / if they are zombies, free them
                             <1>  ;add  bx, 2
000041C9 80C302              <1>  add   bl, 2
                             <1>        ; add $2,r2 / search parent process table
                             <1>                ; / for dying process's name
000041CC 66398B[94710000]    <1>  cmp   [ebx+p.ppid-2], cx
                             <1>        ; cmp p.ppid-2(r2),r3 / found it?
000041D3 7513                <1>  jne   short sysexit_4
                             <1>        ; bne 3f / no
                             <1>  ;shr  bx, 1
000041D5 D0EB                <1>  shr   bl, 1
                             <1>        ; asr r2 / yes, it is a parent
000041D7 80BB[05720000]03    <1>  cmp   byte [ebx+p.stat-1], 3 ; SZOMB, 05/02/2014
                             <1>        ; cmpb p.stat-1(r2),$3 / is the child of this
                             <1>                       ; / dying process a zombie
000041DE 7506                <1>  jne   short sysexit_3
                             <1>        ; bne 2f / no
000041E0 88A3[05720000]      <1>  mov   [ebx+p.stat-1], ah ; 0, SFREE, 05/02/2014
                             <1>        ; clrb p.stat-1(r2) / yes, free the child process
                             <1> sysexit_3: ; 2:
                             <1>  ;shr  bx, 1
000041E6 D0E3                <1>  shl   bl, 1
                             <1>        ; asl r2
                             <1> sysexit_4: ; 3:
                             <1>        ; / search the process name table
                             <1>        ; / for the dying process's parent
000041E8 663993[74710000]    <1>  cmp   [ebx+p.pid-2], dx ; 17/09/2013
                             <1>        ; cmp p.pid-2(r2),r4 / found it?
000041EF 7502                <1>  jne   short sysexit_5
                             <1>        ; bne 3f / no
000041F1 89DE                <1>  mov   esi, ebx
                             <1>        ; mov r2,r5 / yes, put index to p.pid table (parents
                             <1>                ; / process # x2) in r5
                             <1> sysexit_5: ; 3:
                             <1>  ;cmp  bx, nproc + nproc
000041F3 80FB20              <1>  cmp   bl, nproc + nproc
                             <1>        ; cmp r2,$nproc+nproc / has whole table been searched?
000041F6 72D1                <1>  jb    short sysexit_2
                             <1>        ; blt 1b / no, go back
                             <1>        ; mov r5,r1 / yes, r1 now has parents process # x2
000041F8 21F6                <1>  and   esi, esi ; r5=r1
000041FA 7431                <1>  jz    short sysexit_6
                             <1>        ; beq 2f / no parent has been found.
                             <1>             ; / The process just dies
000041FC 66D1EE              <1>  shr   si, 1
                             <1>        ; asr r1 / set up index to p.stat
000041FF 8A86[05720000]      <1>  mov   al, [esi+p.stat-1]
                             <1>        ; movb p.stat-1(r1),r2 / move status of parent to r2
00004205 20C0                <1>  and   al, al
00004207 7424                <1>  jz    short sysexit_6
                             <1>        ; beq 2f / if its been freed, 2f
00004209 3C03                <1>  cmp   al, 3
```

```
                             <1>        ; cmp r2,$3 / is parent a zombie?
0000420B 7420                <1>  je    short sysexit_6
                             <1>        ; beq 2f / yes, 2f
                             <1>  ; BH = 0
0000420D 8A1D[C9740000]      <1>  mov   bl, [u.uno]
                             <1>        ; movb u.uno,r3 / move dying process's number to r3
00004213 C683[05720000]03    <1>  mov   byte [ebx+p.stat-1], 3  ; SZOMB, 05/02/2014
                             <1>        ; movb $3,p.stat-1(r3) / make the process a zombie
                             <1>  ; 05/02/2014
0000421A 3C01                <1>  cmp   al, 1 ; SRUN
0000421C 740F                <1>  je    short sysexit_6
                             <1>  ;cmp  al, 2
                             <1>        ; cmp r2,$2 / is the parent waiting for
                             <1>                ; / this child to die
                             <1>  ;jne  short sysexit_6
                             <1>        ; bne 2f / yes, notify parent not to wait any more
                             <1>  ; 05/02/2014
                             <1>  ; p.stat = 2 --> waiting
                             <1>  ; p.stat = 4 --> sleeping
0000421E C686[05720000]01    <1>  mov   byte [esi+p.stat-1], 1 ; SRUN ; 05/02/2014
                             <1>  ;dec  byte [esi+p.stat-1]
                             <1>        ; decb      p.stat-1(r1) / awaken it by putting it (parent)
00004225 6689F0              <1>  mov   ax, si ; r1  (process number in AL)
                             <1>  ;
                             <1>  ;mov  ebx, runq + 4
                             <1>        ; mov $runq+4,r2 / on the runq
00004228 E83B120000          <1>  call  putlu
                             <1>        ; jsr r0, putlu
                             <1> sysexit_6: ; 2:
                             <1>  ; 31/08/2015
                             <1>        ; / the process dies
0000422D C605[C9740000]00    <1>  mov   byte [u.uno], 0
                             <1>        ; clrb u.uno / put zero as the process number,
                             <1>              ; / so "swap" will
00004234 E861110000          <1>  call  swap
                             <1>        ; jsr r0,swap / overwrite process with another process
                             <1> hlt_sys:
                             <1>  ;sti ; 18/01/2014
                             <1> hlts0:
00004239 F4                  <1>  hlt
0000423A EBFD                <1>  jmp   short hlts0
                             <1>        ; 0 / and thereby kill it; halt?
                             <1>
                             <1>
                             <1> syswait: ; < wait for a processs to die >
                             <1>  ; 17/09/2015
                             <1>  ; 02/09/2015
                             <1>  ; 01/09/2015
                             <1>  ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 24/05/2013 - 05/02/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'syswait' waits for a process die.
                             <1>  ; It works in following way:
                             <1>  ;    1) From the parent process number, the parent's
                             <1>  ;     process name is found. The p.ppid table of parent
                             <1>  ;     names is then searched for this process name.
                             <1>  ;     If a match occurs, r2 contains child's process
                             <1>  ;     number. The child status is checked to see if it is
                             <1>  ;     a zombie, i.e; dead but not waited for (p.stat=3)
                             <1>  ;     If it is, the child process is freed and it's name
                             <1>  ;     is put in (u.r0). A return is then made via 'sysret'.
                             <1>  ;     If the child is not a zombie, nothing happens and
                             <1>  ;     the search goes on through the p.ppid table until
                             <1>  ;     all processes are checked or a zombie is found.
                             <1>  ;    2) If no zombies are found, a check is made to see if
                             <1>  ;     there are any children at all. If there are none,
                             <1>  ;     an error return is made. If there are, the parent's
                             <1>  ;     status is set to 2 (waiting for child to die),
                             <1>  ;     the parent is swapped out, and a branch to 'syswait'
                             <1>  ;     is made to wait on the next process.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     ?
                             <1>  ; Arguments:
                             <1>  ;     -
                             <1>  ; Inputs: -
                             <1>  ; Outputs: if zombie found, it's name put in u.r0.
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>
                             <1> ; / wait for a process to die
                             <1>
                             <1> syswait_0:
0000423C 0FB61D[C9740000]    <1>  movzx ebx, byte [u.uno] ; 01/09/2015
                             <1>        ; movb u.uno,r1 / put parents process number in r1
00004243 D0E3                <1>  shl   bl, 1
                             <1>  ;shl  bx, 1
                             <1>        ; asl r1 / x2 to get index into p.pid table
00004245 668B83[74710000]    <1>  mov   ax, [ebx+p.pid-2]
                             <1>        ; mov p.pid-2(r1),r1 / get the name of this process
0000424C 31F6                <1>  xor   esi, esi
                             <1>        ; clr r2
```

```
0000424E 31C9                <1>  xor   ecx, ecx ; 30/10/2013
                             <1>  ;xor  cl, cl
                             <1>        ; clr r3 / initialize reg 3
                             <1> syswait_1: ; 1:
00004250 6683C602            <1>  add   si, 2
                             <1>        ; add $2,r2 / use r2 for index into p.ppid table
                             <1>                ; / search table of parent processes
                             <1>                ; / for this process name
00004254 663B86[94710000]    <1>  cmp   ax, [esi+p.ppid-2]
                             <1>        ; cmp p.ppid-2(r2),r1 / r2 will contain the childs
                             <1>                          ; / process number
0000425B 7535                <1>  jne   short syswait_3
                             <1>        ;bne 3f / branch if no match of parent process name
                             <1>  ;inc  cx
0000425D FEC1                <1>  inc   cl
                             <1>        ;inc r3 / yes, a match, r3 indicates number of children
0000425F 66D1EE              <1>  shr   si, 1
                             <1>        ; asr r2 / r2/2 to get index to p.stat table
                             <1>  ; The possible states ('p.stat' values) of a process are:
                             <1>  ;     0 = free or unused
                             <1>  ;     1 = active
                             <1>  ;     2 = waiting for a child process to die
                             <1>  ;     3 = terminated, but not yet waited for (zombie).
00004262 80BE[05720000]03    <1>  cmp   byte [esi+p.stat-1], 3 ; SZOMB, 05/02/2014
                             <1>        ; cmpb p.stat-1(r2),$3 / is the child process a zombie?
00004269 7524                <1>  jne   short syswait_2
                             <1>        ; bne 2f / no, skip it
0000426B 88BE[05720000]      <1>  mov   [esi+p.stat-1], bh ; 0
                             <1>        ; clrb p.stat-1(r2) / yes, free it
00004271 66D1E6              <1>  shl   si, 1
                             <1>        ; asl r2 / r2x2 to get index into p.pid table
00004274 0FB786[74710000]    <1>  movzx eax, word [esi+p.pid-2]
0000427B A3[80740000]        <1>  mov   [u.r0], eax
                             <1>        ; mov p.pid-2(r2),*u.r0
                             <1>                    ; / put childs process name in (u.r0)
                             <1>  ;
                             <1>  ; Retro UNIX 386 v1 modification ! (17/09/2015)
                             <1>  ;
                             <1>  ; Parent process ID -p.ppid- field (of the child process)
                             <1>  ; must be cleared in order to prevent infinitive 'syswait'
                             <1>  ; system call loop from the application/program if it calls
                             <1>  ; 'syswait' again (mistakenly) while there is not a zombie
                             <1>  ; or running child process to wait. ('forktest.s', 17/09/2015)
                             <1>  ;
                             <1>  ; Note: syswait will return with error if there is not a
                             <1>  ;       zombie or running process to wait.
                             <1>  ;
00004280 6629C0              <1>  sub   ax, ax
00004283 668986[94710000]    <1>  mov   [esi+p.ppid-2], ax ; 0 ; 17/09/2015
0000428A E910FEFFFF          <1>  jmp   sysret0 ; ax = 0
                             <1>  ;
                             <1>  ;jmp  sysret
                             <1>        ; br sysret1 / return cause child is dead
                             <1> syswait_2: ; 2:
0000428F 66D1E6              <1>  shl   si, 1
                             <1>        ; asl r2 / r2x2 to get index into p.ppid table
                             <1> syswait_3: ; 3:
00004292 6683FE20            <1>  cmp   si, nproc+nproc
                             <1>        ; cmp r2,$nproc+nproc / have all processes been checked?
00004296 72B8                <1>  jb    short syswait_1
                             <1>        ; blt 1b / no, continue search
                             <1>  ;and  cx, cx
00004298 20C9                <1>  and   cl, cl
                             <1>        ; tst r3 / one gets here if there are no children
                             <1>               ; / or children that are still active
                             <1>  ; 30/10/2013
0000429A 750B                <1>  jnz   short syswait_4
                             <1>  ;jz   error
                             <1>        ; beq error1 / there are no children, error
0000429C 890D[80740000]      <1>  mov   [u.r0], ecx ; 0
000042A2 E9D5FDFFFF          <1>  jmp   error
                             <1> syswait_4:
000042A7 8A1D[C9740000]      <1>  mov   bl, [u.uno]
                             <1>        ; movb u.uno,r1 / there are children so put
                             <1>                    ; / parent process number in r1
000042AD FE83[05720000]      <1>  inc   byte [ebx+p.stat-1] ; 2, SWAIT, 05/02/2014
                             <1>        ; incb p.stat-1(r1) / it is waiting for
                             <1>                      ; / other children to die
                             <1>  ; 04/11/2013
000042B3 E8E2100000          <1>  call  swap
                             <1>        ; jsr r0,swap / swap it out, because it's waiting
000042B8 EB82                <1>  jmp   syswait_0
                             <1>        ; br syswait / wait on next process
                             <1>
                             <1> sysfork: ; < create a new process >
                             <1>  ; 18/09/2015
                             <1>  ; 04/09/2015
                             <1>  ; 02/09/2015
                             <1>  ; 01/09/2015
                             <1>  ; 28/08/2015
                             <1>  ; 14/05/2015
                             <1>  ; 10/05/2015
```

```
<1>  ; 09/05/2015
<1>  ; 06/05/2015 (Retro UNIX 386 v1 - Beginning)
<1>  ; 24/05/2013 - 14/02/2014 (Retro UNIX 8086 v1)
<1>  ;
<1>  ; 'sysfork' creates a new process. This process is referred
<1>  ; to as the child process. This new process core image is
<1>  ; a copy of that of the caller of 'sysfork'. The only
<1>  ; distinction is the return location and the fact that (u.r0)
<1>  ; in the old process (parent) contains the process id (p.pid)
<1>  ; of the new process (child). This id is used by 'syswait'.
<1>  ; 'sysfork' works in the following manner:
<1>  ;    1) The process status table (p.stat) is searched to find
<1>  ;     a process number that is unused. If none are found
<1>  ;     an error occurs.
<1>  ;    2) when one is found, it becomes the child process number
<1>  ;     and it's status (p.stat) is set to active.
<1>  ;    3) If the parent had a control tty, the interrupt
<1>  ;     character in that tty buffer is cleared.
<1>  ;    4) The child process is put on the lowest priority run
<1>  ;     queue via 'putlu'.
<1>  ;    5) A new process name is gotten from 'mpid' (actually
<1>  ;     it is a unique number) and is put in the child's unique
<1>  ;     identifier; process id (p.pid).
<1>  ;    6) The process name of the parent is then obtained and
<1>  ;     placed in the unique identifier of the parent process
<1>  ;     name is then put in 'u.r0'.
<1>  ;    7) The child process is then written out on disk by
<1>  ;     'wswap',i.e., the parent process is copied onto disk
<1>  ;     and the child is born. (The child process is written
<1>  ;     out on disk/drum with 'u.uno' being the child process
<1>  ;     number.)
<1>  ;    8) The parent process number is then restored to 'u.uno'.
<1>  ;    9) The child process name is put in 'u.r0'.
<1>  ;   10) The pc on the stack sp + 18 is incremented by 2 to
<1>  ;     create the return address for the parent process.
<1>  ;   11) The 'u.fp' list as then searched to see what files
<1>  ;     the parent has opened. For each file the parent has
<1>  ;     opened, the corresponding 'fsp' entry must be updated
<1>  ;     to indicate that the child process also has opened
<1>  ;     the file. A branch to 'sysret' is then made.

<1>  ;
<1>  ; Calling sequence:
<1>  ;     from shell ?
<1>  ; Arguments:
<1>  ;     -
<1>  ; Inputs: -
<1>  ; Outputs: *u.r0 - child process name
<1>  ; ...............................................................
<1>  ;
<1>  ; Retro UNIX 8086 v1 modification:
<1>  ;     AX = r0 = PID (>0) (at the return of 'sysfork')
<1>  ;     = process id of child a parent process returns
<1>  ;     = process id of parent when a child process returns
<1>  ;
<1>  ;       In original UNIX v1, sysfork is called and returns as
<1>  ;     in following manner: (with an example: c library, fork)
<1>  ;
<1>  ;     1:
<1>  ;           sys   fork
<1>  ;                 br 1f  / child process returns here
<1>  ;           bes   2f     / parent process returns here
<1>  ;           / pid of new process in r0
<1>  ;           rts   pc
<1>  ;     2: / parent process condionally branches here
<1>  ;           mov   $-1,r0 / pid = -1 means error return
<1>  ;           rts   pc
<1>  ;
<1>  ;     1: / child process brances here
<1>  ;           clr   r0   / pid = 0 in child process
<1>  ;           rts   pc
<1>  ;
<1>  ;     In UNIX v7x86 (386) by Robert Nordier (1999)
<1>  ;           // pid = fork();
<1>  ;           //
<1>  ;           // pid == 0 in child process;
<1>  ;           // pid == -1 means error return
<1>  ;           // in child,
<1>  ;           //    parents id is in par_uid if needed
<1>  ;
<1>  ;           _fork:
<1>  ;                 mov   $.fork,eax
<1>  ;                 int   $0x30
<1>  ;                 jmp   1f
<1>  ;                 jnc   2f
<1>  ;                 jmp   cerror
<1>  ;           1:
<1>  ;                 mov   eax,_par_uid
<1>  ;                 xor   eax,eax
<1>  ;           2:
<1>  ;                 ret
<1>  ;
```

```
                              <1>  ;     In Retro UNIX 8086 v1,
                              <1>  ;     'sysfork' returns in following manner:
                              <1>  ;
                              <1>  ;           mov   ax, sys_fork
                              <1>  ;           mov   bx, offset @f ; routine for child
                              <1>  ;           int   20h
                              <1>  ;           jc    error
                              <1>  ;
                              <1>  ;     ; Routine for parent process here (just after 'jc')
                              <1>  ;           mov   word ptr [pid_of_child], ax
                              <1>  ;           jmp   next_routine_for_parent
                              <1>  ;
                              <1>  ;     @@: ; routine for child process here
                              <1>  ;           ....
                              <1>  ;     NOTE: 'sysfork' returns to specified offset
                              <1>  ;            for child process by using BX input.
                              <1>  ;           (at first, parent process will return then
                              <1>  ;           child process will return -after swapped in-
                              <1>  ;           'syswait' is needed in parent process
                              <1>  ;           if return from child process will be waited for.)
                              <1>  ;
                              <1>
                              <1> ; / create a new process
                              <1>  ; EBX = return address for child process
                              <1>       ; (Retro UNIX 8086 v1 modification !)
000042BA 31F6                 <1>  xor   esi, esi
                              <1>        ; clr r1
                              <1> sysfork_1: ; 1: / search p.stat table for unused process number
000042BC 46                   <1>  inc   esi
                              <1>        ; inc r1
000042BD 80BE[05720000]00     <1>  cmp   byte [esi+p.stat-1], 0 ; SFREE, 05/02/2014
                              <1>        ; tstb p.stat-1(r1) / is process active, unused, dead
000042C4 760B                 <1>  jna   short sysfork_2
                              <1>        ; beq 1f / it's unused so branch
000042C6 6683FE10             <1>  cmp   si, nproc
                              <1>        ; cmp r1,$nproc / all processes checked
000042CA 72F0                 <1>  jb    short sysfork_1
                              <1>        ; blt 1b / no, branch back
                              <1>  ;
                              <1>  ; Retro UNIX 8086 v1. modification:
                              <1>  ;     Parent process returns from 'sysfork' to address
                              <1>  ;     which is just after 'sysfork' system call in parent
                              <1>  ;     process. Child process returns to address which is put
                              <1>  ;     in BX register by parent process for 'sysfork'.
                              <1>  ;
                              <1>        ;add $2,18.(sp) / add 2 to pc when trap occured, points
                              <1>                      ; / to old process return
                              <1>        ; br error1 / no room for a new process
000042CC E9ABFDFFFF           <1>  jmp   error
                              <1> sysfork_2: ; 1:
000042D1 E861EDFFFF           <1>  call  allocate_page
000042D6 0F82A0FDFFFF         <1>  jc    error
000042DC 50                   <1>  push  eax   ; UPAGE (user structure page) address
                              <1>  ; Retro UNIX 386 v1 modification!
000042DD E85EEFFFFF           <1>  call  duplicate_page_dir
                              <1>        ; EAX = New page directory
000042E2 730B                 <1>  jnc   short sysfork_3
000042E4 58                   <1>  pop   eax   ; UPAGE (user structure page) address
000042E5 E825EFFFFF           <1>  call  deallocate_page
000042EA E98DFDFFFF           <1>  jmp   error
                              <1> sysfork_3:
                              <1>  ; Retro UNIX 386 v1 modification !
000042EF 56                   <1>  push  esi
000042F0 E81D110000           <1>  call  wswap ; save current user (u) structure, user registers
                              <1>              ; and interrupt return components (for IRET)
000042F5 8705[D3740000]       <1>  xchg  eax, [u.pgdir] ; page directory of the child process
000042FB A3[D7740000]         <1>  mov   [u.ppgdir], eax ; page directory of the parent process
00004300 5E                   <1>  pop   esi
00004301 58                   <1>  pop   eax   ; UPAGE (user structure page) address
                              <1>        ; [u.usp] = esp
00004302 89F7                 <1>  mov   edi, esi
00004304 66C1E702             <1>  shl   di, 2
00004308 8987[12720000]       <1>  mov   [edi+p.upage-4], eax ; memory page for 'user' struct
0000430E A3[CA740000]         <1>  mov   [u.upage], eax ; memory page for 'user' struct (child)
                              <1>  ; 28/08/2015
00004313 0FB605[C9740000]     <1>  movzx eax, byte [u.uno] ; parent process number
                              <1>        ; movb u.uno,-(sp) / save parent process number
0000431A 89C7                 <1>  mov   edi, eax
0000431C 50                   <1>         push eax ; **
0000431D 8A87[D5710000]       <1>  mov     al, [edi+p.ttyc-1] ; console tty (parent)
                              <1>  ; 18/09/2015
                              <1>  ;mov     [esi+p.ttyc-1], al ; set child's console tty
                              <1>  ;mov     [esi+p.waitc-1], ah ; 0 ; reset child's wait channel
00004323 668986[D5710000]     <1>  mov     [esi+p.ttyc-1], ax ; al - set child's console tty
                              <1>                        ; ah - reset child's wait channel
0000432A 89F0                 <1>  mov   eax, esi
0000432C A2[C9740000]         <1>  mov   [u.uno], al ; child process number
                              <1>        ;movb r1,u.uno / set child process number to r1
00004331 FE86[05720000]       <1>         inc     byte [esi+p.stat-1] ; 1, SRUN, 05/02/2014
                              <1>        ; incb p.stat-1(r1) / set p.stat entry for child
                              <1>                    ; / process to active status
                              <1>        ; mov u.ttyp,r2 / put pointer to parent process'
                              <1>                    ; / control tty buffer in r2
                              <1>                 ; beq 2f / branch, if no such tty assigned
                              <1>        ; clrb 6(r2) / clear interrupt character in tty buffer
                              <1>  ; 2:
00004337 53                   <1>  push  ebx  ; * return address for the child process
                              <1>             ; * Retro UNIX 8086 v1 feature only !
                              <1>  ; (Retro UNIX 8086 v1 modification!)
                              <1>        ; mov $runq+4,r2
00004338 E82B110000           <1>  call  putlu
                              <1>        ; jsr r0,putlu / put child process on lowest priority
                              <1>                 ; / run queue
0000433D 66D1E6               <1>  shl   si, 1
                              <1>        ; asl r1 / multiply r1 by 2 to get index
                              <1>              ; / into p.pid table
00004340 66FF05[6E740000]     <1>  inc   word [mpid]
                              <1>        ; inc mpid / increment m.pid; get a new process name
00004347 66A1[6E740000]       <1>  mov   ax, [mpid]
0000434D 668986[74710000]     <1>  mov   [esi+p.pid-2], ax
                              <1>        ;mov mpid,p.pid-2(r1) / put new process name
                              <1>                        ; / in child process' name slot
00004354 5A                   <1>  pop   edx  ; * return address for the child process
                              <1>             ; * Retro UNIX 8086 v1 feature only !
00004355 5B                   <1>        pop   ebx  ; **
                              <1>  ;mov  ebx, [esp] ; ** parent process number
                              <1>        ; movb (sp),r2 / put parent process number in r2
00004356 66D1E3               <1>  shl   bx, 1
                              <1>        ;asl r2 / multiply by 2 to get index into below tables
                              <1>  ;movzx eax, word [ebx+p.pid-2]
00004359 668B83[74710000]     <1>  mov   ax, [ebx+p.pid-2]
                              <1>        ; mov p.pid-2(r2),r2 / get process name of parent
                              <1>                       ; / process
00004360 668986[94710000]     <1>  mov   [esi+p.ppid-2], ax
                              <1>        ; mov r2,p.ppid-2(r1) / put parent process name
                              <1>               ; / in parent process slot for child
00004367 A3[80740000]         <1>  mov   [u.r0], eax
                              <1>        ; mov r2,*u.r0 / put parent process name on stack
                              <1>                  ; / at location where r0 was saved
0000436C 8B2D[78740000]       <1>  mov   ebp, [u.sp] ; points to return address (EIP for IRET)
00004372 895500               <1>  mov   [ebp], edx ; *, CS:EIP -> EIP
                              <1>                 ; * return address for the child process
                              <1>        ; mov $sysret1,-(sp) /
                              <1>        ; mov sp,u.usp / contents of sp at the time when
                              <1>                   ; / user is swapped out
                              <1>        ; mov $sstack,sp / point sp to swapping stack space
                              <1>  ; 04/09/2015 - 01/09/2015
                              <1>  ; [u.usp] = esp
00004375 68[9C400000]         <1>  push  sysret ; ***
0000437A 8925[7C740000]       <1>  mov   [u.usp], esp ; points to 'sysret' address (***)
                              <1>                  ; (for child process)
00004380 31C0                 <1>  xor   eax, eax
00004382 66A3[B0740000]       <1>  mov   [u.ttyp], ax ; 0
                              <1>  ;
00004388 E885100000           <1>  call  wswap ; Retro UNIX 8086 v1 modification !
                              <1>        ;jsr r0,wswap / put child process out on drum
                              <1>        ;jsr r0,unpack / unpack user stack
                              <1>        ;mov u.usp,sp / restore user stack pointer
                              <1>        ; tst (sp)+ / bump stack pointer
                              <1>  ; Retro UNIX 386 v1 modification !
0000438D 58                   <1>  pop   eax ; ***
0000438E 66D1E3               <1>  shl   bx, 1
00004391 8B83[12720000]       <1>  mov     eax, [ebx+p.upage-4] ; UPAGE address ; 14/05/2015
00004397 E89F100000           <1>  call  rswap ; restore parent process 'u' structure,
                              <1>              ; registers and return address (for IRET)
                              <1>        ;movb (sp)+,u.uno / put parent process number in u.uno
0000439C 0FB705[6E740000]     <1>         movzx   eax, word [mpid]
000043A3 A3[80740000]         <1>  mov   [u.r0], eax
                              <1>        ; mov mpid,*u.r0 / put child process name on stack
                              <1>                   ; / where r0 was saved
                              <1>        ; add $2,18.(sp) / add 2 to pc on stack; gives parent
                              <1>                      ; / process return
                              <1>  ;xor  ebx, ebx
000043A8 31F6                 <1>  xor     esi, esi
                              <1>        ;clr r1
                              <1> sysfork_4: ; 1: / search u.fp list to find the files
                              <1>        ; / opened by the parent process
                              <1>  ; 01/09/2015
                              <1>  ;xor  bh, bh
                              <1>  ;mov  bl, [esi+u.fp]
000043AA 8A86[86740000]       <1>  mov   al, [esi+u.fp]
                              <1>        ; movb u.fp(r1),r2 / get an open file for this process
                              <1>         ;or      bl, bl
000043B0 08C0                 <1>  or    al, al
000043B2 740D                 <1>  jz    short sysfork_5
                              <1>        ; beq 2f / file has not been opened by parent,
                              <1>              ; / so branch
000043B4 B40A                 <1>  mov   ah, 10 ; Retro UNIX 386 v1 fsp structure size = 10 bytes
000043B6 F6E4                 <1>  mul   ah
                              <1>  ;movzx      ebx, ax
000043B8 6689C3               <1>  mov   bx, ax
                              <1>  ;shl     bx, 3
                              <1>        ; asl r2 / multiply by 8
                              <1>            ; asl r2 / to get index into fsp table
                              <1>              ; asl r2
000043BB FE83[54720000]       <1>        inc     byte [ebx+fsp-2]
                              <1>        ; incb fsp-2(r2) / increment number of processes
                              <1>                   ; / using file, because child will now be
                              <1>                   ; / using this file
                              <1> sysfork_5: ; 2:
000043C1 46                   <1>         inc     esi
                              <1>        ; inc r1 / get next open file
000043C2 6683FE0A             <1>         cmp     si, 10
                              <1>        ; cmp r1,$10. / 10. files is the maximum number which
                              <1>                ; / can be opened
000043C6 72E2                 <1>  jb    short sysfork_4
                              <1>        ; blt 1b / check next entry
000043C8 E9CFFCFFFF           <1>  jmp   sysret
                              <1>        ; br sysret1
                              <1>
                              <1> sysread: ; < read from file >
                              <1>  ; 13/05/2015
                              <1>  ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 23/05/2013 (Retro UNIX 8086 v1)
                              <1>  ;
                              <1>  ; 'sysread' is given a buffer to read into and the number of
                              <1>  ; characters to be read. If finds the file from the file
                              <1>  ; descriptor located in *u.r0 (r0). This file descriptor
                              <1>  ; is returned from a successful open call (sysopen).
                              <1>  ; The i-number of file is obtained via 'rw1' and the data
                              <1>  ; is read into core via 'readi'.
                              <1>  ;
                              <1>  ; Calling sequence:
                              <1>  ;     sysread; buffer; nchars
                              <1>  ; Arguments:
                              <1>  ;     buffer - location of contiguous bytes where
                              <1>  ;              input will be placed.
                              <1>  ;     nchars - number of bytes or characters to be read.
                              <1>  ; Inputs: *u.r0 - file descriptor (& arguments)
                              <1>  ; Outputs: *u.r0 - number of bytes read.
                              <1>  ; .............................................................
                              <1>  ;
                              <1>  ; Retro UNIX 8086 v1 modification:
                              <1>  ;       'sysread' system call has three arguments; so,
                              <1>  ;     * 1st argument, file descriptor is in BX register
                              <1>  ;     * 2nd argument, buffer address/offset in CX register
                              <1>  ;     * 3rd argument, number of bytes is in DX register
                              <1>  ;
                              <1>  ;     AX register (will be restored via 'u.r0') will return
                              <1>  ;     to the user with number of bytes read.
                              <1>  ;
000043CD E83D000000           <1>  call  rw1
000043D2 0F82A4FCFFFF         <1>  jc    error ; 13/05/2015, ax < 1
                              <1>        ; jsr r0,rw1 / get i-number of file to be read into r1
000043D8 F6C480               <1>  test  ah, 80h
                              <1>        ; tst r1 / negative i-number?
000043DB 0F859BFCFFFF         <1>  jnz   error
                              <1>        ; ble error1 / yes, error 1 to read
                              <1>                ; / it should be positive
000043E1 E814150000           <1>  call  readi
                              <1>        ; jsr r0,readi / read data into core
000043E6 EB18                 <1>  jmp   short rw0
                              <1>        ; br 1f
                              <1> syswrite: ; < write to file >
                              <1>  ; 13/05/2015
                              <1>  ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 23/05/2013 (Retro UNIX 8086 v1)
                              <1>  ;
                              <1>  ; 'syswrite' is given a buffer to write onto an output file
                              <1>  ; and the number of characters to write. If finds the file
                              <1>  ; from the file descriptor located in *u.r0 (r0). This file
                              <1>  ; descriptor is returned from a successful open or create call
                              <1>  ; (sysopen or syscreat). The i-number of file is obtained via
                              <1>  ; 'rw1' and buffer is written on the output file via 'write'.
                              <1>  ;
                              <1>  ; Calling sequence:
                              <1>  ;     syswrite; buffer; nchars
                              <1>  ; Arguments:
                              <1>  ;     buffer - location of contiguous bytes to be writtten.
                              <1>  ;     nchars - number of characters to be written.
                              <1>  ; Inputs: *u.r0 - file descriptor (& arguments)
                              <1>  ; Outputs: *u.r0 - number of bytes written.
                              <1>  ; .............................................................
                              <1>  ;
                              <1>  ; Retro UNIX 8086 v1 modification:
                              <1>  ;       'syswrite' system call has three arguments; so,
                              <1>  ;     * 1st argument, file descriptor is in BX register
                              <1>  ;     * 2nd argument, buffer address/offset in CX register
                              <1>  ;     * 3rd argument, number of bytes is in DX register
                              <1>  ;
                              <1>  ;     AX register (will be restored via 'u.r0') will return
                              <1>  ;     to the user with number of bytes written.
                              <1>  ;
000043E8 E822000000           <1>  call  rw1
000043ED 0F8289FCFFFF         <1>  jc    error ; 13/05/2015, ax < 1
                              <1>        ; jsr r0,rw1 / get i-number in r1 of file to write
```

```
000043F3 F6C480              <1>         test ah, 80h
                             <1>        ; tst r1 / positive i-number ?
000043F6 744E                <1>         jz   short rw3 ; 13/05/2015
                             <1>  ;jz   error
                             <1>        ; bge error1 / yes, error 1
                             <1>                ; / negative i-number means write
000043F8 66F7D8              <1>         neg  ax
                             <1>        ; neg r1 / make it positive
000043FB E8F0160000          <1>  call  writei
                             <1>              ; jsr r0,writei / write data
                             <1> rw0: ; 1:
00004400 A1[A8740000]        <1>         mov  eax, [u.nread]
00004405 A3[80740000]        <1>  mov   [u.r0], eax
                             <1>        ; mov u.nread,*u.r0 / put no. of bytes transferred
                             <1>                     ; / into (u.r0)
0000440A E98DFCFFFF          <1>  jmp   sysret
                             <1>              ; br sysret1
                             <1> rw1:
                             <1>  ; 14/05/2015
                             <1>  ; 13/05/2015
                             <1>  ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 23/05/2013 - 24/05/2013 (Retro UNIX 8086 v1)
                             <1>  ; System call registers: bx, cx, dx (through 'sysenter')
                             <1>  ;
                             <1>  ;mov  [u.base], ecx     ; buffer address/offset
                             <1>                   ;(in the user's virtual memory space)
                             <1>  ;mov  [u.count], edx
                             <1>        ; jsr r0,arg; u.base / get buffer pointer
                             <1>                ; jsr r0,arg; u.count / get no. of characters
                             <1>  ;;mov eax, ebx ; file descriptor
                             <1>        ; mov *u.r0,r1 / put file descriptor
                             <1>                    ; / (index to u.fp table) in r1
                             <1>  ; 13/05/2015
0000440F C705[80740000]0000- <1>  mov   dword [u.r0], 0 ; r/w transfer count = 0 (reset)
00004417 0000                <1>
                             <1>  ;
                             <1>  ;; call     getf
                             <1>         ; eBX = File descriptor
00004419 E8D60A0000          <1>  call  getf1 ; calling point in 'getf' from 'rw1'
                             <1>        ; jsr r0,getf / get i-number of the file in r1
                             <1>  ; AX = I-number of the file ; negative i-number means write
                             <1>  ; 13/05/2015
0000441E 6683F801            <1>  cmp   ax, 1
00004422 7217                <1>  jb    short rw2
                             <1>  ;
00004424 890D[A0740000]      <1>  mov   [u.base], ecx     ; buffer address/offset
                             <1>                   ;(in the user's virtual memory space)
0000442A 8915[A4740000]      <1>  mov   [u.count], edx
                             <1>  ; 14/05/2015
00004430 C705[CF740000]0000- <1>         mov     dword [u.error], 0 ; reset the last error code
00004438 0000                <1>
0000443A C3                  <1>  retn
                             <1>               ; rts r0
                             <1> rw2:
                             <1>  ; 13/05/2015
0000443B C705[CF740000]0A00- <1>  mov   dword [u.error], ERR_FILE_NOT_OPEN ; file not open !
00004443 0000                <1>
00004445 C3                  <1>  retn
                             <1> rw3:
                             <1>  ; 13/05/2015
00004446 C705[CF740000]0B00- <1>  mov   dword [u.error], ERR_FILE_ACCESS ; permission denied !
0000444E 0000                <1>
00004450 F9                  <1>  stc
00004451 C3                  <1>  retn
                             <1>
                             <1> sysopen: ;<open file>
                             <1>  ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 22/05/2013 - 27/05/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysopen' opens a file in following manner:
                             <1>  ;    1) The second argument in a sysopen says whether to
                             <1>  ;     open the file ro read (0) or write (>0).
                             <1>  ;    2) I-node of the particular file is obtained via 'namei'.
                             <1>  ;    3) The file is opened by 'iopen'.
                             <1>  ;    4) Next housekeeping is performed on the fsp table
                             <1>  ;     and the user's open file list - u.fp.
                             <1>  ;     a) u.fp and fsp are scanned for the next available slot.
                             <1>  ;     b) An entry for the file is created in the fsp table.
                             <1>  ;     c) The number of this entry is put on u.fp list.
                             <1>  ;     d) The file descriptor index to u.fp list is pointed
                             <1>  ;        to by u.r0.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysopen; name; mode
                             <1>  ; Arguments:
                             <1>  ;     name - file name or path name
                             <1>  ;     mode - 0 to open for reading
                             <1>  ;            1 to open for writing
                             <1>  ; Inputs: (arguments)
                             <1>  ; Outputs: *u.r0 - index to u.fp list (the file descriptor)
                             <1>  ;             is put into r0's location on the stack.
                             <1>  ; ................................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;        'sysopen' system call has two arguments; so,
                             <1>  ;     * 1st argument, name is pointed to by BX register
                             <1>  ;     * 2nd argument, mode is in CX register
                             <1>  ;
                             <1>  ;     AX register (will be restored via 'u.r0') will return
                             <1>  ;     to the user with the file descriptor/number
                             <1>  ;     (index to u.fp list).
                             <1>  ;
                             <1>  ;call arg2
                             <1>  ; * name - 'u.namep' points to address of file/path name
                             <1>  ;          in the user's program segment ('u.segmnt')
                             <1>  ;          with offset in BX register (as sysopen argument 1).
                             <1>  ; * mode - sysopen argument 2 is in CX register
                             <1>  ;          which is on top of stack.
                             <1>  ;
                             <1>  ; jsr r0,arg2 / get sys args into u.namep and on stack
                             <1>  ;
                             <1>        ; system call registers: ebx, ecx (through 'sysenter')
                             <1>
00004452 891D[98740000]      <1>  mov   [u.namep], ebx
00004458 6651                <1>  push  cx
0000445A E8CC0A0000          <1>  call  namei
                             <1>        ; jsr r0,namei / i-number of file in r1
                             <1>        ;and  ax, ax
                             <1>  ;jz   error ; File not found
0000445F 723B                <1>  jc    short fnotfound ; 14/05/2015
                             <1>  ;jc   error ; 27/05/2013
                             <1>        ; br  error2 / file not found
00004461 665A                <1>        pop   dx ; mode
00004463 6652                <1>  push  dx
                             <1>  ;or   dx, dx
00004465 08D2                <1>  or    dl, dl
                             <1>        ; tst (sp) / is mode = 0 (2nd arg of call;
                             <1>                 ; / 0 means, open for read)
00004467 7403                <1>  jz    short sysopen_0
                             <1>        ; beq 1f / yes, leave i-number positive
00004469 66F7D8              <1>  neg   ax
                             <1>              ; neg r1 / open for writing so make i-number negative
                             <1> sysopen_0: ;1:
0000446C E8281A0000          <1>  call  iopen
                             <1>        ;jsr r0,iopen / open file whose i-number is in r1
00004471 665A                <1>  pop   dx
                             <1>  ;and  dx, dx
00004473 20D2                <1>  and   dl, dl
                             <1>              ; tst (sp)+ / pop the stack and test the mode
00004475 7403                <1>  jz    short sysopen_2
                             <1>              ; beq op1 / is open for read op1
                             <1> sysopen_1: ;op0:
00004477 66F7D8              <1>  neg   ax
                             <1>              ; neg r1
                             <1>             ;/ make i-number positive if open for writing [???]
                             <1>  ;; NOTE: iopen always make i-number positive.
                             <1>  ;; Here i-number becomes negative again. [22/05/2013]
                             <1> sysopen_2: ;op1:
0000447A 31F6                <1>         xor     esi, esi
                             <1>              ; clr r2 / clear registers
0000447C 31DB                <1>         xor     ebx, ebx
                             <1>        ; clr r3
                             <1> sysopen_3: ;1: / scan the list of entries in fsp table
0000447E 389E[86740000]      <1>         cmp     [esi+u.fp], bl ; 0
                             <1>        ; tstb u.fp(r2) / test the entry in the u.fp list
00004484 7625                <1>         jna      short sysopen_4
                             <1>        ; beq 1f / if byte in list is 0 branch
00004486 46                  <1>         inc     esi
                             <1>        ; inc r2 / bump r2 so next byte can be checked
00004487 6683FE0A            <1>         cmp     si, 10
                             <1>        ; cmp r2,$10. / reached end of list?
0000448B 72F1                <1>  jb    short sysopen_3
                             <1>        ; blt 1b / no, go back
                             <1> toomanyf:
                             <1>  ; 14/05/2015
0000448D C705[CF740000]0D00- <1>  mov   dword [u.error], ERR_TOO_MANY_FILES ; too many open files !
00004495 0000                <1>
00004497 E9E0FBFFFF          <1>  jmp   error
                             <1>              ; br error2 / yes, error (no files open)
                             <1> fnotfound:
                             <1>  ; 14/05/2015
0000449C C705[CF740000]0C00- <1>  mov   dword [u.error], ERR_FILE_NOT_FOUND ; file not found !
000044A4 0000                <1>
000044A6 E9D1FBFFFF          <1>  jmp   error
                             <1>
                             <1> sysopen_4: ; 1:
000044AB 6683BB[56720000]00  <1>         cmp     word [ebx+fsp], 0
                             <1>        ; tst fsp(r3) / scan fsp entries
000044B3 7610                <1>         jna     short sysopen_5
                             <1>        ; beq 1f / if 0 branch
                             <1>  ; 14/05/2015 - Retro UNIX 386 v1 modification !
000044B5 6683C30A            <1>         add     bx, 10 ; fsp structure size = 10 bytes/entry
                             <1>        ; add $8.,r3 / add 8 to r3
                             <1>              ; / to bump it to next entry mfsp table
```

```
000044B9 6681FBF401          <1>        cmp     bx, nfiles*10
                             <1>       ; cmp r3,$[nfiles*8.] / done scanning
000044BE 72EB                <1>  jb    short sysopen_4
                             <1>             ; blt 1b / no, back
000044C0 E9B7FBFFFF          <1>  jmp   error
                             <1>             ; br error2 / yes, error
                             <1> sysopen_5: ; 1: / r2 has index to u.fp list; r3, has index to fsp table
000044C5 668983[56720000]    <1>        mov     [ebx+fsp], ax
                             <1>       ; mov r1,fsp(r3) / put i-number of open file
                             <1>             ; / into next available entry in fsp table,
000044CC 668B3D[66740000]    <1>  mov   di, [cdev] ; word ? byte ?
000044D3 6689BB[58720000]    <1>        mov     [ebx+fsp+2], di ; device number
                             <1>       ; mov cdev,fsp+2(r3) / put # of device in next word
000044DA 31FF                <1>        xor  edi, edi
000044DC 89BB[5A720000]      <1>        mov     [ebx+fsp+4], edi ; offset pointer (0)
                             <1>       ; clr fsp+4(r3)
000044E2 6689BB[5E720000]    <1>        mov     [ebx+fsp+8], di ; open count (0), deleted flag (0)
                             <1>             ; clr fsp+6(r3) / clear the next two words
000044E9 89D8                <1>       mov   eax, ebx
000044EB B30A                <1>  mov   bl, 10
000044ED F6F3                <1>  div   bl
                             <1>       ; asr r3
                             <1>       ; asr r3 / divide by 8
                             <1>       ; asr r3 ; / to get number of the fsp entry-1
000044EF FEC0                <1>  inc   al
                             <1>             ; inc r3 / add 1 to get fsp entry number
000044F1 8886[86740000]      <1>        mov     [esi+u.fp], al
                             <1>       ; movb r3,u.fp(r2) / move entry number into
                             <1>             ; / next available slot in u.fp list
000044F7 8935[80740000]      <1>        mov     [u.r0], esi
                             <1>       ; mov r2,*u.r0 / move index to u.fp list
                             <1>                  ; / into r0 loc on stack
000044FD E99AFBFFFF          <1>        jmp  sysret
                             <1>       ; br sysret2
                             <1>
                             <1>  ;
                             <1>  ; 'fsp' table (10 bytes/entry)
                             <1>  ; bit 15                         bit 0
                             <1>  ; ---|-------------------------------------------
                             <1>  ; r/w|            i-number of open file
                             <1>  ; ---|-------------------------------------------
                             <1>  ;              device number
                             <1>  ; -----------------------------------------------
                             <1>  ; offset pointer, r/w pointer to file (bit 0-15)
                             <1>  ; -----------------------------------------------
                             <1>  ; offset pointer, r/w pointer to file (bit 16-31)
                             <1>  ; ----------------------|------------------------
                             <1>  ;  flag that says file  | number of processes
                             <1>  ;   has been deleted    | that have file open
                             <1>  ; ----------------------|------------------------
                             <1>  ;
                             <1>
                             <1> syscreat: ; < create file >
                             <1>  ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 27/05/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'syscreat' called with two arguments; name and mode.
                             <1>  ; u.namep points to name of the file and mode is put
                             <1>  ; on the stack. 'namei' is called to get i-number of the file.
                             <1>  ; If the file aready exists, it's mode and owner remain
                             <1>  ; unchanged, but it is truncated to zero length. If the file
                             <1>  ; did not exist, an i-node is created with the new mode via
                             <1>  ; 'maknod' whether or not the file already existed, it is
                             <1>  ; open for writing. The fsp table is then searched for a free
                             <1>  ; entry. When a free entry is found, proper data is placed
                             <1>  ; in it and the number of this entry is put in the u.fp list.
                             <1>  ; The index to the u.fp (also know as the file descriptor)
                             <1>  ; is put in the user's r0.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     syscreate; name; mode
                             <1>  ; Arguments:
                             <1>  ;     name - name of the file to be created
                             <1>  ;     mode - mode of the file to be created
                             <1>  ; Inputs: (arguments)
                             <1>  ; Outputs: *u.r0 - index to u.fp list
                             <1>  ;              (the file descriptor of new file)
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       'syscreate' system call has two arguments; so,
                             <1>  ;     * 1st argument, name is pointed to by BX register
                             <1>  ;     * 2nd argument, mode is in CX register
                             <1>  ;
                             <1>  ;     AX register (will be restored via 'u.r0') will return
                             <1>  ;     to the user with the file descriptor/number
                             <1>  ;     (index to u.fp list).
                             <1>  ;
                             <1>  ;call arg2
                             <1>  ; * name - 'u.namep' points to address of file/path name
                             <1>  ;          in the user's program segment ('u.segmnt')
                             <1>  ;          with offset in BX register (as sysopen argument 1).
                             <1>  ; * mode - sysopen argument 2 is in CX register
                             <1>  ;          which is on top of stack.
                             <1>  ;
                             <1>              ; jsr r0,arg2 / put file name in u.namep put mode
                             <1>                  ; / on stack
00004502 891D[98740000]      <1>  mov   [u.namep], ebx ; file name address
00004508 6651                <1>  push  cx ; mode
0000450A E81C0A0000          <1>  call  namei
                             <1>        ; jsr r0,namei / get the i-number
                             <1>         ;and ax, ax
                             <1>  ;jz   short syscreat_1
0000450F 7214                <1>  jc    short syscreat_1
                             <1>        ; br  2f / if file doesn't exist 2f
00004511 66F7D8              <1>  neg   ax
                             <1>              ; neg r1 / if file already exists make i-number
                             <1>               ; / negative (open for writing)
00004514 E880190000          <1>  call  iopen
                             <1>              ; jsr r0,iopen /
00004519 E831130000          <1>  call  itrunc
                             <1>              ; jsr r0,itrunc / truncate to 0 length
0000451E 6659                <1>  pop   cx ; pop mode (did not exist in original Unix v1 !?)
00004520 E952FFFFFF          <1>         jmp     sysopen_1
                             <1>              ; br op0
                             <1> syscreat_1: ; 2: / file doesn't exist
00004525 6658                <1>  pop   ax
                             <1>              ; mov (sp)+,r1 / put the mode in r1
00004527 30E4                <1>  xor   ah, ah
                             <1>              ; bic $!377,r1 / clear upper byte
00004529 E8D00C0000          <1>  call  maknod
                             <1>              ; jsr r0,maknod / make an i-node for this file
0000452E 66A1[B2740000]      <1>  mov   ax, [u.dirbuf]
                             <1>              ; mov u.dirbuf,r1 / put i-number
                             <1>                      ; / for this new file in r1
00004534 E93EFFFFFF          <1>         jmp     sysopen_1
                             <1>              ; br op0 / open the file
                             <1>
                             <1> sysmkdir: ; < make directory >
                             <1>  ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 27/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysmkdir' creates an empty directory whose name is
                             <1>  ; pointed to by arg 1. The mode of the directory is arg 2.
                             <1>  ; The special entries '.' and '..' are not present.
                             <1>  ; Errors are indicated if the directory already exists or
                             <1>  ; user is not the super user.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysmkdir; name; mode
                             <1>  ; Arguments:
                             <1>  ;     name - points to the name of the directory
                             <1>  ;     mode - mode of the directory
                             <1>  ; Inputs: (arguments)
                             <1>  ; Outputs: -
                             <1>  ;    (sets 'directory' flag to 1;
                             <1>  ;    'set user id on execution' and 'executable' flags to 0)
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       'sysmkdir' system call has two arguments; so,
                             <1>  ;     * 1st argument, name is pointed to by BX register
                             <1>  ;     * 2nd argument, mode is in CX register
                             <1>  ;
                             <1>
                             <1> ; / make a directory
                             <1>
                             <1>  ;call arg2
                             <1>  ; * name - 'u.namep' points to address of file/path name
                             <1>  ;          in the user's program segment ('u.segmnt')
                             <1>  ;          with offset in BX register (as sysopen argument 1).
                             <1>  ; * mode - sysopen argument 2 is in CX register
                             <1>  ;          which is on top of stack.
                             <1>
                             <1>        ; jsr r0,arg2 / put file name in u.namep put mode
                             <1>                  ; / on stack
00004539 891D[98740000]      <1>  mov   [u.namep], ebx
0000453F 6651                <1>  push  cx ; mode
00004541 E8E5090000          <1>  call  namei
                             <1>              ; jsr r0,namei / get the i-number
                             <1>              ;     br .+4 / if file not found branch around error
                             <1>         ;xor ax, ax
                             <1>  ;jnz  error
00004546 731C                <1>  jnc   short dir_exists ; 14/05/2015
                             <1>  ;jnc  error
                             <1>        ; br  error2 / directory already exists (error)
00004548 803D[C6740000]00    <1>  cmp   byte [u.uid], 0 ; 02/08/2013
                             <1>              ;tstb u.uid / is user the super user
0000454F 7622                <1>  jna   short dir_access_err ; 14/05/2015
                             <1>  ;jna  error
                             <1>              ;bne error2 / no, not allowed
00004551 6658                <1>  pop   ax
                             <1>              ;mov (sp)+,r1 / put the mode in r1
00004553 6683E0CF            <1>  and   ax, 0FFCFh ; 1111111111001111b
```

```
15612                                <1>              ;bic $!317,r1 / all but su and ex
15613                                <1>  ;or   ax , 4000h ; 1011111111111111b
15614 00004557 80CC40                <1>  or    ah, 40h ; Set bit 14 to 1
15615                                <1>              ;bis $40000,r1 / directory flag
15616 0000455A E89F0C0000            <1>  call  maknod
15617                                <1>              ;jsr r0,maknod / make the i-node for the directory
15618 0000455F E938FBFFFF            <1>  jmp   sysret
15619                                <1>              ;br sysret2 /
15620                                <1> dir_exists:
15621                                <1>  ; 14/05/2015
15622 00004564 C705[CF740000]0E00-   <1>  mov   dword [u.error], ERR_DIR_EXISTS ; dir. already exists !
15623 0000456C 0000                  <1>
15624 0000456E E909FBFFFF            <1>  jmp   error
15625                                <1> dir_access_err:
15626                                <1>  ; 14/05/2015
15627 00004573 C705[CF740000]0B00-   <1>  mov   dword [u.error], ERR_DIR_ACCESS ; permission denied !
15628 0000457B 0000                  <1>
15629 0000457D E9FAFAFFFF            <1>  jmp   error
15630                                <1>
15631                                <1> sysclose: ;<close file>
15632                                <1>  ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15633                                <1>  ; 22/05/2013 - 26/05/2013 (Retro UNIX 8086 v1)
15634                                <1>  ;
15635                                <1>  ; 'sysclose', given a file descriptor in 'u.r0', closes the
15636                                <1>  ; associated file. The file descriptor (index to 'u.fp' list)
15637                                <1>  ; is put in r1 and 'fclose' is called.
15638                                <1>  ;
15639                                <1>  ; Calling sequence:
15640                                <1>  ;     sysclose
15641                                <1>  ; Arguments:
15642                                <1>  ;     -
15643                                <1>  ; Inputs: *u.r0 - file descriptor
15644                                <1>  ; Outputs: -
15645                                <1>  ; ...............................................................
15646                                <1>  ;
15647                                <1>  ; Retro UNIX 8086 v1 modification:
15648                                <1>  ;      The user/application program puts file descriptor
15649                                <1>  ;         in BX register as 'sysclose' system call argument.
15650                                <1>  ;      (argument transfer method 1)
15651                                <1>
15652                                <1>  ; / close the file
15653                                <1>
15654 00004582 89D8                  <1>  mov   eax, ebx
15655 00004584 E820090000            <1>  call  fclose
15656                                <1>        ; mov *u.r0,r1 / move index to u.fp list into r1
15657                                <1>        ; jsr r0,fclose / close the file
15658                                <1>                   ; br error2 / unknown file descriptor
15659                                <1>        ; br sysret2
15660                                <1>  ; 14/05/2015
15661 00004589 0F830DFBFFFF          <1>  jnc   sysret
15662 0000458F C705[CF740000]0A00-   <1>  mov   dword [u.error], ERR_FILE_NOT_OPEN ; file not open !
15663 00004597 0000                  <1>
15664 00004599 E9DEFAFFFF            <1>  jmp   error
15665                                <1>
15666                                <1> sysemt:
15667                                <1>  ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
15668                                <1>  ; 10/12/2013 - 20/04/2014 (Retro UNIX 8086 v1)
15669                                <1>  ;
15670                                <1>  ; Retro UNIX 8086 v1 modification:
15671                                <1>  ;     'Enable Multi Tasking'  system call instead
15672                                <1>  ;     of 'Emulator Trap' in original UNIX v1 for PDP-11.
15673                                <1>  ;
15674                                <1>  ; Retro UNIX 8086 v1 feature only!
15675                                <1>  ;     Using purpose: Kernel will start without time-out
15676                                <1>  ;     (internal clock/timer) functionality.
15677                                <1>  ;     Then etc/init will enable clock/timer for
15678                                <1>  ;     multi tasking. (Then it will not be disabled again
15679                                <1>  ;     except hardware reset/restart.)
15680                                <1>  ;
15681                                <1>
15682 0000459E 803D[C6740000]00      <1>  cmp   byte [u.uid], 0 ; root ?
15683                                <1>  ;ja   error
15684 000045A5 0F877AFBFFFF          <1>  ja    badsys ; 14/05/2015
15685                                <1> emt_0:
15686 000045AB FA                    <1>  cli
15687 000045AC 21DB                  <1>  and   ebx, ebx
15688 000045AE 7410                  <1>  jz    short emt_2
15689                                <1>  ; Enable multi tasking -time sharing-
15690 000045B0 B8[B1540000]          <1>  mov   eax, clock
15691                                <1> emt_1:
15692 000045B5 A3[1B080000]          <1>  mov   [x_timer], eax
15693 000045BA FB                    <1>  sti
15694 000045BB E9DCFAFFFF            <1>  jmp   sysret
15695                                <1> emt_2:
15696                                <1>  ; Disable multi tasking -time sharing-
15697 000045C0 B8[23080000]          <1>  mov   eax, u_timer
15698 000045C5 EBEE                  <1>  jmp   short emt_1
15699                                <1>
15700                                <1>  ; Original UNIX v1 'sysemt' routine
15701                                <1> ;sysemt:
15702                                <1>        ;
15703                                <1>  ;jsr    r0,arg; 30 / put the argument of the sysemt call
```

```
                             <1>               ; / in loc 30
                             <1>         ;cmp    30,$core / was the argument a lower address
                             <1>              ; / than core
                             <1>         ;blo    1f / yes, rtssym
                             <1>         ;cmp    30,$ecore / no, was it higher than "core"
                             <1>              ; / and less than "ecore"
                             <1>         ;blo    2f / yes, sysret2
                             <1> ;1:
                             <1>         ;mov    $rtssym,30
                             <1> ;2:
                             <1>         ;br     sysret2
                             <1>
                             <1> sysilgins:
                             <1>  ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 03/06/2013
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;     not a valid system call ! (not in use)
                             <1>  ;
000045C7 E959FBFFFF          <1>  jmp   badsys
                             <1>  ;jmp  error
                             <1>  ;;jmp       sysret
                             <1>
                             <1>  ; Original UNIX v1 'sysemt' routine
                             <1> ;sysilgins: / calculate proper illegal instruction trap address
                             <1>         ;jsr    r0,arg; 10 / take address from sysilgins call
                             <1>                ;/ put it in loc 8.,
                             <1>         ;cmp    10,$core / making it the illegal instruction
                             <1>               ; / trap address
                             <1>         ;blo    1f / is the address a user core address?
                             <1>        ; / yes, go to 2f
                             <1>         ;cmp    10,$ecore
                             <1>         ;blo    2f
                             <1> ;1:
                             <1>         ;mov    $fpsym,10 / no, make 'fpsum' the illegal
                             <1>            ; / instruction trap address for the system
                             <1> ;2:
                             <1>         ;br     sysret2 / return to the caller via 'sysret'
                             <1>
                             <1> sysmdate: ; < change the modification time of a file >
                             <1>  ; 16/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 03/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysmdate' is given a file name. It gets inode of this
                             <1>  ; file into core. The user is checked if he is the owner
                             <1>  ; or super user. If he is neither an error occurs.
                             <1>  ; 'setimod' is then called to set the i-node modification
                             <1>  ; byte and the modification time, but the modification time
                             <1>  ; is overwritten by whatever get put on the stack during
                             <1>  ; a 'systime' system call. This calls are restricted to
                             <1>  ; the super user.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysmdate; name
                             <1>  ; Arguments:
                             <1>  ;     name - points to the name of file
                             <1>  ; Inputs: (arguments)
                             <1>  ; Outputs: -
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;      The user/application program puts address
                             <1>  ;      of the file name in BX register
                             <1>  ;      as 'sysmdate' system call argument.
                             <1>  ;
                             <1> ; / change the modification time of a file
                             <1>        ; jsr r0,arg; u.namep / point u.namep to the file name
000045CC 891D[98740000]      <1>         mov  [u.namep], ebx
000045D2 E854090000          <1>  call  namei
                             <1>        ; jsr r0,namei / get its i-number
000045D7 0F82BFFEFFFF        <1>          jc   fnotfound ; file not found !
                             <1>  ;jc   error
                             <1>        ; br error2 / no, such file
000045DD E838110000          <1>  call  iget
                             <1>        ; jsr r0,iget / get i-node into core
000045E2 A0[C6740000]        <1>  mov   al, [u.uid]
000045E7 3A05[59710000]      <1>  cmp   al, [i.uid]
                             <1>                ; cmpb u.uid,i.uid / is user same as owner
000045ED 7413                <1>  je    short mdate_1
                             <1>                ; beq 1f / yes
000045EF 20C0                <1>  and   al, al
                             <1>        ; tstb u.uid / no, is user the super user
                             <1>  ;jnz  error
                             <1>        ; bne error2 / no, error
000045F1 740F                <1>  jz    short mdate_1
000045F3 C705[CF740000]0B00- <1>  mov   dword [u.error], ERR_FILE_ACCESS ; permission denied !
000045FB 0000                <1>
000045FD E97AFAFFFF          <1>  jmp   error
                             <1> mdate_1: ;1:
00004602 E826120000          <1>  call  setimod
                             <1>                ; jsr r0,setimod / fill in modification data,
                             <1>                       ; / time etc.
00004607 BE[F0700000]        <1>  mov   esi, p_time
```

```
0000460C BF[70710000]         <1>  mov   edi, i.mtim
00004611 A5                   <1>  movsd
                              <1>        ; mov 4(sp),i.mtim / move present time to
                              <1>              ; mov 2(sp),i.mtim+2 / modification time
00004612 E985FAFFFF           <1>         jmp  sysret
                              <1>        ; br sysret2
                              <1>
                              <1> sysstty: ; < set tty status and mode >
                              <1>  ; 17/11/2015
                              <1>  ; 12/11/2015
                              <1>  ; 29/10/2015
                              <1>  ; 17/10/2015
                              <1>  ; 13/10/2015
                              <1>  ; 29/06/2015
                              <1>  ; 27/06/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 02/06/2013 - 12/07/2014 (Retro UNIX 8086 v1)
                              <1>  ;
                              <1>  ; 'sysstty' sets the status and mode of the typewriter
                              <1>  ; whose file descriptor is in (u.r0).
                              <1>  ;
                              <1>  ; Calling sequence:
                              <1>  ;     sysstty; arg
                              <1>  ; Arguments:
                              <1>  ;     arg - address of 3 consequitive words that contain
                              <1>  ;           the source of status data
                              <1>  ; Inputs: ((*u.r0 - file descriptor & argument))
                              <1>  ; Outputs: ((status in address which is pointed to by arg))
                              <1>  ; ...............................................................
                              <1>  ;
                              <1>  ; Retro UNIX 8086 v1 modification:
                              <1>  ;     'sysstty' system call will set the tty
                              <1>  ;     (clear keyboard buffer and set cursor position)
                              <1>  ;      in following manner:
                              <1>  ;   NOTE: All of tty setting functions are here (16/01/2014)
                              <1>  ;
                              <1>  ; Inputs:
                              <1>  ;     BX = 0 --> means
                              <1>  ;        If CL = FFh
                              <1>  ;           set cursor position for console tty, only
                              <1>  ;           CH will be ignored (char. will not be written)
                              <1>  ;        If CH = 0 (CL < FFh)
                              <1>  ;           set console tty for (current) process
                              <1>  ;           CL = tty number (0 to 9)
                              <1>  ;           (If CH = 0, character will not be written)
                              <1>  ;          If CH > 0 (CL < FFh)
                              <1>  ;             CL = tty number (0 to 9)
                              <1>  ;           CH = character will be written
                              <1>  ;             at requested cursor position (in DX)
                              <1>  ;        DX = cursor position for tty number 0 to 7.
                              <1>        ;           (only tty number 0 to 7)
                              <1>  ;          DL = communication parameters (for serial ports)
                              <1>  ;             (only for COM1 and COM2 serial ports)
                              <1>  ;        DH < 0FFh -> DL is valid, initialize serial port
                              <1>  ;                 or set cursor position
                              <1>  ;        DH = 0FFh -> DL is not valid
                              <1>  ;           do not set serial port parameters
                              <1>  ;           or do not set cursor position
                              <1>  ;
                              <1>  ;     BX > 0 --> points to name of tty
                              <1>  ;        CH > 0 -->
                              <1>  ;           CH = character will be written in current
                              <1>  ;                 cursor position (for tty number from 0 to 7)
                              <1>  ;           or character will be sent to serial port
                              <1>  ;           (for tty number 8 or 9)
                              <1>  ;           CL = color of the character if tty number < 8.
                              <1>  ;        CH = 0 --> Do not write a character,
                              <1>  ;           set mode (tty 8 to 9) or
                              <1>  ;           set current cursor positions (tty 0 to 7) only.
                              <1>  ;        DX = cursor position for tty number 0 to 7.
                              <1>  ;        DH = FFh --> Do not set cursor pos (or comm. params.)
                              <1>  ;           (DL is not valid)
                              <1>  ;        DL = communication parameters
                              <1>  ;           for tty number 8 or 9 (COM1 or COM2).
                              <1>  ; Outputs:
                              <1>  ;     cf = 0 -> OK
                              <1>  ;          AL = tty number (0 to 9)
                              <1>  ;          AH = line status if tty number is 8 or 9
                              <1>  ;          AH = process number (of the caller)
                              <1>  ;     cf = 1 means error (requested tty is not ready)
                              <1>  ;          AH = FFh if the tty is locked
                              <1>  ;             (owned by another process)
                              <1>  ;             = process number (of the caller)
                              <1>  ;             (if < FFh and tty number < 8)
                              <1>  ;          AL = tty number (0FFh if it does not exist)
                              <1>  ;          AH = line status if tty number is 8 or 9
                              <1>  ;     NOTE: Video page will be cleared if cf = 0.
                              <1>  ;
                              <1>  ; 27/06/2015 (32 bit modifications)
                              <1>  ; 14/01/2014
00004617 31C0                 <1>  xor   eax, eax
00004619 6648                 <1>  dec   ax ; 17/10/2015
0000461B A3[80740000]         <1>  mov   [u.r0], eax ; 0FFFFh
```

```
00004620 21DB                <1>  and   ebx, ebx
00004622 0F85CB000000        <1>         jnz     sysstty_6
                             <1> ; set console tty
                             <1>  ; 29/10/2015
                             <1>  ; 17/01/2014
00004628 80F909              <1>  cmp   cl, 9
0000462B 7613                <1>  jna   short sysstty_0
                             <1>  ; 17/11/2015
0000462D 80F9FF              <1>  cmp   cl, 0FFh
00004630 7202                <1>  jb    short sysstty_13
00004632 88CD                <1>  mov   ch, cl ; force CH value to FFh
                             <1> sysstty_13:
00004634 8A1D[C9740000]      <1>  mov   bl, [u.uno] ; process number
0000463A 8A8B[D5710000]      <1>  mov   cl, [ebx+p.ttyc-1] ; current/console tty
                             <1> sysstty_0:
                             <1>  ; 29/06/2015
00004640 6652                <1>  push  dx
00004642 6651                <1>  push  cx
00004644 30D2                <1>  xor   dl, dl      ; sysstty call sign
00004646 88C8                <1>  mov   al, cl
00004648 A2[80740000]        <1>  mov   [u.r0], al ; tyy number (0 to 9)
0000464D E8E1180000          <1>  call  ottyp
00004652 6659                <1>  pop   cx
00004654 665A                <1>  pop   dx
                             <1>  ;
00004656 7257                <1>  jc    short sysstty_pd_err
                             <1>  ;
00004658 80F908              <1>  cmp   cl, 8
0000465B 7222                <1>  jb    short sysstty_2
                             <1>  ;
0000465D 80FEFF              <1>  cmp   dh, 0FFh
00004660 741D                <1>  je    short sysstty_2
                             <1>        ; set communication parameters for serial ports
                             <1>  ; 29/10/2015
00004662 88D4                <1>  mov   ah, dl ; communication parameters
                             <1>        ; ah = 0E3h = 11100011b = 115200 baud,
                             <1>        ;                  THRE int + RDA int
                             <1>        ; ah = 23h = 00100011b = 9600 baud,
                             <1>        ;                  THRE int + RDA int
00004664 28C0                <1>  sub   al, al ; 0
                             <1>  ; 12/07/2014
00004666 80F909              <1>  cmp   cl, 9
00004669 7202                <1>  jb    short sysstty_1
0000466B FEC0                <1>  inc   al
                             <1> sysstty_1:
0000466D 6651                <1>  push  cx
                             <1>  ; 29/06/2015
0000466F E8B5F4FFFF          <1>  call  sp_setp ; Set serial port communication parameters
00004674 66890D[81740000]    <1>  mov   [u.r0+1], cx ; Line status (ah)
                             <1>                   ; Modem status (EAX bits 16 to 23)
0000467B 6659                <1>  pop   cx
0000467D 7265                <1>         jc      short sysstty_tmout_err ; 29/10/2015
                             <1> sysstty_2:
                             <1>  ; 17/01/2014
0000467F 20ED                <1>  and   ch, ch      ; set cursor position
                             <1>              ; or comm. parameters ONLY
00004681 750D                <1>  jnz   short sysstty_3
00004683 0FB61D[C9740000]    <1>  movzx ebx, byte [u.uno] ; process number
0000468A 888B[D5710000]      <1>  mov   [ebx+p.ttyc-1], cl ; console tty
                             <1> sysstty_3:
                             <1>  ; 16/01/2014
00004690 88E8                <1>  mov   al, ch ; character  ; 0 to FFh
                             <1>  ; 17/11/2015
00004692 B507                <1>  mov   ch, 7  ; Default color (light gray)
00004694 38E9                <1>  cmp   cl, ch ; 7 (tty number)
00004696 0F86C5000000        <1>         jna     sysstty_9
                             <1> sysstty_12:
                             <1>  ;; BX = 0, CL = 8 or CL = 9
                             <1>  ; (Set specified serial port as console tty port)
                             <1>  ; CH = character to be written
                             <1>  ; 15/04/2014
                             <1>  ; CH = 0 --> initialization only
                             <1>  ; AL = character
                             <1>  ; 26/06/2014
0000469C 880D[CE740000]      <1>  mov   [u.ttyn], cl
                             <1>  ; 12/07/2014
000046A2 88CC                <1>  mov   ah, cl ; tty number (8 or 9)
000046A4 20C0                <1>  and   al, al
000046A6 7416                <1>  jz    short sysstty_4 ; al = ch = 0
                             <1>  ; 04/07/2014
000046A8 E8941E0000          <1>  call  sndc
                             <1>  ; 12/07/2014
000046AD EB1B                <1>  jmp   short sysstty_5
                             <1> sysstty_pd_err: ; 29/06/2015
                             <1>  ; 'permission denied !' error
000046AF C705[CF740000]0B00- <1>  mov   dword [u.error], ERR_NOT_OWNER
000046B7 0000                <1>
000046B9 E9BEF9FFFF          <1>  jmp   error
                             <1> sysstty_4:
                             <1>  ; 12/07/2014
                             <1>  ;xchg       ah, al      ; al = 0 -> al = ah, ah = 0
000046BE 88E0                <1>  mov   al, ah ; 29/06/2015
```

```
000046C0 2C08                <1>  sub   al, 8
                             <1>  ; 27/06/2015
000046C2 E85AF4FFFF          <1>  call  sp_status ; get serial port status
                             <1>  ; AL = Line status, AH = Modem status
                             <1>  ; 12/11/2015
000046C7 3C80                <1>  cmp   al, 80h
000046C9 F5                  <1>  cmc
                             <1> sysstty_5:
000046CA 66A3[81740000]      <1>  mov   [u.r0+1], ax ; ah = line status
                             <1>             ; EAX bits 16-23 = modem status
000046D0 9C                  <1>  pushf
000046D1 30D2                <1>  xor   dl, dl ; sysstty call sign
000046D3 A0[CE740000]        <1>  mov   al, [u.ttyn] ; 26/06/2014
000046D8 E869190000          <1>  call  cttyp
000046DD 9D                  <1>  popf
000046DE 0F83B8F9FFFF        <1>  jnc   sysret      ; time out error
                             <1>
                             <1> sysstty_tmout_err:
000046E4 C705[CF740000]1900- <1>  mov   dword [u.error], ERR_TIME_OUT
000046EC 0000                <1>
000046EE E989F9FFFF          <1>  jmp   error
                             <1> sysstty_6:
000046F3 6652                <1>  push  dx
000046F5 6651                <1>  push  cx
000046F7 891D[98740000]      <1>  mov   [u.namep], ebx
000046FD E829080000          <1>  call  namei
00004702 6659                <1>  pop   cx
00004704 665A                <1>  pop   dx
00004706 720E                <1>  jc    short sysstty_inv_dn
                             <1>  ;
00004708 6683F813            <1>  cmp   ax, 19  ; inode number of /dev/COM2
0000470C 7708                <1>  ja    short sysstty_inv_dn ; 27/06/2015
                             <1>  ;
0000470E 3C0A                <1>  cmp   al, 10 ; /dev/tty0 .. /dev/tty7
                             <1>               ; /dev/COM1, /dev/COM2
00004710 7213                <1>  jb    short sysstty_7
00004712 2C0A                <1>  sub   al, 10
00004714 EB20                <1>  jmp   short sysstty_8
                             <1> sysstty_inv_dn:
                             <1>  ; 27/06/2015
                             <1>  ; Invalid device name (not a tty) ! error
                             <1>  ; (Device is not a tty or device name not found)
00004716 C705[CF740000]1800- <1>  mov   dword [u.error], ERR_INV_DEV_NAME
0000471E 0000                <1>
00004720 E957F9FFFF          <1>  jmp   error
                             <1> sysstty_7:
00004725 3C01                <1>  cmp   al, 1 ; /dev/tty
00004727 75ED                <1>  jne   short sysstty_inv_dn ; 27/06/2015
00004729 0FB61D[C9740000]    <1>  movzx ebx, byte [u.uno] ; process number
00004730 8A83[D5710000]      <1>  mov   al, [ebx+p.ttyc-1] ; console tty
                             <1> sysstty_8:
00004736 A2[80740000]        <1>  mov   [u.r0], al
0000473B 6652                <1>  push  dx
0000473D 6650                <1>  push  ax
0000473F 6651                <1>  push  cx
00004741 E8ED170000          <1>  call  ottyp
00004746 6659                <1>  pop   cx
00004748 6658                <1>  pop   ax
0000474A 665A                <1>  pop   dx
0000474C 0F825DFFFFFF        <1>        jc      sysstty_pd_err ; 'permission denied !'
                             <1>  ; 29/10/2015
00004752 86E9                <1>  xchg  ch, cl
                             <1>        ; cl = character, ch = color code
00004754 86C1                <1>  xchg  al, cl
                             <1>        ; al = character, cl = tty number
00004756 80F907              <1>  cmp   cl, 7
00004759 0F873DFFFFFF        <1>        ja      sysstty_12
                             <1>  ;
                             <1>  ; 16/01/2014
0000475F 30FF                <1>  xor   bh, bh
                             <1>  ;
                             <1> sysstty_9:   ; tty 0 to tty 7
                             <1>  ; al = character
00004761 80FEFF              <1>  cmp   dh, 0FFh ; Do not set cursor position
00004764 740F                <1>  je    short sysstty_10
00004766 6651                <1>  push  cx
00004768 6650                <1>  push  ax
                             <1>  ; movzx, ebx, cl
0000476A 88CB                <1>  mov   bl, cl ; (tty number = video page number)
0000476C E8D0CEFFFF          <1>  call  set_cpos
00004771 6658                <1>  pop   ax
00004773 6659                <1>  pop   cx
                             <1> sysstty_10:
                             <1>  ; 29/10/2015
00004775 08C0                <1>  or    al, al ; character
00004777 740F                <1>  jz      short sysstty_11 ; al = 0
                             <1>  ; 17/11/2015
00004779 3CFF                <1>  cmp   al, 0FFh
0000477B 730B                <1>  jnb   short sysstty_11
                             <1>        ; ch > 0 and ch < FFh
                             <1>  ; write a character at current cursor position
0000477D 88EC                <1>  mov   ah, ch ; color/attribute
```

```
                             <1>  ; 12/07/2014
0000477F 6651                <1>  push  cx
00004781 E806D0FFFF          <1>  call  write_c_current
00004786 6659                <1>  pop   cx
                             <1> sysstty_11:
                             <1>  ; 14/01/2014
00004788 30D2                <1>  xor   dl, dl ; sysstty call sign
                             <1>  ; 18/01/2014
                             <1>  ;movzx      eax, cl ; 27/06/2015
0000478A 88C8                <1>  mov   al, cl
0000478C E8B5180000          <1>  call  cttyp
00004791 E906F9FFFF          <1>  jmp   sysret
                             <1>
                             <1> ; Original UNIX v1 'sysstty' routine:
                             <1> ; gtty:
                             <1> ;sysstty: / set mode of typewriter; 3 consequtive word arguments
                             <1>         ;jsr    r0,gtty / r1 will have offset to tty block,
                             <1>  ;           / r2 has source
                             <1>         ;mov    r2,-(sp)
                             <1>         ;mov    r1,-(sp) / put r1 and r2 on the stack
                             <1> ;1: / flush the clist wait till typewriter is quiescent
                             <1>         ;mov    (sp),r1 / restore r1 to tty block offset
                             <1>         ;movb   tty+3(r1),0f / put cc offset into getc argument
                             <1>         ;mov    $240,*$ps / set processor priority to 5
                             <1>         ;jsr    r0,getc; 0:../ put character from clist in r1
                             <1>         ;       br .+4 / list empty, skip branch
                             <1>         ;br     1b / get another character until list is empty
                             <1>         ;mov    0b,r1 / move cc offset to r1
                             <1>         ;inc    r1 / bump it for output clist
                             <1>         ;tstb   cc(r1) / is it 0
                             <1>         ;beq    1f / yes, no characters to output
                             <1>  ;mov    r1,0f / no, put offset in sleep arg
                             <1>         ;jsr    r0,sleep; 0:.. / put tty output process to sleep
                             <1>         ;br     1b / try to calm it down again
                             <1> ;1:
                             <1>         ;mov    (sp)+,r1
                             <1>         ;mov    (sp)+,r2 / restore registers
                             <1>  ;mov    (r2)+,r3 / put reader control status in r3
                             <1>         ;beq    1f / if 0, 1f
                             <1>         ;mov    r3,rcsr(r1) / move r.c. status to reader
                             <1>         ;                   / control status register
                             <1> ;1:
                             <1>         ;mov    (r2)+,r3 / move pointer control status to r3
                             <1>         ;beq    1f / if 0 1f
                             <1>         ;mov    r3,tcsr(r1) / move p.c. status to printer
                             <1>  ;               / control status reg
                             <1> ;1:
                             <1>         ;mov    (r2)+,tty+4(r1) / move to flag byte of tty block
                             <1>         ;jmp     sysret2 / return to user
                             <1>
                             <1> sysgtty: ; < get tty status >
                             <1>  ; 23/11/2015
                             <1>  ; 29/10/2015
                             <1>  ; 17/10/2015
                             <1>  ; 28/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 30/05/2013 - 12/07/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysgtty' gets the status of tty in question.
                             <1>  ; It stores in the three words addressed by it's argument
                             <1>  ; the status of the typewriter whose file descriptor
                             <1>  ; in (u.r0).
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysgtty; arg
                             <1>  ; Arguments:
                             <1>  ;     arg - address of 3 words destination of the status
                             <1>  ; Inputs: ((*u.r0 - file descriptor))
                             <1>  ; Outputs: ((status in address which is pointed to by arg))
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;     'sysgtty' system call will return status of tty
                             <1>  ;     (keyboard, serial port and video page status)
                             <1>  ;      in following manner:
                             <1>  ;
                             <1>  ; Inputs:
                             <1>  ;     BX = 0 --> means
                             <1>  ;          CH = 0 -->   'return status of the console tty'
                             <1>  ;                      for (current) process
                             <1>  ;          CL = 0 --> return keyboard status (tty 0 to 9)
                             <1>  ;          CL = 1 --> return video page status (tty 0 to 7)
                             <1>  ;          CL = 1 --> return serial port status (tty 8 & 9)
                             <1>  ;          CH > 0 -->   tty number + 1
                             <1>  ;
                             <1>  ;     BX > 0 --> points to name of tty
                             <1>  ;          CL = 0 --> return keyboard status
                             <1>  ;          CL = 1 --> return video page status
                             <1>  ;          CH = undefined
                             <1>  ;
                             <1>  ; Outputs:
                             <1>  ;     cf = 0 ->
                             <1>  ;
                             <1>  ;          AL = tty number from 0 to 9
                             <1>  ;             (0 to 7 is also the video page of the tty)
                             <1>  ;          AH = 0 if the tty is free/unused
                             <1>  ;          AH = the process number of the caller
                             <1>  ;          AH = FFh if the tty is locked by another process
                             <1>  ;
                             <1>  ;       (if calling is for serial port status)
                             <1>  ;          BX = serial port status if tty number is 8 or 9
                             <1>  ;             (BH = modem status, BL = Line status)
                             <1>  ;          CX = 0FFFFh (if data is ready)
                             <1>  ;          CX = 0 (if data is not ready or undefined)
                             <1>  ;
                             <1>  ;       (if calling is for keyboard status)
                             <1>  ;          BX = current character in tty/keyboard buffer
                             <1>  ;             (BH = scan code, BL = ascii code)
                             <1>  ;             (BX=0 if there is not a waiting character)
                             <1>  ;          CX  is undefined
                             <1>  ;
                             <1>  ;       (if calling is for video page status)
                             <1>  ;          BX = cursor position on the video page
                             <1>  ;             if tty number < 8
                             <1>  ;             (BH = row, BL = column)
                             <1>  ;          CX = current character (in cursor position)
                             <1>  ;             on the video page of the tty
                             <1>  ;             if tty number < 8
                             <1>  ;             (CH = color, CL = character)
                             <1>  ;
                             <1>  ;     cf = 1 means error (requested tty is not ready)
                             <1>  ;
                             <1>  ;          AH = FFh if the caller is not owner of
                             <1>  ;             specified tty or console tty
                             <1>  ;          AL = tty number (0FFh if it does not exist)
                             <1>  ;          BX, CX are undefined if cf = 1
                             <1>  ;
                             <1>  ;       (If tty number is 8 or 9)
                             <1>  ;          AL = tty number
                             <1>  ;          AH = the process number of the caller
                             <1>  ;          BX = serial port status
                             <1>  ;            (BH = modem status, BL = Line status)
                             <1>  ;          CX = 0
                             <1>  ;
                             <1>
                             <1> gtty:   ; get (requested) tty number
                             <1>  ; 17/10/2015
                             <1>  ; 28/06/2015 (Retro UNIX 386 v1 - 32 bit modifications)
                             <1>  ; 30/05/2013 - 12/07/2014
                             <1>  ; Retro UNIX 8086 v1 modification !
                             <1>  ;
                             <1>  ; ((Modified regs: eAX, eBX, eCX, eDX, eSI, eDI, eBP))
                             <1>  ;
                             <1>  ; 28/06/2015 (32 bit modifications)
                             <1>  ; 16/01/2014
00004796 31C0                <1>  xor   eax, eax
00004798 6648                <1>  dec   ax ; 17/10/2015
0000479A A3[80740000]        <1>  mov   [u.r0], eax ; 0FFFFh
0000479F 80F901              <1>  cmp   cl, 1
000047A2 760F                <1>  jna   short sysgtty_0
                             <1> sysgtty_invp:
                             <1>  ; 28/06/2015
000047A4 C705[CF740000]1700- <1>         mov     dword [u.error], ERR_INV_PARAMETER ; 'invalid parameter !'
000047AC 0000                <1>
000047AE E9C9F8FFFF          <1>  jmp   error
                             <1> sysgtty_0:
000047B3 21DB                <1>  and   ebx, ebx
000047B5 7430                <1>  jz    short sysgtty_1
                             <1>  ;
000047B7 891D[98740000]      <1>  mov   [u.namep], ebx
000047BD 6651                <1>  push  cx ; 23/11/2015
000047BF E867070000          <1>  call  namei
000047C4 6659                <1>  pop   cx ; 23/11/2015
000047C6 7210                <1>  jc    short sysgtty_inv_dn ; 28/06/2015
                             <1>  ;
000047C8 6683F801            <1>  cmp   ax, 1
000047CC 7622                <1>  jna   short sysgtty_2
000047CE 6683E80A            <1>  sub   ax, 10
000047D2 6683F809            <1>  cmp   ax, 9
                             <1>  ;ja   short sysgtty_inv_dn
                             <1>  ;mov  ch, al
                             <1>  ;jmp  short sysgtty_4
                             <1>  ; 23/11/2015
000047D6 7629                <1>  jna   short sysgtty_4
                             <1> sysgtty_inv_dn:
                             <1>  ; 28/06/2015
                             <1>  ; Invalid device name (not a tty) ! error
                             <1>  ; (Device is not a tty or device name not found)
000047D8 C705[CF740000]1800- <1>  mov   dword [u.error], ERR_INV_DEV_NAME
000047E0 0000                <1>
000047E2 E995F8FFFF          <1>  jmp   error
                             <1> sysgtty_1:
                             <1>  ; 16/01/2014
000047E7 80FD0A              <1>  cmp   ch, 10
000047EA 77B8                <1>  ja    short sysgtty_invp ; 28/06/2015
```

```
000047EC FECD                <1>  dec   ch ; 0 -> FFh (negative)
000047EE 790F                <1>  jns   short sysgtty_3 ; not negative
                             <1>  ;
                             <1> sysgtty_2:
                             <1>  ; get tty number of console tty
000047F0 8A25[C9740000]      <1>  mov   ah, [u.uno]
                             <1>  ; 28/06/2015
000047F6 0FB6DC              <1>  movzx       ebx, ah
000047F9 8AAB[D5710000]      <1>  mov   ch, [ebx+p.ttyc-1]
                             <1> sysgtty_3:
000047FF 88E8                <1>  mov   al, ch
                             <1> sysgtty_4:
00004801 A2[80740000]        <1>  mov   [u.r0], al
                             <1>  ; 28/06/2015
                             <1>  ;cmp  al, 9
                             <1>  ;ja   short sysgtty_invp
00004806 8B2D[7C740000]      <1>  mov   ebp, [u.usp]
                             <1>  ; 23/11/2015
0000480C 20C9                <1>  and   cl, cl
0000480E 7436                <1>  jz    short sysgtty_6 ; keyboard status
00004810 3C08                <1>  cmp   al, 8 ; cmp ch, 8
00004812 7232                <1>  jb    short sysgtty_6 ; video page status
                             <1>  ; serial port status
                             <1>  ; 12/07/2014
                             <1>  ;mov  dx, 0
                             <1>  ;je   short sysgtty_5
                             <1>  ;inc  dl
                             <1> ;sysgtty_5:
                             <1>  ; 28/06/2015
00004814 2C08                <1>  sub   al, 8
00004816 E806F3FFFF          <1>  call  sp_status ; serial (COM) port (line) status
                             <1>  ; AL = Line status, AH = Modem status
0000481B 66894510            <1>  mov   [ebp+16], ax ; serial port status (in EBX)
0000481F 8A25[C9740000]      <1>  mov   ah, [u.uno]
00004825 8825[81740000]      <1>         mov     [u.r0+1], ah
0000482B 66C745180000        <1>  mov   word [ebp+24], 0 ; data status (0 = not ready)
                             <1>                    ; (in ECX)
00004831 A880                <1>  test  al, 80h
00004833 7565                <1>  jnz   short sysgtty_dnr_err ; 29/06/2015
00004835 A801                <1>  test  al, 1
00004837 0F845FF8FFFF        <1>  jz    sysret
0000483D 66FF4D18            <1>  dec   word [ebp+24] ; data status (FFFFh = ready)
00004841 E956F8FFFF          <1>  jmp   sysret
                             <1> sysgtty_6:
00004846 A2[CE740000]        <1>  mov   [u.ttyn], al ; tty number
                             <1>  ;movzx      ebx, al
0000484B 88C3                <1>  mov   bl, al ; tty number (0 to 9)
0000484D D0E3                <1>  shl   bl, 1  ; aligned to word
                             <1>  ; 22/04/2014 - 29/06/2015
0000484F 81C3[F4700000]      <1>         add     ebx, ttyl
00004855 8A23                <1>  mov   ah, [ebx]
00004857 3A25[C9740000]      <1>  cmp   ah, [u.uno]
0000485D 7404                <1>  je    short sysgtty_7
0000485F 20E4                <1>  and   ah, ah
                             <1>  ;jz   short sysgtty_7
00004861 7506                <1>  jnz   short sysgtty_8
                             <1>  ;mov  ah, 0FFh
                             <1> sysgtty_7:
00004863 8825[81740000]      <1>         mov     [u.r0+1], ah
                             <1> sysgtty_8:
00004869 08C9                <1>  or    cl, cl
0000486B 7510                <1>  jnz   short sysgtty_9
0000486D B001                <1>  mov   al, 1  ; test a key is available
0000486F E85E1C0000          <1>  call  getc
00004874 66894510            <1>  mov   [ebp+16], ax ; bx, character
00004878 E91FF8FFFF          <1>  jmp   sysret
                             <1> sysgtty_9:
0000487D 8A1D[CE740000]      <1>  mov   bl, [u.ttyn]
                             <1>  ; bl = video page number
00004883 E8D21D0000          <1>  call  get_cpos
                             <1>  ; dx = cursor position
00004888 66895510            <1>  mov   [ebp+16], dx ; bx
                             <1>  ;mov  bl, [u.ttyn]
                             <1>  ; bl = video page number
0000488C E8DA1D0000          <1>  call  read_ac_current
                             <1>  ; ax = character and attribute/color
00004891 66894518            <1>  mov   [ebp+24], ax ; cx
00004895 E902F8FFFF          <1>  jmp   sysret
                             <1> sysgtty_dnr_err:
                             <1>  ; 'device not responding !' error
                             <1>  ;mov  dword [u.error], ERR_TIME_OUT ; 25
0000489A C705[CF740000]1900- <1>  mov   dword [u.error], ERR_DEV_NOT_RESP ;  25
000048A2 0000                <1>
000048A4 E9D3F7FFFF          <1>  jmp   error
                             <1>
                             <1> ; Original UNIX v1 'sysgtty' routine:
                             <1> ; sysgtty:
                             <1>         ;jsr    r0,gtty / r1 will have offset to tty block,
                             <1>  ;            / r2 has destination
                             <1>         ;mov    rcsr(r1),(r2)+ / put reader control status
                             <1>  ;                        / in 1st word of dest
                             <1>         ;mov    tcsr(r1),(r2)+ / put printer control status
                             <1>  ;                      / in 2nd word of dest
                             <1>         ;mov    tty+4(r1),(r2)+ / put mode in 3rd word
                             <1>         ;jmp    sysret2 / return to user
                             <1> 
                             <1> ; Original UNIX v1 'gtty' routine:
                             <1> ; gtty:
                             <1>         ;jsr    r0,arg; u.off / put first arg in u.off
                             <1>         ;mov    *u.r0,r1 / put file descriptor in r1
                             <1>         ;jsr    r0,getf / get the i-number of the file
                             <1>         ;tst    r1 / is it open for reading
                             <1>         ;bgt    1f / yes
                             <1>         ;neg    r1 / no, i-number is negative,
                             <1>  ;          / so make it positive
                             <1> ;1:
                             <1>         ;sub    $14.,r1 / get i-number of tty0
                             <1>         ;cmp    r1,$ntty-1 / is there such a typewriter
                             <1>         ;bhis   error9 / no, error
                             <1>         ;asl    r1 / 0%2
                             <1>         ;asl    r1 / 0%4 / yes
                             <1>         ;asl    r1 / 0%8 / multiply by 8 so r1 points to
                             <1>  ;            ; / tty block
                             <1>         ;mov    u.off,r2 / put argument in r2
                             <1>         ;rts    r0 / return
                                 %include 'u2.s'        ; 11/05/2015
                             <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS2.INC
                             <1> ; Last Modification: 23/10/2015
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
                             <1> ; (v0.1 - Beginning: 11/07/2012)
                             <1> ;
                             <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                             <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                             <1> ; <Bell Laboratories (17/3/1972)>
                             <1> ; <Preliminary Release of UNIX Implementation Document>
                             <1> ;
                             <1> ; Retro UNIX 8086 v1 - U2.ASM (24/03/2014) //// UNIX v1 -> u2.s
                             <1> ;
                             <1> ; ****************************************************************************
                             <1> 
                             <1> syslink:
                             <1>  ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 19/06/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'syslink' is given two arguments, name 1 and name 2.
                             <1>  ; name 1 is a file that already exists. name 2 is the name
                             <1>  ; given to the entry that will go in the current directory.
                             <1>  ; name2 will then be a link to the name 1 file. The i-number
                             <1>  ; in the name 2 entry of current directory is the same
                             <1>  ; i-number for the name 1 file.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     syslink; name 1; name 2
                             <1>  ; Arguments:
                             <1>  ;     name 1 - file name to which link will be created.
                             <1>  ;     name 2 - name of entry in current directory that
                             <1>  ;            links to name 1.
                             <1>  ; Inputs: -
                             <1>  ; Outputs: -
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       'syslink' system call has two arguments; so,
                             <1>  ;     * 1st argument, name 1 is pointed to by BX register
                             <1>  ;     * 2nd argument, name 2 is pointed to by CX register
                             <1>  ;
                             <1>        ; / name1, name2
                             <1>        ;jsr r0,arg2 / u.namep has 1st arg u.off has 2nd
000048A9 891D[98740000]      <1>  mov   [u.namep], ebx
000048AF 51                  <1>  push  ecx
000048B0 E876060000          <1>  call  namei
                             <1>        ; jsr r0,namei / find the i-number associated with
                             <1>                  ; / the 1st path name
                             <1>        ;;and ax, ax
                             <1>  ;;jz  error ; File not found
                             <1>  ;jc   error
                             <1>        ; br error9 / cannot be found
000048B5 730F                <1>  jnc   short syslink0
                             <1>  ;pop  ecx
                             <1>  ; 'file not found !' error
000048B7 C705[CF740000]0C00- <1>  mov   dword [u.error], ERR_FILE_NOT_FOUND ; 12
000048BF 0000                <1>
000048C1 E9B6F7FFFF          <1>  jmp   error
                             <1> syslink0:
000048C6 E84F0E0000          <1>  call  iget
                             <1>        ; jsr r0,iget / get the i-node into core
000048CB 8F05[98740000]      <1>  pop   dword [u.namep] ; ecx
                             <1>        ; mov (sp)+,u.namep / u.namep points to 2nd name
000048D1 6650                <1>  push  ax
                             <1>        ; mov r1,-(sp) / put i-number of name1 on the stack
                             <1>                 ; / (a link to this file is to be created)
000048D3 66FF35[66740000]    <1>  push  word [cdev]
                             <1>        ; mov cdev,-(sp) / put i-nodes device on the stack
```

```
000048DA E855000000          <1>  call  isdir
                             <1>        ; jsr r0,isdir / is it a directory
000048DF E847060000          <1>  call  namei
                             <1>        ; jsr r0,namei / no, get i-number of name2
                             <1>  ;jnc  error
                             <1>        ; br .+4   / not found
                             <1>               ; / so r1 = i-number of current directory
                             <1>               ; / ii = i-number of current directory
                             <1>        ; br error9 / file already exists., error
000048E4 720F                <1>  jc    short syslink1
                             <1>  ; pop ax
                             <1>  ; pop ax
                             <1>  ; 'file exists !' error
000048E6 C705[CF740000]0E00- <1>  mov   dword [u.error], ERR_FILE_EXISTS ; 14
000048EE 0000                <1>
000048F0 E987F7FFFF          <1>  jmp   error
                             <1> syslink1:
000048F5 6659                <1>  pop   cx
                             <1>  ;cmp  cx, [cdev]
000048F7 3A0D[66740000]      <1>  cmp   cl, [cdev]
                             <1>  ;jne  error
                             <1>        ; cmp (sp)+,cdev / u.dirp now points to
                             <1>                     ; / end of current directory
                             <1>          ; bne error9
000048FD 740F                <1>  je    short syslink2
                             <1>  ; 'not same drive !' error
000048FF C705[CF740000]1500- <1>  mov   dword [u.error],  ERR_DRV_NOT_SAME ; 21
00004907 0000                <1>
00004909 E96EF7FFFF          <1>  jmp   error
                             <1> syslink2:
0000490E 6658                <1>  pop   ax
00004910 6650                <1>  push  ax
00004912 66A3[B2740000]      <1>  mov   [u.dirbuf], ax
                             <1>        ; mov (sp),u.dirbuf / i-number of name1 into u.dirbuf
00004918 E8A8000000          <1>  call  mkdir
                             <1>        ; jsr r0,mkdir / make directory entry for name2
                             <1>                   ; / in current directory
0000491D 6658                <1>  pop   ax
                             <1>        ; mov (sp)+,r1 / r1 has i-number of name1
0000491F E8F60D0000          <1>  call  iget
                             <1>        ; jsr r0,iget / get i-node into core
00004924 FE05[58710000]      <1>  inc   byte [i.nlks]
                             <1>        ; incb i.nlks / add 1 to its number of links
0000492A E8FE0E0000          <1>  call  setimod
                             <1>        ; jsr r0,setimod / set the i-node modified flag
0000492F E968F7FFFF          <1>  jmp   sysret
                             <1>
                             <1> isdir:
                             <1>  ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 04/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'isdir' check to see if the i-node whose i-number is in r1
                             <1>  ;  is a directory. If it is, an error occurs, because 'isdir'
                             <1>  ;  called by syslink and sysunlink to make sure directories
                             <1>  ;  are not linked. If the user is the super user (u.uid=0),
                             <1>  ; 'isdir' does not bother checking. The current i-node
                             <1>  ;  is not disturbed.
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - contains the i-number whose i-node is being checked.
                             <1>  ;    u.uid - user id
                             <1>  ; OUTPUTS ->
                             <1>  ;    r1 - contains current i-number upon exit
                             <1>  ;      (current i-node back in core)
                             <1>  ;
                             <1>  ; ((AX = R1))
                             <1>  ;
                             <1>         ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
                             <1>  ;
                             <1>
                             <1>  ; / if the i-node whose i-number is in r1 is a directory
                             <1>  ; / there is an error unless super user made the call
                             <1>
00004934 803D[C6740000]00    <1>  cmp   byte [u.uid], 0
                             <1>        ; tstb u.uid / super user
0000493B 762D                <1>  jna   short isdir1
                             <1>        ; beq 1f / yes, don't care
0000493D 66FF35[62740000]    <1>  push  word [ii]
                             <1>        ; mov ii,-(sp) / put current i-number on stack
00004944 E8D10D0000          <1>  call  iget
                             <1>        ; jsr r0,iget / get i-node into core (i-number in r1)
00004949 66F705[56710000]00- <1>  test  word [i.flgs], 4000h ; Bit 14 : Directory flag
00004951 40                  <1>
                             <1>        ; bit $40000,i.flgs / is it a directory
                             <1>  ;jnz  error
                             <1>        ; bne error9 / yes, error
00004952 740F                <1>  jz    short isdir0
00004954 C705[CF740000]0B00- <1>  mov   dword [u.error], ERR_NOT_FILE  ; 11 ; ERR_DIR_ACCESS
0000495C 0000                <1>
                             <1>                    ; 'permission denied !' error
                             <1>  ; pop ax
0000495E E919F7FFFF          <1>  jmp   error
```

```
                             <1> isdir0:
00004963 6658                <1>  pop   ax
                             <1>        ; mov (sp)+,r1 / no, put current i-number in r1 (ii)
00004965 E8B00D0000          <1>  call  iget
                             <1>        ; jsr r0,iget / get it back in
                             <1> isdir1: ; 1:
0000496A C3                  <1>  retn
                             <1>        ; rts r0
                             <1> 
                             <1> sysunlink:
                             <1>  ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 19/06/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysunlink' removes the entry for the file pointed to by
                             <1>  ; name from its directory. If this entry was the last link
                             <1>  ; to the file, the contents of the file are freed and the
                             <1>  ; file is destroyed. If, however, the file was open in any
                             <1>  ; process, the actual destruction is delayed until it is
                             <1>  ; closed, even though the directory entry has disappeared.
                             <1>  ;
                             <1>  ; The error bit (e-bit) is set to indicate that the file
                             <1>  ; does not exist or that its directory can not be written.
                             <1>  ; Write permission is not required on the file itself.
                             <1>  ; It is also illegal to unlink a directory (except for
                             <1>  ; the superuser).
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysunlink; name
                             <1>  ; Arguments:
                             <1>  ;     name - name of directory entry to be removed
                             <1>  ; Inputs: -
                             <1>  ; Outputs: -
                             <1>  ; ............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;      The user/application program puts address of the name
                             <1>  ;         in BX register as 'sysunlink' system call argument.
                             <1> 
                             <1>  ; / name - remove link name
0000496B 891D[98740000]      <1>  mov   [u.namep], ebx
                             <1>        ;jsr r0,arg; u.namep / u.namep points to name
00004971 E8B5050000          <1>  call  namei
                             <1>        ; jsr r0,namei / find the i-number associated
                             <1>                   ; / with the path name
                             <1>  ;jc   error
                             <1>        ; br error9 / not found
00004976 730F                <1>  jnc   short sysunlink1
                             <1>  ; 'file not found !' error
00004978 C705[CF740000]0C00- <1>  mov   dword [u.error], ERR_FILE_NOT_FOUND ; 12
00004980 0000                <1>
00004982 E9F5F6FFFF          <1>  jmp   error
                             <1> sysunlink1:
00004987 6650                <1>  push  ax
                             <1>        ; mov r1,-(sp) / put its i-number on the stack
00004989 E8A6FFFFFF          <1>  call  isdir
                             <1>        ; jsr r0,isdir / is it a directory
0000498E 6631C0              <1>  xor   ax, ax
00004991 66A3[B2740000]      <1>  mov   [u.dirbuf], ax ; 0
                             <1>        ; clr u.dirbuf / no, clear the location that will
                             <1>                 ; / get written into the i-number portion
                             <1>               ; / of the entry
00004997 832D[9C740000]0A    <1>  sub   dword [u.off], 10
                             <1>        ; sub $10.,u.off / move u.off back 1 directory entry
0000499E E86B000000          <1>  call  wdir
                             <1>        ; jsr r0,wdir / free the directory entry
000049A3 6658                <1>  pop   ax
                             <1>        ; mov (sp)+,r1 / get i-number back
000049A5 E8700D0000          <1>  call  iget
                             <1>        ; jsr r0,iget / get i-node
000049AA E87E0E0000          <1>  call  setimod
                             <1>        ; jsr r0,setimod / set modified flag
000049AF FE0D[58710000]      <1>  dec   byte [i.nlks]
                             <1>        ; decb i.nlks / decrement the number of links
000049B5 0F85E1F6FFFF        <1>  jnz   sysret
                             <1>        ; bgt sysret9 / if this was not the last link
                             <1>                  ; / to file return
                             <1>  ; AX = r1 = i-number
000049BB E88F090000          <1>  call  anyi
                             <1>        ; jsr r0,anyi / if it was, see if anyone has it open.
                             <1>               ; / Then free contents of file and destroy it.
000049C0 E9D7F6FFFF          <1>  jmp   sysret
                             <1>        ; br sysret9
                             <1> 
                             <1> mkdir:
                             <1>  ; 12/10/2015
                             <1>  ; 17/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 29/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'mkdir' makes a directory entry from the name pointed to
                             <1>  ; by u.namep into the current directory.
                             <1>  ;
                             <1>  ; INPUTS ->
```

```
                              <1>  ;    u.namep - points to a file name
                              <1>  ;                that is about to be a directory entry.
                              <1>  ;    ii - current directory's i-number.
                              <1>  ; OUTPUTS ->
                              <1>  ;    u.dirbuf+2 - u.dirbuf+10 - contains file name.
                              <1>  ;    u.off - points to entry to be filled
                              <1>  ;          in the current directory
                              <1>  ;    u.base - points to start of u.dirbuf.
                              <1>  ;    r1 - contains i-number of current directory
                              <1>  ;
                              <1>  ; ((AX = R1)) output
                              <1>  ;
                              <1>  ;    (Retro UNIX Prototype : 11/11/2012, UNIXCOPY.ASM)
                              <1>        ;    ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
                              <1>  ;
                              <1>
                              <1>  ; 17/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
000049C5 31C0                 <1>  xor   eax, eax
000049C7 BF[B4740000]         <1>        mov     edi, u.dirbuf+2
000049CC 89FE                 <1>  mov   esi, edi
000049CE AB                   <1>  stosd
000049CF AB                   <1>  stosd
                              <1>        ; jsr r0,copyz; u.dirbuf+2; u.dirbuf+10. / clear this
000049D0 89F7                 <1>  mov   edi, esi ; offset to u.dirbuf
                              <1>  ; 12/10/2015 ([u.namep] -> ebp)
                              <1>  ;mov  ebp, [u.namep]
000049D2 E899060000           <1>  call  trans_addr_nmbp ; convert virtual address to physical
                              <1>        ; esi = physical address (page start + offset)
                              <1>        ; ecx = byte count in the page (1 - 4096)
                              <1>  ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
                              <1>        ; mov u.namep,r2 / r2 points to name of directory entry
                              <1>        ; mov $u.dirbuf+2,r3 / r3 points to u.dirbuf+2
                              <1> mkdir_1: ; 1:
000049D7 45                   <1>  inc   ebp ; 12/10/2015
                              <1>  ;
                              <1>  ; / put characters in the directory name in u.dirbuf+2 - u.dirbuf+10
                              <1>   ; 01/08/2013
000049D8 AC                   <1>  lodsb
                              <1>        ; movb (r2)+,r1 / move character in name to r1
000049D9 20C0                 <1>  and   al, al
000049DB 7427                 <1>  jz    short mkdir_3
                              <1>        ; beq 1f / if null, done
000049DD 3C2F                 <1>  cmp   al, '/'
                              <1>        ; cmp r1,$'/ / is it a "/"?
000049DF 7414                 <1>  je    short mkdir_err
                              <1>  ;je   error
                              <1>        ; beq error9 / yes, error
                              <1>  ; 12/10/2015
000049E1 6649                 <1>  dec   cx
000049E3 7505                 <1>  jnz   short mkdir_2
                              <1>  ; 12/10/2015 ([u.namep] -> ebp)
000049E5 E88C060000           <1>  call  trans_addr_nm ; convert virtual address to physical
                              <1>        ; esi = physical address (page start + offset)
                              <1>        ; ecx = byte count in the page
                              <1>  ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
                              <1> mkdir_2:
000049EA 81FF[BC740000]       <1>  cmp     edi, u.dirbuf+10
                              <1>        ; cmp r3,$u.dirbuf+10. / have we reached the last slot for
                              <1>                         ; / a char?
000049F0 74E5                 <1>  je    short mkdir_1
                              <1>        ; beq 1b / yes, go back
000049F2 AA                   <1>  stosb
                              <1>        ; movb r1,(r3)+ / no, put the char in the u.dirbuf
000049F3 EBE2                 <1>  jmp   short mkdir_1
                              <1>        ; br 1b / get next char
                              <1> mkdir_err:
                              <1>  ; 17/06/2015
000049F5 C705[CF740000]1300-  <1>  mov   dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
000049FD 0000                 <1>
000049FF E978F6FFFF           <1>  jmp   error
                              <1>
                              <1> mkdir_3: ; 1:
00004A04 A1[94740000]         <1>  mov   eax, [u.dirp]
00004A09 A3[9C740000]         <1>  mov   [u.off], eax
                              <1>        ; mov u.dirp,u.off / pointer to empty current directory
                              <1>                      ; / slot to u.off
                              <1> wdir: ; 29/04/2013
00004A0E C705[A0740000]-      <1>        mov     dword [u.base], u.dirbuf
00004A14 [B2740000]           <1>
                              <1>        ; mov $u.dirbuf,u.base / u.base points to created file name
00004A18 C705[A4740000]0A00-  <1>        mov     dword [u.count], 10
00004A20 0000                 <1>
                              <1>        ; mov $10.,u.count / u.count = 10
00004A22 66A1[62740000]       <1>  mov   ax, [ii]
                              <1>        ; mov ii,r1 / r1 has i-number of current directory
00004A28 B201                 <1>  mov   dl, 1 ; owner flag mask ; RETRO UNIX 8086 v1 modification !
00004A2A E8C60D0000           <1>  call  access
                              <1>        ; jsr r0,access; 1 / get i-node and set its file up
                              <1>                      ; / for writing
                              <1>  ; AX = i-number of current directory
                              <1>  ; 01/08/2013
00004A2F FE05[E1740000]       <1>  inc     byte [u.kcall] ; the caller is 'mkdir' sign
```

```
00004A35 E8B6100000           <1>  call  writei
                              <1>        ; jsr r0,writei / write into directory
00004A3A C3                   <1>  retn
                              <1>        ; rts r0
                              <1>
                              <1> sysexec:
                              <1>  ; 23/10/2015
                              <1>  ; 19/10/2015
                              <1>  ; 18/10/2015
                              <1>  ; 10/10/2015
                              <1>  ; 26/08/2015
                              <1>  ; 05/08/2015
                              <1>  ; 29/07/2015
                              <1>  ; 25/07/2015
                              <1>  ; 24/07/2015
                              <1>  ; 21/07/2015
                              <1>  ; 20/07/2015
                              <1>  ; 02/07/2015
                              <1>  ; 01/07/2015
                              <1>  ; 25/06/2015
                              <1>  ; 24/06/2015
                              <1>  ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
                              <1>  ;
                              <1>  ; 'sysexec' initiates execution of a file whose path name if
                              <1>  ; pointed to by 'name' in the sysexec call.
                              <1>  ; 'sysexec' performs the following operations:
                              <1>  ;    1. obtains i-number of file to be executed via 'namei'.
                              <1>  ;    2. obtains i-node of file to be exceuted via 'iget'.
                              <1>  ;    3. sets trap vectors to system routines.
                              <1>  ;    4. loads arguments to be passed to executing file into
                              <1>  ;     highest locations of user's core
                              <1>  ;    5. puts pointers to arguments in locations immediately
                              <1>  ;     following arguments.
                              <1>  ;    6.     saves number of arguments in next location.
                              <1>  ;    7. intializes user's stack area so that all registers
                              <1>  ;     will be zeroed and the PS is cleared and the PC set
                              <1>  ;     to core when 'sysret' restores registers
                              <1>  ;     and does an rti.
                              <1>  ;    8. inializes u.r0 and u.sp
                              <1>  ;    9. zeros user's core down to u.r0
                              <1>  ;   10.     reads executable file from storage device into core
                              <1>  ;     starting at location 'core'.
                              <1>  ;   11.     sets u.break to point to end of user's code with
                              <1>  ;     data area appended.
                              <1>  ;   12.     calls 'sysret' which returns control at location
                              <1>  ;     'core' via 'rti' instruction.
                              <1>  ;
                              <1>  ; Calling sequence:
                              <1>  ;     sysexec; namep; argp
                              <1>  ; Arguments:
                              <1>  ;     namep - points to pathname of file to be executed
                              <1>  ;     argp  - address of table of argument pointers
                              <1>  ;     argp1... argpn - table of argument pointers
                              <1>  ;     argp1:<...0> ... argpn:<...0> - argument strings
                              <1>  ; Inputs: (arguments)
                              <1>  ; Outputs: -
                              <1>  ; ...............................................................
                              <1>  ;
                              <1>  ; Retro UNIX 386 v1 modification:
                              <1>  ;     User application runs in it's own virtual space
                              <1>  ;     which is izolated from kernel memory (and other
                              <1>  ;     memory pages) via 80386 paging in ring 3
                              <1>  ;     privilige mode. Virtual start address is always 0.
                              <1>  ;     User's core memory starts at linear address 400000h
                              <1>  ;     (the end of the 1st 4MB).
                              <1>  ;
                              <1>  ; Retro UNIX 8086 v1 modification:
                              <1>  ;     user/application segment and system/kernel segment
                              <1>  ;     are different and sysenter/sysret/sysrele routines
                              <1>  ;     are different (user's registers are saved to
                              <1>  ;     and then restored from system's stack.)
                              <1>  ;
                              <1>  ;     NOTE: Retro UNIX 8086 v1 'arg2' routine gets these
                              <1>  ;           arguments which were in these registers;
                              <1>  ;           but, it returns by putting the 1st argument
                              <1>  ;           in 'u.namep' and the 2nd argument
                              <1>  ;           on top of stack. (1st argument is offset of the
                              <1>  ;           file/path name in the user's program segment.)
                              <1>
                              <1>  ;call arg2
                              <1>  ; * name - 'u.namep' points to address of file/path name
                              <1>  ;          in the user's program segment ('u.segmnt')
                              <1>  ;          with offset in BX register (as sysopen argument 1).
                              <1>  ; * argp - sysexec argument 2 is in CX register
                              <1>  ;          which is on top of stack.
                              <1>  ;
                              <1>        ; jsr r0,arg2 / arg0 in u.namep,arg1 on top of stack
                              <1>
                              <1>  ; 23/06/2015 (32 bit modifications)
                              <1>
00004A3B 891D[98740000]       <1>  mov   [u.namep], ebx ; argument 1
```

```
                             <1>         ; 18/10/2015
00004A41 890D[F8740000]      <1>  mov     [argv], ecx  ; * ; argument 2
00004A47 E8DF040000          <1>  call  namei
                             <1>        ; jsr r0,namei / namei returns i-number of file
                             <1>                    ; / named in sysexec call in r1
                             <1>  ;jc   error
                             <1>        ; br error9
00004A4C 731E                <1>  jnc   short sysexec_0
                             <1>  ;
                             <1>  ; 'file not found !' error
00004A4E C705[CF740000]0C00- <1>  mov   dword [u.error], ERR_FILE_NOT_FOUND
00004A56 0000                <1>
00004A58 E91FF6FFFF          <1>  jmp   error
                             <1> sysexec_not_exf:
                             <1>  ; 'not executable file !' error
00004A5D C705[CF740000]1600- <1>  mov   dword [u.error], ERR_NOT_EXECUTABLE
00004A65 0000                <1>
00004A67 E910F6FFFF          <1>  jmp   error
                             <1> sysexec_0:
00004A6C E8A90C0000          <1>  call  iget
                             <1>        ; jsr r0,iget / get i-node for file to be executed
00004A71 66F705[56710000]10- <1>         test    word [i.flgs], 10h
00004A79 00                  <1>
                             <1>        ; bit $20,i.flgs / is file executable
00004A7A 74E1                <1>  jz    short sysexec_not_exf
                             <1>  ;jz   error
                             <1>        ; beq error9
                             <1>  ;;
00004A7C E818140000          <1>  call  iopen
                             <1>        ; jsr r0,iopen / gets i-node for file with i-number
                             <1>                    ; / given in r1 (opens file)
                             <1>  ; AX = i-number of the file
00004A81 66F705[56710000]20- <1>  test  word [i.flgs], 20h
00004A89 00                  <1>
                             <1>        ; bit $40,i.flgs / test user id on execution bit
00004A8A 7415                <1>  jz    short sysexec_1
                             <1>        ; beq 1f
00004A8C 803D[C6740000]00    <1>  cmp   byte [u.uid], 0 ; 02/08/2013
                             <1>        ; tstb u.uid / test user id
00004A93 760C                <1>  jna   short sysexec_1
                             <1>        ; beq 1f / super user
00004A95 8A0D[59710000]      <1>  mov   cl, [i.uid]
00004A9B 880D[C6740000]      <1>  mov   [u.uid], cl ; 02/08/2013
                             <1>        ; movb i.uid,u.uid / put user id of owner of file
                             <1>                      ; / as process user id
                             <1> sysexec_1:
                             <1>  ; 18/10/2215
                             <1>  ; 10/10/2015
                             <1>  ; 24/07/2015
                             <1>  ; 21/07/2015
                             <1>  ; 25/06/2015
                             <1>  ; 24/06/2015
                             <1>         ; Moving arguments to the end of [u.upage]
                             <1>  ; (by regarding page borders in user's memory space)
                             <1>  ;
                             <1>  ; 10/10/2015
                             <1>  ; 21/07/2015
00004AA1 89E5                <1>  mov   ebp, esp ; (**)
                             <1>  ; 18/10/2015
00004AA3 89EF                <1>  mov   edi, ebp
00004AA5 B900010000          <1>  mov   ecx, MAX_ARG_LEN ; 256
                             <1>  ;sub  edi, MAX_ARG_LEN ; 256
00004AAA 29CF                <1>  sub   edi, ecx
00004AAC 89FC                <1>  mov   esp, edi
00004AAE 31C0                <1>  xor   eax, eax
00004AB0 A3[A8740000]        <1>  mov   [u.nread], eax ; 0
00004AB5 49                  <1>  dec   ecx ; 256 - 1
00004AB6 890D[A4740000]      <1>  mov   [u.count], ecx ; MAX_ARG_LEN - 1 ; 255
                             <1>  ;mov  dword [u.count], MAX_ARG_LEN - 1 ; 255
                             <1> sysexec_2:
00004ABC 8B35[F8740000]      <1>  mov   esi, [argv] ; 18/10/2015
00004AC2 E873020000          <1>  call  get_argp
00004AC7 B904000000          <1>  mov   ecx, 4 ; mov ecx, 4
                             <1> sysexec_3:
00004ACC 21C0                <1>  and   eax, eax
00004ACE 7456                <1>  jz    short sysexec_6
                             <1>  ; 18/10/2015
00004AD0 010D[F8740000]      <1>  add   [argv], ecx ; 4
00004AD6 66FF05[F6740000]    <1>  inc   word [argc]
                             <1>  ;
00004ADD A3[A0740000]        <1>  mov   [u.base], eax
                             <1>  ; 23/10/2015
00004AE2 66C705[DF740000]00- <1>  mov   word [u.pcount], 0
00004AEA 00                  <1>
                             <1> sysexec_4:
00004AEB E8BB110000          <1>  call  cpass ; get a character from user's core memory
00004AF0 750B                <1>         jnz      short sysexec_5
                             <1>        ; (max. 255 chars + null)
                             <1>  ; 18/10/2015
00004AF2 28C0                <1>  sub   al, al
00004AF4 AA                  <1>  stosb
00004AF5 FF05[A8740000]      <1>  inc   dword [u.nread]
```

```
00004AFB EB29                 <1>  jmp   short sysexec_6
                              <1> sysexec_5:
00004AFD AA                   <1>  stosb
00004AFE 20C0                 <1>  and   al, al
00004B00 75E9                 <1>  jnz   short sysexec_4
00004B02 B904000000           <1>  mov   ecx, 4
00004B07 390D[F4740000]       <1>  cmp   [ncount], ecx ; 4
00004B0D 72AD                 <1>  jb    short sysexec_2
00004B0F 8B35[F0740000]       <1>  mov   esi, [nbase]
00004B15 010D[F0740000]       <1>  add   [nbase], ecx ; 4
00004B1B 66290D[F4740000]     <1>  sub   [ncount], cx
00004B22 8B06                 <1>  mov   eax, [esi]
00004B24 EBA6                 <1>  jmp   short sysexec_3
                              <1> sysexec_6:
                              <1>  ; 18/10/2015
                              <1>  ; argument list transfer from user's core memory to
                              <1>  ; kernel stack frame is OK here.
                              <1>  ; [u.nread] = ; argument list length
                              <1>  ;mov  [argv], esp ; start address of argument list
                              <1>  ;
                              <1>  ; 18/10/2015
                              <1>  ; 24/07/2015
                              <1>         ; 21/07/2015
                              <1>  ; 02/07/2015
                              <1>  ; 25/06/2015
                              <1>  ; 24/06/2015
                              <1>  ; 23/06/2015
                              <1>  ;
00004B26 8B1D[D7740000]       <1>  mov   ebx, [u.ppgdir] ; parent's page directory
00004B2C 21DB                 <1>  and   ebx, ebx  ; /etc/init ? (u.ppgdir = 0)
00004B2E 740A                 <1>  jz    short sysexec_7
00004B30 A1[D3740000]         <1>  mov   eax, [u.pgdir] ; physical address of page directory
00004B35 E836E6FFFF           <1>  call  deallocate_page_dir
                              <1> sysexec_7:
00004B3A E866E5FFFF           <1>  call  make_page_dir
                              <1>  ;jc   short sysexec_14
00004B3F 0F82CDEDFFFF         <1>  jc    panic  ; allocation error
                              <1>               ; after a deallocation would be nonsence !?
                              <1>  ; 24/07/2015
                              <1>  ; map kernel pages (1st 4MB) to PDE 0
                              <1>  ;     of the user's page directory
                              <1>  ;     (It is needed for interrupts!)
                              <1>  ; 18/10/2015
00004B45 8B15[A8700000]       <1>  mov   edx, [k_page_dir] ; Kernel's page directory
00004B4B 8B02                 <1>  mov   eax, [edx] ; physical address of
                              <1>                 ; kernel's first page table (1st 4 MB)
                              <1>                 ; (PDE 0 of kernel's page directory)
00004B4D 8B15[D3740000]       <1>  mov   edx, [u.pgdir]
00004B53 8902                 <1>  mov   [edx], eax ; PDE 0 (1st 4MB)
                              <1>  ;
                              <1>  ; 20/07/2015
00004B55 BB00004000           <1>  mov   ebx, CORE ; start address = 0 (virtual) + CORE
                              <1>  ; 18/10/2015
00004B5A BE[E8740000]         <1>  mov   esi, pcore ; physical start address
                              <1> sysexec_8:
00004B5F B907000000           <1>  mov   ecx, PDE_A_USER + PDE_A_WRITE + PDE_A_PRESENT
00004B64 E85AE5FFFF           <1>  call  make_page_table
00004B69 0F82A3EDFFFF         <1>  jc    panic
                              <1>  ;mov  ecx, PTE_A_USER + PTE_A_WRITE + PTE_A_PRESENT
00004B6F E85DE5FFFF           <1>  call  make_page ; make new page, clear and set the pte
00004B74 0F8298EDFFFF         <1>  jc    panic
                              <1>  ;
00004B7A 8906                 <1>  mov   [esi], eax ; 24/06/2015
                              <1>  ; ebx = virtual address (24/07/2015)
00004B7C E873EAFFFF           <1>  call  add_to_swap_queue
                              <1>  ; 18/10/2015
00004B81 81FE[EC740000]       <1>  cmp   esi, ecore ; user's stack (last) page ?
00004B87 740C                 <1>  je    short sysexec_9 ; yes
00004B89 BE[EC740000]         <1>  mov   esi, ecore  ; physical address of the last page
                              <1>  ; 20/07/2015
00004B8E BB00F0FFFF           <1>  mov   ebx, (ECORE - PAGE_SIZE) + CORE
                              <1>  ; ebx = virtual end address + segment base address - 4K
00004B93 EBCA                 <1>         jmp     short sysexec_8
                              <1>
                              <1> sysexec_9:
                              <1>  ; 18/10/2015
                              <1>  ; 26/08/2015
                              <1>  ; 25/06/2015
                              <1>  ; move arguments from kernel stack to [ecore]
                              <1>  ; (argument list/line will be copied from kernel stack
                              <1>  ; frame to the last (stack) page of user's core memory)
                              <1>  ; 18/10/2015
00004B95 8B3D[EC740000]       <1>  mov   edi, [ecore]
00004B9B 81C700100000         <1>  add   edi, PAGE_SIZE
00004BA1 0FB705[F6740000]     <1>  movzx eax, word [argc]
00004BA8 09C0                 <1>  or    eax, eax
00004BAA 7509                 <1>  jnz   short sysexec_10
00004BAC 89FB                 <1>  mov   ebx, edi
00004BAE 83EB04               <1>  sub   ebx, 4
00004BB1 8903                 <1>  mov   [ebx], eax ; 0
00004BB3 EB40                 <1>  jmp   short sysexec_13
                              <1> sysexec_10:
```

```
00004BB5 8B0D[A8740000]      <1>  mov   ecx, [u.nread]
                             <1>  ;mov  esi, [argv}
00004BBB 89E6                <1>  mov   esi, esp ; start address of argument list
00004BBD 29CF                <1>  sub   edi, ecx ; page end address - argument list length
00004BBF 89C2                <1>  mov   edx, eax
00004BC1 FEC2                <1>  inc   dl ; argument count + 1 for argc value
00004BC3 C0E202              <1>  shl   dl, 2  ; 4 * (argument count + 1)
00004BC6 89FB                <1>  mov   ebx, edi
00004BC8 80E3FC              <1>  and   bl, 0FCh ; 32 bit (dword) alignment
00004BCB 29D3                <1>  sub   ebx, edx
00004BCD 89FA                <1>  mov   edx, edi
00004BCF F3A4                <1>  rep   movsb
00004BD1 89D6                <1>  mov   esi, edx
00004BD3 89DF                <1>  mov   edi, ebx
00004BD5 BA00F0BFFF          <1>  mov   edx, ECORE - PAGE_SIZE ; virtual addr. of the last page
00004BDA 2B15[EC740000]      <1>  sub   edx, [ecore] ; difference (virtual - physical)
00004BE0 AB                  <1>  stosd ; eax = argument count
                             <1> sysexec_11:
00004BE1 89F0                <1>  mov   eax, esi
00004BE3 01D0                <1>  add   eax, edx
00004BE5 AB                  <1>  stosd  ; eax = virtual address
00004BE6 FE0D[F6740000]      <1>  dec   byte [argc]
00004BEC 7407                <1>  jz    short sysexec_13
                             <1> sysexec_12:
00004BEE AC                  <1>  lodsb
00004BEF 20C0                <1>  and   al, al
00004BF1 75FB                <1>  jnz   short sysexec_12
00004BF3 EBEC                <1>  jmp   short sysexec_11
                             <1>  ;
                             <1>  ; 1:
                             <1>        ; mov (sp)+,r5 / r5 now contains address of list of
                             <1>                   ; / pointers to arguments to be passed
                             <1>        ; mov $1,u.quit / u.quit determines handling of quits;
                             <1>                    ; / u.quit = 1 take quit
                             <1>        ; mov $1,u.intr / u.intr determines handling of
                             <1>                   ; / interrupts; u.intr = 1 take interrupt
                             <1>        ; mov $rtssym,30 / emt trap vector set to take
                             <1>                     ; / system routine
                             <1>        ; mov $fpsym,*10 / reserved instruction trap vector
                             <1>                     ; / set to take system routine
                             <1>        ; mov $sstack,sp / stack space used during swapping
                             <1>        ; mov r5,-(sp) / save arguments pointer on stack
                             <1>        ; mov $ecore,r5 / r5 has end of core
                             <1>        ; mov $core,r4 / r4 has start of users core
                             <1>        ; mov r4,u.base / u.base has start of users core
                             <1>        ; mov (sp),r2 / move arguments list pointer into r2
                             <1>  ; 1:
                             <1>        ; tst (r2)+ / argument char = "nul"
                             <1>        ; bne 1b
                             <1>        ; tst -(r2) / decrement r2 by 2; r2 has addr of
                             <1>                ; / end of argument pointer list
                             <1>  ; 1:
                             <1>       ; / move arguments to bottom of users core
                             <1>        ; mov -(r2),r3 / (r3) last non zero argument ptr
                             <1>        ; cmp r2,(sp) / is r2 = beginning of argument
                             <1>                  ; / ptr list
                             <1>        ; blo 1f / branch to 1f when all arguments
                             <1>               ; / are moved
                             <1>        ; mov -(r2),r3 / (r3) last non zero argument ptr
                             <1>  ; 2:
                             <1>        ; tstb (r3)+
                             <1>        ; bne 2b / scan argument for \0 (nul)
                             <1>
                             <1>  ; 2:
                             <1>        ; movb -(r3),-(r5) / move argument char
                             <1>                     ; / by char starting at "ecore"
                             <1>        ; cmp r3,(r2) / moved all characters in
                             <1>                  ; / this argument
                             <1>        ; bhi 2b / branch 2b if not
                             <1>        ; mov r5,(r4)+ / move r5 into top of users core;
                             <1>                   ; / r5 has pointer to nth arg
                             <1>        ; br 1b / string
                             <1>  ; 1:
                             <1>        ; clrb -(r5)
                             <1>        ; bic $1,r5 / make r5 even, r5 points to
                             <1>              ; / last word of argument strings
                             <1>        ; mov $core,r2
                             <1>
                             <1>  ; 1: / move argument pointers into core following
                             <1>        ; / argument strings
                             <1>        ; cmp r2,r4
                             <1>        ; bhis 1f / branch to 1f when all pointers
                             <1>              ; / are moved
                             <1>        ; mov (r2)+,-(r5)
                             <1>        ; br 1b
                             <1>  ; 1:
                             <1>        ; sub $core,r4 / gives number of arguments *2
                             <1>        ; asr r4 / divide r4 by 2 to calculate
                             <1>               ; / the number of args stored
                             <1>        ; mov r4,-(r5) / save number of arguments ahead
                             <1>                   ; / of the argument pointers
                             <1> sysexec_13:
```

```
                             <1>  ; 19/10/2015
                             <1>  ; 18/10/2015
                             <1>  ; 29/07/2015
                             <1>  ; 25/07/2015
                             <1>  ; 24/07/2015
                             <1>  ; 20/07/2015
                             <1>  ; 25/06/2015
                             <1>  ; 24/06/2015
                             <1>  ; 23/06/2015
                             <1>  ;
                             <1>  ; moving arguments to [ecore] is OK here..
                             <1>  ; 18/10/2015
00004BF5 89EC                <1>  mov   esp, ebp ; (**) restore kernel stack pointer
                             <1>  ; ebx = beginning addres of argument list pointers
                             <1>  ;     in user's stack
                             <1>  ; 19/10/2015
00004BF7 2B1D[EC740000]      <1>  sub   ebx, [ecore]
00004BFD 81C300F0BFFF        <1>  add     ebx, (ECORE - PAGE_SIZE)
                             <1>              ; end of core - 4096 (last page)
                             <1>              ; (virtual address)
00004C03 891D[F8740000]      <1>  mov   [argv], ebx
00004C09 891D[AC740000]      <1>  mov   [u.break], ebx ; available user memory
                             <1>  ;
00004C0F 29C0                <1>  sub   eax, eax
00004C11 C705[A4740000]2000- <1>  mov   dword [u.count], 32 ; Executable file header size
00004C19 0000                <1>
                             <1>        ; mov $14,u.count
00004C1B C705[90740000]-     <1>  mov   dword [u.fofp], u.off
00004C21 [9C740000]          <1>
                             <1>        ; mov $u.off,u.fofp
00004C25 A3[9C740000]        <1>  mov   [u.off], eax ; 0
                             <1>        ; clr u.off / set offset in file to be read to zero
                             <1>  ; 25/07/2015
00004C2A A3[A0740000]        <1>  mov   [u.base], eax ; 0, start of user's core (virtual)
                             <1>  ; 25/06/2015
00004C2F 66A1[62740000]      <1>  mov   ax, [ii]
                             <1>  ; AX = i-number of the executable file
00004C35 E8C00C0000          <1>  call  readi
                             <1>        ; jsr r0,readi / read in first six words of
                             <1>              ; / user's file, starting at $core
                             <1>        ; mov sp,r5 / put users stack address in r5
                             <1>        ; sub $core+40.,r5 / subtract $core +40,
                             <1>                    ; / from r5 (leaves number of words
                             <1>                    ; / less 26 available for
                             <1>                    ; / program in user core
                             <1>        ; mov r5,u.count /
                             <1>  ; 25/06/2015
00004C3A 8B0D[AC740000]      <1>  mov   ecx, [u.break] ; top of user's stack (physical addr.)
00004C40 890D[A4740000]      <1>  mov   [u.count], ecx ; save for overrun check
                             <1>  ;
00004C46 8B0D[A8740000]      <1>  mov   ecx, [u.nread]
00004C4C 890D[AC740000]      <1>  mov   [u.break], ecx ; virtual address (offset from start)
00004C52 80F920              <1>  cmp   cl, 32
00004C55 7540                <1>        jne    short sysexec_15
                             <1>  ;:
                             <1>  ; 25/06/2015
                             <1>  ; Retro UNIX 386 v1 (32 bit) executable file header format
                             <1>  ; 18/10/2015
00004C57 8B35[E8740000]      <1>  mov   esi, [pcore] ; start address of user's core memory
                             <1>                     ; (phys. start addr. of the exec. file)
00004C5D AD                  <1>  lodsd
00004C5E 663DEB1E            <1>  cmp   ax, 1EEBh ; EBH, 1Eh -> jump to +32
00004C62 7533                <1>  jne   short sysexec_15
                             <1>        ; cmp core,$405 / br .+14 is first instruction
                             <1>                    ; / if file is standard a.out format
                             <1>        ; bne 1f / branch, if not standard format
00004C64 AD                  <1>  lodsd
00004C65 89C1                <1>  mov   ecx, eax ; text (code) section size
00004C67 AD                  <1>  lodsd
00004C68 01C1                <1>  add   ecx, eax ; + data section size (initialized data)
                             <1>        ; mov core+2,r5 / put 2nd word of users program in r5;
                             <1>                      ; / number of bytes in program text
                             <1>        ; sub $14,r5 / subtract 12
00004C6A 89CB                <1>  mov   ebx, ecx
                             <1>  ;
                             <1>  ; 25/06/2015
                             <1>  ; NOTE: These are for next versions of Retro UNIX 386
                             <1>  ;     and SINGLIX operating systems (as code template).
                             <1>  ;     Current Retro UNIX 386 v1 files can be max. 64KB
                             <1>  ;     due to RUFS (floppy disk file system) restriction...
                             <1>  ;     Overrun is not possible for current version.
                             <1>  ;
00004C6C AD                  <1>  lodsd
00004C6D 01C3                <1>  add   ebx, eax ; + bss section size (for overrun checking)
00004C6F 3B1D[A4740000]      <1>  cmp   ebx, [u.count]
00004C75 7711                <1>  ja    short sysexec_14  ; program overruns stack !
                             <1>  ;
                             <1>  ; 24/07/2015
                             <1>  ; add bss section size to [u.break]
00004C77 0105[AC740000]      <1>  add   [u.break], eax
                             <1>  ;
00004C7D 83E920              <1>  sub   ecx, 32  ; header size (already loaded)
```

```
                             <1>  ;cmp  ecx, [u.count]
                             <1>  ;jnb  short sysexec_16
                             <1>        ; cmp r5,u.count /
                             <1>        ; bgt 1f / branch if r5 greater than u.count
00004C80 890D[A4740000]      <1>  mov   [u.count], ecx ; required read count
                             <1>        ; mov r5,u.count
                             <1>  ;
00004C86 EB2A                <1>  jmp   short sysexec_16
                             <1>  ;
                             <1> sysexec_14:
                             <1>  ; 23/06/2015
                             <1>  ; insufficient (out of) memory
00004C88 C705[CF740000]0100- <1>  mov   dword [u.error], ERR_MINOR_IM ; 1
00004C90 0000                <1>
00004C92 E9E5F3FFFF          <1>  jmp   error
                             <1>  ;
                             <1> sysexec_15:
                             <1>  ; 25/06/2015
00004C97 0FB715[5A710000]    <1>         movzx   edx, word [i.size] ; file size
00004C9E 29CA                <1>  sub   edx, ecx ; file size - loaded bytes
00004CA0 7627                <1>  jna   short sysexec_17 ; no need to next read
00004CA2 01D1                <1>  add   ecx, edx ; [i.size]
00004CA4 3B0D[A4740000]      <1>  cmp   ecx, [u.count] ; overrun check (!)
00004CAA 77DC                <1>  ja    short sysexec_14
00004CAC 8915[A4740000]      <1>  mov   [u.count], edx
                             <1> sysexec_16:
00004CB2 66A1[62740000]      <1>  mov   ax, [ii] ; i-number
00004CB8 E83D0C0000          <1>  call  readi
                             <1>        ; add core+10,u.nread / add size of user data area
                             <1>                              ; / to u.nread
                             <1>        ; br 2f
                             <1>  ; 1:
                             <1>        ; jsr r0,readi / read in rest of file
                             <1>  ; 2:
00004CBD 8B0D[A8740000]      <1>  mov   ecx, [u.nread]
00004CC3 010D[AC740000]      <1>  add   [u.break], ecx
                             <1>        ; mov u.nread,u.break / set users program break to end of
                             <1>                          ; / user code
                             <1>        ; add $core+14,u.break / plus data area
                             <1> sysexec_17: ; 20/07/2015
                             <1>  ;mov  ax, [ii] ;rgc i-number
00004CC9 E8F8120000          <1>  call  iclose
                             <1>        ; jsr r0,iclose / does nothing
00004CCE 31C0                <1>         xor     eax, eax
00004CD0 FEC0                <1>  inc   al
00004CD2 66A3[BE740000]      <1>  mov   [u.intr], ax ; 1 (interrupt/time-out is enabled)
00004CD8 66A3[C0740000]      <1>  mov   [u.quit], ax ; 1 ('crtl+brk' signal is enabled)
                             <1>  ; 02/07/2015
00004CDE 833D[D7740000]00    <1>         cmp  dword [u.ppgdir], 0  ; is the caller sys_init (kernel) ?
00004CE5 770C                <1>  ja    short sysexec_18 ; no, the caller is user process
                             <1>  ; If the caller is kernel (sys_init), 'sysexec' will come here
00004CE7 8B15[A8700000]      <1>  mov   edx, [k_page_dir] ; kernel's page directory
00004CED 8915[D7740000]      <1>  mov   [u.ppgdir], edx ; next time 'sysexec' must not come here
                             <1> sysexec_18:
                             <1>  ; 18/10/2015
                             <1>  ; 05/08/2015
                             <1>  ; 29/07/2015
00004CF3 8B2D[F8740000]      <1>  mov   ebp, [argv] ; user's stack pointer must point to argument
                             <1>                    ; list pointers (argument count)
00004CF9 FA                  <1>  cli
00004CFA 8B25[44700000]      <1>         mov     esp, [tss.esp0]  ; ring 0 (kernel) stack pointer
                             <1>  ;mov        esp, [u.sp] ; Restore Kernel stack
                             <1>                    ; for this process
                             <1>  ;add  esp, 20 ; --> EIP, CS, EFLAGS, ESP, SS
                             <1>  ;xor  eax, eax ; 0
00004D00 FEC8                <1>  dec   al ; eax = 0
00004D02 66BA2300            <1>  mov   dx, UDATA
00004D06 6652                <1>  push  dx  ; user's stack segment
00004D08 55                  <1>  push  ebp ; user's stack pointer
                             <1>              ; (points to number of arguments)
00004D09 FB                  <1>  sti
00004D0A 9C                  <1>  pushfd      ; EFLAGS
                             <1>        ; Set IF for enabling interrupts in user mode
                             <1>  ;or   dword [esp], 200h
                             <1>  ;
                             <1>  ;mov  bx, UCODE
                             <1>  ;push bx ; user's code segment
00004D0B 6A1B                <1>  push  UCODE
                             <1>  ;push 0
00004D0D 50                  <1>  push  eax ; EIP (=0) - start address -
                             <1>        ; clr -(r5) / popped into ps when rti in
                             <1>                ; / sysrele is executed
                             <1>        ; mov $core,-(r5) / popped into pc when rti
                             <1>                        ; / in sysrele is executed
                             <1>        ;mov r5,0f / load second copyz argument
                             <1>        ;tst -(r5) / decrement r5
00004D0E 8925[78740000]      <1>  mov   [u.sp], esp ; 29/07/2015
                             <1>  ; 05/08/2015
                             <1>  ; Remedy of a General Protection Fault during 'iretd' is here !
                             <1>  ; ('push dx' would cause to general protection fault,
                             <1>  ; after 'pop ds' etc.)
                             <1>  ;
                             <1>  ;; push dx ; ds (UDATA)
                             <1>  ;; push dx ; es (UDATA)
                             <1>  ;; push dx ; fs (UDATA)
                             <1>  ;; push dx ; gs (UDATA)
                             <1>  ;
                             <1>  ; This is a trick to prevent general protection fault
                             <1>  ; during 'iretd' intruction at the end of 'sysrele' (in u1.s):
00004D14 8EC2                <1>  mov   es, dx ; UDATA
00004D16 06                  <1>  push  es ; ds (UDATA)
00004D17 06                  <1>  push  es ; es (UDATA)
00004D18 06                  <1>  push  es ; fs (UDATA)
00004D19 06                  <1>  push  es ; gs (UDATA)
00004D1A 66BA1000            <1>  mov   dx, KDATA
00004D1E 8EC2                <1>  mov   es, dx
                             <1>  ;
                             <1>  ;; pushad simulation
00004D20 89E5                <1>  mov   ebp, esp ; esp before pushad
00004D22 50                  <1>  push  eax ; eax (0)
00004D23 50                  <1>  push  eax ; ecx (0)
00004D24 50                  <1>  push  eax ; edx (0)
00004D25 50                  <1>  push  eax ; ebx (0)
00004D26 55                  <1>  push  ebp ; esp before pushad
00004D27 50                  <1>  push  eax ; ebp (0)
00004D28 50                  <1>  push  eax ; esi (0)
00004D29 50                  <1>  push  eax ; edi (0)
                             <1>  ;
00004D2A A3[80740000]        <1>  mov   [u.r0], eax ; eax = 0
00004D2F 8925[7C740000]      <1>  mov   [u.usp], esp
                             <1>        ; mov r5,u.r0 /
                             <1>        ; sub $16.,r5 / skip 8 words
                             <1>        ; mov r5,u.sp / assign user stack pointer value,
                             <1>        ;             / effectively zeroes all regs
                             <1>                 ; / when sysrele is executed
                             <1>        ; jsr r0,copyz; core; 0:0 / zero user's core
                             <1>        ; clr u.break
                             <1>        ; mov r5,sp / point sp to user's stack
                             <1>  ;
00004D35 E965F3FFFF          <1>  jmp   sysret0
                             <1>  ;jmp  sysret
                             <1>        ; br sysret3 / return to core image at $core
                             <1>
                             <1> get_argp:
                             <1>  ; 18/10/2015 (nbase, ncount)
                             <1>  ; 21/07/2015
                             <1>  ; 24/06/2015 (Retro UNIX 386 v1)
                             <1>  ; Get (virtual) address of argument from user's core memory
                             <1>  ;
                             <1>  ; INPUT:
                             <1>  ;     esi = virtual address of argument pointer
                             <1>  ; OUTPUT:
                             <1>  ;     eax = virtual address of argument
                             <1>  ;
                             <1>  ; Modified registers: EAX, EBX, ECX, EDX, ESI
                             <1>  ;
00004D3A 833D[D7740000]00    <1>  cmp     dword [u.ppgdir], 0 ; /etc/init ?
                             <1>                         ; (the caller is kernel)
00004D41 7667                <1>         jna     short get_argpk
                             <1>  ;
00004D43 89F3                <1>        mov   ebx, esi
00004D45 E880E9FFFF          <1>  call  get_physical_addr ; get physical address
00004D4A 0F8289000000        <1>         jc      get_argp_err
00004D50 A3[F0740000]        <1>  mov   [nbase], eax ; physical address
00004D55 66890D[F4740000]    <1>  mov   [ncount], cx ; remain byte count in page (1-4096)
00004D5C B804000000          <1>  mov   eax, 4 ; 21/07/2015
00004D61 6639C1              <1>  cmp   cx, ax ; 4
00004D64 735D                <1>  jnb   short get_argp2
00004D66 89F3                <1>  mov   ebx, esi
00004D68 01CB                <1>  add   ebx, ecx
00004D6A E85BE9FFFF          <1>  call  get_physical_addr ; get physical address
00004D6F 7268                <1>  jc    short get_argp_err
                             <1>  ;push esi
00004D71 89C6                <1>  mov   esi, eax
00004D73 66870D[F4740000]    <1>  xchg  cx, [ncount]
00004D7A 8735[F0740000]      <1>  xchg  esi, [nbase]
00004D80 B504                <1>  mov   ch, 4
00004D82 28CD                <1>  sub   ch, cl
                             <1> get_argp0:
00004D84 AC                  <1>  lodsb
00004D85 6650                <1>  push  ax
00004D87 FEC9                <1>  dec   cl
00004D89 75F9                <1>         jnz     short get_argp0
00004D8B 8B35[F0740000]      <1>  mov   esi, [nbase]
                             <1>  ; 21/07/2015
00004D91 0FB6C5              <1>  movzx eax, ch
00004D94 0105[F0740000]      <1>  add   [nbase], eax
00004D9A 662905[F4740000]    <1>  sub   [ncount], ax
                             <1> get_argp1:
00004DA1 AC                  <1>  lodsb
00004DA2 FECD                <1>  dec   ch
00004DA4 743D                <1>         jz      short get_argp3
00004DA6 6650                <1>         push ax
00004DA8 EBF7                <1>  jmp     short get_argp1
```

```
                             <1> get_argpk:
                             <1>  ; Argument is in kernel's memory space
00004DAA 66C705[F4740000]00- <1>  mov   word [ncount], PAGE_SIZE ; 4096
00004DB2 10                  <1>
00004DB3 8935[F0740000]      <1>  mov   [nbase], esi
00004DB9 8305[F0740000]04    <1>  add   dword [nbase], 4
00004DC0 8B06                <1>  mov   eax, [esi] ; virtual addr. = physcal addr.
00004DC2 C3                  <1>  retn
                             <1> get_argp2:
                             <1>  ; 21/07/2015
                             <1>  ;mov  eax, 4
00004DC3 8B15[F0740000]      <1>  mov   edx, [nbase] ; 18/10/2015
00004DC9 0105[F0740000]      <1>  add   [nbase], eax
00004DCF 662905[F4740000]    <1>  sub   [ncount], ax
                             <1>  ;
00004DD6 8B02                <1>  mov   eax, [edx]
00004DD8 C3                  <1>  retn
                             <1> get_argp_err:
00004DD9 A3[CF740000]        <1>  mov   [u.error], eax
00004DDE E999F2FFFF          <1>  jmp   error
                             <1> get_argp3:
00004DE3 B103                <1>  mov   cl, 3
                             <1> get_argp4:
00004DE5 C1E008              <1>  shl   eax, 8
00004DE8 665A                <1>  pop   dx
00004DEA 88D0                <1>  mov   al, dl
00004DEC E2F7                <1>         loop    get_argp4
                             <1>  ;pop  esi
00004DEE C3                  <1>  retn
                             <1>
                             <1> sysfstat:
                             <1>  ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 19/06/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysfstat' is identical to 'sysstat' except that it operates
                             <1>  ; on open files instead of files given by name. It puts the
                             <1>  ; buffer address on the stack, gets the i-number and
                             <1>  ; checks to see if the file is open for reading or writing.
                             <1>  ; If the file is open for writing (i-number is negative)
                             <1>  ; the i-number is set positive and a branch into 'sysstat'
                             <1>  ; is made.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysfstat; buf
                             <1>  ; Arguments:
                             <1>  ;     buf - buffer address
                             <1>  ;
                             <1>  ; Inputs: *u.r0 - file descriptor
                             <1>  ; Outputs: buffer is loaded with file information
                             <1>  ; ...........................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       'sysfstat' system call has two arguments; so,
                             <1>  ;     * 1st argument, file descriptor is in BX register
                             <1>  ;     * 2nd argument, buf is pointed to by CX register
                             <1>
                             <1>  ; / set status of open file
                             <1>        ; jsr r0,arg; u.off / put buffer address in u.off
00004DEF 51                  <1>  push  ecx
                             <1>        ; mov u.off,-(sp) / put buffer address on the stack
                             <1>        ; mov *u.r0,r1 / put file descriptor in r1
                             <1>        ; jsr r0,getf / get the files i-number
                             <1>  ; BX = file descriptor (file number)
00004DF0 E8FF000000          <1>  call  getf1
00004DF5 6621C0              <1>  and   ax, ax ; i-number of the file
                             <1>        ; tst r1 / is it 0?
                             <1>  ;jz   error
                             <1>        ; beq error3 / yes, error
00004DF8 750F                <1>  jnz   short sysfstat1
00004DFA C705[CF740000]0A00- <1>  mov   dword [u.error], ERR_FILE_NOT_OPEN  ; 'file not open !'
00004E02 0000                <1>
00004E04 E973F2FFFF          <1>  jmp   error
                             <1> sysfstat1:
00004E09 80FC80              <1>  cmp   ah, 80h
00004E0C 7223                <1>         jb      short sysstat1
                             <1>        ; bgt 1f / if i-number is negative (open for writing)
00004E0E 66F7D8              <1>  neg   ax
                             <1>        ; neg r1 / make it positive, then branch
00004E11 EB1E                <1>  jmp   short sysstat1
                             <1>        ; br 1f / to 1f
                             <1> sysstat:
                             <1>  ; 18/10/2015
                             <1>  ; 07/10/2015
                             <1>  ; 02/09/2015
                             <1>  ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 19/06/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysstat' gets the status of a file. Its arguments are the
                             <1>  ; name of the file and buffer address. The buffer is 34 bytes
                             <1>  ; long and information about the file placed in it.
                             <1>  ; sysstat calls 'namei' to get the i-number of the file.
                             <1>  ; Then 'iget' is called to get i-node in core. The buffer
                             <1>  ; is then loaded and the results are given in the UNIX
                             <1>  ; Programmers Manual sysstat (II).
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysstat; name; buf
                             <1>  ; Arguments:
                             <1>  ;     name - points to the name of the file
                             <1>  ;     buf - address of a 34 bytes buffer
                             <1>  ; Inputs: -
                             <1>  ; Outputs: buffer is loaded with file information
                             <1>  ; ..............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       'sysstat' system call has two arguments; so,
                             <1>  ;     Retro UNIX 8086 v1 argument transfer method 2 is used
                             <1>  ;     to get sysstat system call arguments from the user;
                             <1>  ;     * 1st argument, name is pointed to by BX register
                             <1>  ;     * 2nd argument, buf is pointed to by CX register
                             <1>  ;
                             <1>  ;     NOTE: Retro UNIX 8086 v1 'arg2' routine gets these
                             <1>  ;           arguments which were in these registers;
                             <1>  ;           but, it returns by putting the 1st argument
                             <1>  ;           in 'u.namep' and the 2nd argument
                             <1>  ;           on top of stack. (1st argument is offset of the
                             <1>  ;           file/path name in the user's program segment.)
                             <1>
                             <1>  ; / ; name of file; buffer - get files status
                             <1>        ; jsr r0,arg2 / get the 2 arguments
00004E13 891D[98740000]      <1>  mov   [u.namep], ebx
00004E19 51                  <1>  push  ecx
00004E1A E80C010000          <1>  call  namei
                             <1>        ; jsr r0,namei / get the i-number for the file
                             <1>  ;jc   error
                             <1>        ; br error3 / no such file, error
00004E1F 7310                <1>  jnc   short sysstat1
                             <1>  ; pop       ecx
                             <1> sysstat_err0:
                             <1>  ; 'file not found !' error
00004E21 C705[CF740000]0C00- <1>  mov   dword [u.error], ERR_FILE_NOT_FOUND ; 12
00004E29 0000                <1>
00004E2B E94CF2FFFF          <1>  jmp   error
                             <1>
00004E30 00                  <1> statx: db 0
                             <1>
                             <1> sysstat1: ; 1:
00004E31 E8E4080000          <1>  call  iget
                             <1>        ; jsr r0,iget / get the i-node into core
                             <1>  ; 07/10/2015 (ax = [ii], inode number)
                             <1>  ; 02/09/2015
00004E36 8F05[A0740000]      <1>  pop   dword [u.base]
                             <1>        ; mov (sp)+,r3 / move u.off to r3 (points to buffer)
00004E3C E858000000          <1>  call  sysstat_gpa ; get physical address
00004E41 730A                <1>  jnc   short sysstat2
                             <1> sysstat_err1:
00004E43 A3[CF740000]        <1>  mov   dword [u.error], eax ; error code
00004E48 E92FF2FFFF          <1>  jmp   error
                             <1> sysstat2:
00004E4D A0[62740000]        <1>  mov   al, [ii] ; 07/10/2015 (result of 'iget' call, above)
00004E52 AA                  <1>  stosb
00004E53 FF05[A0740000]      <1>  inc   dword [u.base]
00004E59 6649                <1>  dec   cx
00004E5B 7505                <1>  jnz   short sysstat3
00004E5D E837000000          <1>  call  sysstat_gpa
                             <1>  ;jc   short sysstat_err1
                             <1> sysstat3:
00004E62 A0[63740000]        <1>  mov   al, [ii+1] ; 07/10/2015 (result of 'iget' call, above)
00004E67 AA                  <1>  stosb
                             <1>        ; mov r1,(r3)+ / put i-number in 1st word of buffer
00004E68 FF05[A0740000]      <1>  inc   dword [u.base]
                             <1>  ;dec  word [u.pcount]
00004E6E 6649                <1>  dec   cx
00004E70 7505                <1>  jnz   short sysstat4
00004E72 E822000000          <1>  call  sysstat_gpa
                             <1>  ;jc   short sysstat_err1
                             <1> sysstat4:
00004E77 BE[56710000]        <1>  mov   esi, inode
                             <1>        ; mov $inode,r2 / r2 points to i-node
                             <1> sysstat5: ; 1:
00004E7C A4                  <1>  movsb
                             <1>        ; mov (r2)+,(r3)+ / move rest of i-node to buffer
00004E7D FF05[A0740000]      <1>  inc   dword [u.base]
                             <1>  ;dec  word [u.pcount]
00004E83 6649                <1>  dec   cx
00004E85 7505                <1>  jnz   short sysstat6
00004E87 E80D000000          <1>  call  sysstat_gpa
                             <1>  ;jc   short sysstat_err1
                             <1> sysstat6:
00004E8C 81FE[76710000]      <1>  cmp   esi, inode + 32
                             <1>        ; cmp r2,$inode+32 / done?
00004E92 75E8                <1>  jne   short sysstat5
                             <1>        ; bne 1b / no, go back
00004E94 E903F2FFFF          <1>  jmp   sysret
```

```
                             <1>        ; br sysret3 / return through sysret
                             <1>  ;
                             <1> sysstat_gpa: ; get physical address of file status buffer
                             <1>  ; 02/09/2015
00004E99 8B1D[A0740000]      <1>  mov   ebx, [u.base]
                             <1>  ; 07/10/2015
00004E9F E826E8FFFF          <1>  call  get_physical_addr ; get physical address
                             <1>  ;jc   short sysstat_gpa1
00004EA4 729D                <1>  jc    short sysstat_err1
                             <1>  ; 18/10/2015
00004EA6 89C7                <1>  mov   edi, eax ; physical address
                             <1>  ;mov  [u.pcount], cx ; remain bytes in page
                             <1> ;sysstat_gpa1:
00004EA8 C3                  <1>  retn
                             <1> 
                             <1> fclose:
                             <1>  ; 18/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ;            (32 bit offset pointer modification)
                             <1>  ; 19/04/2013 - 12/01/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Given the file descriptor (index to the u.fp list)
                             <1>  ; 'fclose' first gets the i-number of the file via 'getf'.
                             <1>  ; If i-node is active (i-number > 0) the entry in
                             <1>  ; u.fp list is cleared. If all the processes that opened
                             <1>  ; that file close it, then fsp etry is freed and the file
                             <1>  ; is closed. If not a return is taken.
                             <1>  ; If the file has been deleted while open, 'anyi' is called
                             <1>  ; to see anyone else has it open, i.e., see if it is appears
                             <1>  ; in another entry in the fsp table. Upon return from 'anyi'
                             <1>  ; a check is made to see if the file is special.
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - contains the file descriptor (value=0,1,2...)
                             <1>  ;    u.fp - list of entries in the fsp table
                             <1>  ;    fsp - table of entries (4 words/entry) of open files.
                             <1>  ; OUTPUTS ->
                             <1>  ;    r1 - contains the same file descriptor
                             <1>  ;    r2 - contains i-number
                             <1>  ;
                             <1>  ; ((AX = R1))
                             <1>  ; ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification : CF = 1
                             <1>  ;              if i-number of the file is 0. (error)
                             <1>  ;
00004EA9 0FB7D0              <1>  movzx edx, ax ; **
00004EAC 6650                <1>  push  ax ; ***
                             <1>        ; mov r1,-(sp) / put r1 on the stack (it contains
                             <1>                   ; / the index to u.fp list)
00004EAE E83F000000          <1>  call  getf
                             <1>        ; jsr r0,getf / r1 contains i-number,
                             <1>                  ; / cdev has device =, u.fofp
                             <1>                  ; / points to 3rd word of fsp entry
00004EB3 6683F801            <1>  cmp   ax, 1 ; r1
                             <1>        ; tst r1 / is i-number 0?
00004EB7 7236                <1>  jb    short fclose_2
                             <1>        ; beq 1f / yes, i-node not active so return
                             <1>        ; tst (r0)+ / no, jump over error return
00004EB9 89D3                <1>  mov   ebx, edx ; **
00004EBB 6689C2              <1>  mov   dx, ax ; *
                             <1>        ; mov r1,r2 / move i-number to r2 ;*
                             <1>        ; mov (sp),r1 / restore value of r1 from the stack
                             <1>                  ; / which is index to u.fp ; **
00004EBE C683[86740000]00    <1>  mov   byte [ebx+u.fp], 0
                             <1>        ; clrb u.fp(r1) / clear that entry in the u.fp list
00004EC5 8B1D[90740000]      <1>  mov   ebx, [u.fofp]
                             <1>        ; mov u.fofp,r1 / r1 points to 3rd word in fsp entry
                             <1> fclose_0:
00004ECB FE4B04              <1>  dec   byte [ebx+4] ; 18/06/2015
                             <1>        ; decb 2(r1) / decrement the number of processes
                             <1>                 ; / that have opened the file
00004ECE 791F                <1>  jns   short fclose_2 ; jump if not negative (jump if bit 7 is 0)
                             <1>        ; bge 1f / if all processes haven't closed the file, return
                             <1>  ;
00004ED0 6652                <1>  push  dx ;*
                             <1>        ; mov r2,-(sp) / put r2 on the stack (i-number)
00004ED2 6631C0              <1>  xor   ax, ax ; 0
00004ED5 668943FC            <1>  mov   [ebx-4], ax ; 0
                             <1>        ; clr -4(r1) / clear 1st word of fsp entry
00004ED9 8A4305              <1>  mov   al, [ebx+5] ; 18/06/2015
                             <1>        ; tstb      3(r1) / has this file been deleted
00004EDC 20C0                <1>  and   al, al
00004EDE 7408                <1>  jz    short fclose_1
                             <1>        ; beq 2f / no, branch
00004EE0 6689D0              <1>  mov   ax, dx ; *
                             <1>        ; mov r2,r1 / yes, put i-number back into r1
                             <1>  ; AX = inode number
00004EE3 E867040000          <1>  call  anyi
                             <1>        ; jsr r0,anyi / free all blocks related to i-number
                             <1>                  ; / check if file appears in fsp again
                             <1> fclose_1: ; 2:
00004EE8 6658                <1>  pop   ax ; *
```

```
                             <1>        ; mov (sp)+,r1 / put i-number back into r1
00004EEA E8D7100000          <1>  call  iclose ; close if it is special file
                             <1>        ; jsr r0,iclose / check to see if its a special file
                             <1> fclose_2: ; 1:
00004EEF 6658                <1>  pop   ax ; ***
                             <1>        ; mov (sp)+,r1 / put index to u.fp back into r1
00004EF1 C3                  <1>  retn
                             <1>        ; rts r0
                             <1>
                             <1> getf:  ; / get the device number and the i-number of an open file
                             <1>  ; 13/05/2015
                             <1>  ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 19/04/2013 - 18/11/2013 (Retro UNIX 8086 v1)
                             <1>  ;
00004EF2 89C3                <1>  mov   ebx, eax
                             <1> getf1: ;; Calling point from 'rw1' (23/05/2013)
00004EF4 83FB0A              <1>  cmp   ebx, 10
                             <1>        ; cmp r1,$10. / user limited to 10 open files
00004EF7 730A                <1>         jnb  short getf2 ; 13/05/2015
                             <1>  ;jnb     error
                             <1>        ; bhis error3 / u.fp is table of users open files,
                             <1>                 ; / index in fsp table
00004EF9 8A9B[86740000]      <1>  mov   bl, [ebx+u.fp]
                             <1>        ; movb      u.fp(r1),r1 / r1 contains number of entry
                             <1>                         ; / in fsp table
00004EFF 08DB                <1>  or    bl, bl
00004F01 7503                <1>  jnz   short getf3
                             <1>  ;jz   short getf4
                             <1>        ; beq 1f / if its zero return
                             <1> getf2:
                             <1>  ; 'File not open !' error (ax=0)
00004F03 29C0                <1>  sub   eax, eax
00004F05 C3                  <1>  retn
                             <1> getf3:
                             <1>  ; Retro UNIX 386 v1 modification ! (11/05/2015)
                             <1>  ;
                             <1>  ; 'fsp' table (10 bytes/entry)
                             <1>  ; bit 15                        bit 0
                             <1>  ; ---|-------------------------------------------
                             <1>  ; r/w|           i-number of open file
                             <1>  ; ---|-------------------------------------------
                             <1>  ;              device number
                             <1>  ; -----------------------------------------------
                             <1>  ; offset pointer, r/w pointer to file (bit 0-15)
                             <1>  ; -----------------------------------------------
                             <1>  ; offset pointer, r/w pointer to file (bit 16-31)
                             <1>  ; ----------------------|------------------------
                             <1>  ;  flag that says file  | number of processes
                             <1>  ;   has been deleted    | that have file open
                             <1>  ; ----------------------|------------------------
                             <1>  ;
00004F06 B80A000000          <1>  mov   eax, 10
00004F0B F6E3                <1>  mul   bl
00004F0D BB[50720000]        <1>  mov   ebx, fsp - 6 ; the 3rd word in the fsp entry
00004F12 01C3                <1>  add   ebx, eax
                             <1>        ; asl r1
                             <1>        ; asl r1 / multiply by 8 to get index into
                             <1>               ; / fsp table entry
                             <1>        ; asl r1
                             <1>        ; add $fsp-4,r1 / r1 is pointing at the 3rd word
                             <1>                     ; / in the fsp entry
00004F14 891D[90740000]      <1>  mov   [u.fofp], ebx
                             <1>        ; mov r1,u.fofp / save address of 3rd word
                             <1>                     ; / in fsp entry in u.fofp
00004F1A 4B                  <1>  dec   ebx
00004F1B 4B                  <1>  dec   ebx
00004F1C 668B03              <1>  mov   ax, [ebx]
                             <1>  ;mov  [cdev], al ; ;;Retro UNIX 8086 v1 !
00004F1F 66A3[66740000]      <1>  mov   [cdev], ax ; ;;in fact (!)
                             <1>                    ;;dev number is in 1 byte
                             <1>        ; mov -(r1),cdev / remove the device number  cdev
00004F25 4B                  <1>  dec   ebx
00004F26 4B                  <1>  dec   ebx
00004F27 668B03              <1>  mov   ax, [ebx]
                             <1>        ; mov -(r1),r1 / and the i-number  r1
                             <1> getf4: ; 1:
00004F2A C3                  <1>  retn
                             <1>        ; rts r0
                             <1>
                             <1> namei:
                             <1>  ; 18/10/2015 (nbase, ncount)
                             <1>  ; 12/10/2015
                             <1>  ; 21/08/2015
                             <1>  ; 18/07/2015
                             <1>  ; 02/07/2015
                             <1>  ; 17/06/2015
                             <1>  ; 16/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 24/04/2013 - 31/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'namei' takes a file path name and returns i-number of
                             <1>  ; the file in the current directory or the root directory
                             <1>  ; (if the first character of the pathname is '/').
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    u.namep - points to a file path name
                             <1>  ;    u.cdir - i-number of users directory
                             <1>  ;    u.cdev - device number on which user directory resides
                             <1>  ; OUTPUTS ->
                             <1>  ;    r1 - i-number of file
                             <1>  ;    cdev
                             <1>  ;    u.dirbuf - points to directory entry where a match
                             <1>  ;               occurs in the search for file path name.
                             <1>  ;             If no match u.dirb points to the end of
                             <1>  ;               the directory and r1 = i-number of the current
                             <1>  ;             directory.
                             <1>  ; ((AX = R1))
                             <1>  ;
                             <1>  ; (Retro UNIX Prototype : 07/10/2012 - 05/01/2013, UNIXCOPY.ASM)
                             <1>         ; ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))
                             <1>  ;
                             <1>
00004F2B 66A1[84740000]      <1>  mov   ax, [u.cdir]
                             <1>        ; mov u.cdir,r1 / put the i-number of current directory
                             <1>                    ; / in r1
00004F31 668B15[C4740000]    <1>  mov   dx, [u.cdrv]
00004F38 668915[66740000]    <1>  mov   [cdev], dx      ; NOTE: Retro UNIX 8086 v1
                             <1>                        ; device/drive number is in 1 byte,
                             <1>                        ; not in 1 word!
                             <1>        ; mov u.cdev,cdev / device number for users directory
                             <1>                    ; / into cdev
                             <1>  ; 12/10/2015
                             <1>  ; 16/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
                             <1>         ; convert virtual (pathname) addr to physical address
00004F3F E82C010000          <1>  call    trans_addr_nmbp ; 12/10/2015
                             <1>        ; esi = physical address of [u.namep]
                             <1>        ; ecx = byte count in the page
00004F44 803E2F              <1>  cmp   byte [esi], '/'
                             <1>        ; cmpb *u.namep,$'/ / is first char in file name a /
00004F47 751E                <1>  jne   short namei_1
                             <1>        ; bne 1f
00004F49 FF05[98740000]      <1>  inc   dword [u.namep]
                             <1>        ; inc u.namep / go to next char
00004F4F 6649                <1>  dec   cx ; remain byte count in the page
00004F51 7506                <1>  jnz   short namei_0
                             <1>  ; 12/10/2015
00004F53 E818010000          <1>  call  trans_addr_nmbp ; convert virtual address to physical
                             <1>        ; esi = physical address (page start + offset)
                             <1>        ; ecx = byte count in the page
00004F58 4E                  <1>  dec   esi
                             <1> namei_0:
00004F59 46                  <1>  inc   esi  ; go to next char
00004F5A 66A1[70740000]      <1>  mov   ax, [rootdir] ; 09/07/2013
                             <1>        ; mov rootdir,r1 / put i-number of rootdirectory in r1
00004F60 C605[66740000]00    <1>  mov   byte [cdev], 0
                             <1>        ; clr cdev / clear device number
                             <1> namei_1: ; 1:
00004F67 F606FF              <1>  test  byte [esi], 0FFh
00004F6A 74BE                <1>  jz    short getf4
                             <1>  ;jz      nig
                             <1>        ; tstb *u.namep / is the character in file name a nul
                             <1>        ; beq nig / yes, end of file name reached;
                             <1>              ; / branch to "nig"
                             <1> namei_2: ; 1:
                             <1>  ; 18/10/2015
00004F6C 8935[F0740000]      <1>  mov   [nbase], esi
00004F72 66890D[F4740000]    <1>  mov   [ncount], cx
                             <1>  ;
                             <1>  ;mov  dx, 2
00004F79 B202                <1>  mov   dl, 2 ; user flag (read, non-owner)
00004F7B E875080000          <1>  call  access
                             <1>        ; jsr r0,access; 2 / get i-node with i-number r1
                             <1>  ; 'access' will not return here if user has not "r" permission !
00004F80 66F705[56710000]00- <1>  test  word [i.flgs], 4000h
00004F88 40                  <1>
                             <1>        ; bit $40000,i.flgs / directory i-node?
00004F89 746A                <1>         jz     short namei_err
                             <1>        ; beq error3 / no, got an error
                             <1>  ; 16/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
00004F8B 31C0                <1>  xor   eax, eax
00004F8D A3[9C740000]        <1>  mov   [u.off], eax ; 0
00004F92 66A1[5A710000]      <1>  mov   ax, [i.size]
00004F98 A3[94740000]        <1>  mov   [u.dirp], eax
                             <1>        ; mov i.size,u.dirp / put size of directory in u.dirp
                             <1>        ; clr u.off / u.off is file offset used by user
00004F9D C705[90740000]-     <1>  mov   dword [u.fofp], u.off
00004FA3 [9C740000]          <1>
                             <1>        ; mov $u.off,u.fofp / u.fofp is a pointer to
                             <1>                      ; / the offset portion of fsp entry
                             <1> namei_3: ; 2:
00004FA7 C705[A0740000]-     <1>  mov   dword [u.base], u.dirbuf
00004FAD [B2740000]          <1>
                             <1>        ; mov $u.dirbuf,u.base / u.dirbuf holds a file name
                             <1>                        ; / copied from a directory
00004FB1 C705[A4740000]0A00- <1>  mov   dword [u.count], 10
```

```
00004FB9 0000                <1>
                             <1>        ; mov $10.,u.count / u.count is byte count
                             <1>                     ; / for reads and writes
00004FBB 66A1[62740000]      <1>  mov   ax, [ii]
                             <1>  ; 31/07/2013 ('namei_r') - 16/06/2015 ('u.kcall')
00004FC1 FE05[E1740000]      <1>  inc     byte [u.kcall] ; the caller is 'namei' sign
00004FC7 E82E090000          <1>        call  readi
                             <1>        ; jsr r0,readi / read 10. bytes of file
                             <1>              ; with i-number (r1); i.e. read a directory entry
00004FCC 8B0D[A8740000]      <1>  mov   ecx, [u.nread]
00004FD2 09C9                <1>  or    ecx, ecx
                             <1>          ; tst u.nread
00004FD4 741B                <1>  jz    short nib
                             <1>          ; ble nib / gives error return
                             <1>  ;
00004FD6 668B1D[B2740000]    <1>  mov   bx, [u.dirbuf]
00004FDD 6621DB              <1>  and   bx, bx
                             <1>          ; tst u.dirbuf /
00004FE0 7522                <1>  jnz   short namei_4
                             <1>          ; bne 3f / branch when active directory entry
                             <1>               ; / (i-node word in entry non zero)
00004FE2 A1[9C740000]        <1>  mov   eax, [u.off]
00004FE7 83E80A              <1>  sub   eax, 10
00004FEA A3[94740000]        <1>  mov   [u.dirp], eax
                             <1>          ; mov u.off,u.dirp
                             <1>          ; sub $10.,u.dirp
00004FEF EBB6                <1>  jmp   short namei_3
                             <1>          ; br 2b
                             <1>
                             <1>  ; 18/07/2013
                             <1> nib:
00004FF1 31C0                <1>  xor   eax, eax  ; xor ax, ax ; ax = 0 -> file not found
00004FF3 F9                  <1>  stc
                             <1> nig:
00004FF4 C3                  <1>  retn
                             <1>
                             <1> namei_err:
                             <1>  ; 16/06/2015
00004FF5 C705[CF740000]1300- <1>  mov   dword [u.error], ERR_NOT_DIR ; 'not a directory !' error
00004FFD 0000                <1>
00004FFF E978F0FFFF          <1>  jmp   error
                             <1>
                             <1> namei_4: ; 3:
                             <1>  ; 18/10/2015
                             <1>  ; 12/10/2015
                             <1>  ; 21/08/2015
                             <1>  ; 18/07/2015
00005004 8B2D[98740000]      <1>  mov   ebp, [u.namep]
                             <1>          ; mov u.namep,r2 / u.namep points into a file name string
0000500A BF[B4740000]        <1>  mov   edi, u.dirbuf + 2
                             <1>          ; mov $u.dirbuf+2,r3 / points to file name of directory entry
                             <1>  ; 18/10/2015
0000500F 8B35[F0740000]      <1>  mov   esi, [nbase]
00005015 668B0D[F4740000]    <1>  mov   cx, [ncount]
                             <1>  ;
0000501C 6621C9              <1>  and   cx, cx
0000501F 7505                <1>  jnz   short namei_5
                             <1>  ;
00005021 E850000000          <1>  call  trans_addr_nm ; convert virtual address to physical
                             <1>          ; esi = physical address (page start + offset)
                             <1>          ; ecx = byte count in the page
                             <1> namei_5: ; 3:
00005026 45                  <1>  inc   ebp ; 18/07/2015
00005027 AC                  <1>  lodsb   ; mov al, [esi] ; inc esi (al = r4)
                             <1>          ; movb (r2)+,r4 / move a character from u.namep string into r4
00005028 08C0                <1>  or    al, al
0000502A 741D                <1>  jz    short namei_7
                             <1>          ; beq 3f / if char is nul, then the last char in string
                             <1>              ; / has been moved
0000502C 3C2F                <1>  cmp   al, '/'
                             <1>          ; cmp r4,$'/ / is char a </>
0000502E 7419                <1>  je    short namei_7
                             <1>          ; beq 3f
                             <1>  ; 12/10/2015
00005030 6649                <1>  dec   cx ; remain byte count in the page
00005032 7505                <1>  jnz   short namei_6
00005034 E83D000000          <1>  call  trans_addr_nm ; convert virtual address to physical
                             <1>          ; esi = physical address (page start + offset)
                             <1>          ; ecx = byte count in the page
                             <1> namei_6:
00005039 81FF[BC740000]      <1>         cmp     edi, u.dirbuf + 10
                             <1>        ; cmp r3,$u.dirbuf+10. / have I checked
                             <1>                         ; / all 8 bytes of file name
0000503F 74E5                <1>  je    short namei_5
                             <1>          ; beq 3b
00005041 AE                  <1>  scasb
                             <1>          ; cmpb (r3)+,r4 / compare char in u.namep string to file name
                             <1>                    ; / char read from directory
00005042 74E2                <1>  je    short namei_5
                             <1>          ; beq 3b / branch if chars match
                             <1>
00005044 E95EFFFFFF          <1>         jmp    namei_3 ; 2b
```

```
                              <1>        ; br 2b / file names do not match go to next directory entry
                              <1> namei_7: ; 3:
00005049 81FF[BC740000]       <1>  cmp   edi, u.dirbuf + 10
                              <1>        ; cmp r3,$u.dirbuf+10. / if equal all 8 bytes were matched
0000504F 740A                 <1>  je    short namei_8
                              <1>        ; beq 3f
00005051 8A27                 <1>  mov   ah, [edi]
                              <1>  ;inc  edi
00005053 20E4                 <1>  and   ah, ah
                              <1>        ; tstb (r3)+ /
00005055 0F854CFFFFFF         <1>         jnz     namei_3
                              <1>        ; bne 2b
                              <1> namei_8: ; 3
0000505B 892D[98740000]       <1>  mov   [u.namep], ebp ; 18/07/2015
                              <1>        ; mov r2,u.namep / u.namep points to char
                              <1>                     ; / following a / or nul
                              <1>  ;mov  bx, [u.dirbuf]
                              <1>        ; mov u.dirbuf,r1 / move i-node number in directory
                              <1>                   ; / entry to r1
00005061 20C0                 <1>  and   al, al
                              <1>        ; tst r4 / if r4 = 0 the end of file name reached,
                              <1>              ;  / if r4 = </> then go to next directory
                              <1>  ; mov ax, bx
00005063 66A1[B2740000]       <1>  mov   ax, [u.dirbuf] ; 17/06/2015
00005069 0F85FDFEFFFF         <1>         jnz     namei_2
                              <1>        ; bne 1b
                              <1>  ; AX = i-number of the file
                              <1> ;;nig:
0000506F C3                   <1>  retn
                              <1>        ; tst (r0)+ / gives non-error return
                              <1> ;;nib:
                              <1> ;;     xor   ax, ax ; Retro UNIX 8086 v1 modification !
                              <1>               ; ax = 0 -> file not found
                              <1> ;;     stc   ; 27/05/2013
                              <1> ;;     retn
                              <1>        ; rts r0
                              <1>
                              <1> trans_addr_nmbp:
                              <1>  ; 18/10/2015
                              <1>  ; 12/10/2015
00005070 8B2D[98740000]       <1>  mov   ebp, [u.namep]
                              <1> trans_addr_nm:
                              <1>  ; Convert virtual (pathname) address to physical address
                              <1>  ; (Retro UNIX 386 v1 feature only !)
                              <1>  ; 18/10/2015
                              <1>  ; 12/10/2015 (u.pnbase & u.pncount has been removed from code)
                              <1>  ; 02/07/2015
                              <1>  ; 17/06/2015
                              <1>  ; 16/06/2015
                              <1>  ;
                              <1>  ; INPUTS:
                              <1>  ;     ebp = pathname address (virtual) ; [u.namep]
                              <1>  ;     [u.pgdir] = user's page directory
                              <1>  ; OUTPUT:
                              <1>  ;       esi = physical address of the pathname
                              <1>  ;     ecx = remain byte count in the page
                              <1>  ;
                              <1>  ; (Modified registers: EAX, EBX, ECX, EDX, ESI)
                              <1>  ;
00005076 833D[D7740000]00     <1>         cmp     dword [u.ppgdir], 0  ; /etc/init ? (sysexec)
0000507D 7618                 <1>  jna   short trans_addr_nmk ; the caller is os kernel;
                              <1>                         ; it is already physical address
0000507F 50                   <1>        push  eax
00005080 89EB                 <1>  mov   ebx, ebp ; [u.namep] ; pathname address (virtual)
00005082 E843E6FFFF           <1>        call  get_physical_addr ; get physical address
00005087 7204                 <1>  jc    short tr_addr_nm_err
                              <1>  ; 18/10/2015
                              <1>  ; eax = physical address
                              <1>  ; cx = remain byte count in page (1-4096)
                              <1>        ; 12/10/2015 (cx = [u.pncount])
00005089 89C6                 <1>  mov   esi, eax ; 12/10/2015 (esi=[u.pnbase])
0000508B 58                   <1>  pop   eax
0000508C C3                   <1>  retn
                              <1>
                              <1> tr_addr_nm_err:
0000508D A3[CF740000]         <1>  mov   [u.error], eax
                              <1>  ;pop  eax
00005092 E9E5EFFFFF           <1>  jmp   error
                              <1>
                              <1> trans_addr_nmk:
                              <1>  ; 12/10/2015
                              <1>  ; 02/07/2015
00005097 8B35[98740000]       <1>  mov   esi, [u.namep]  ; [u.pnbase]
0000509D 66B90010             <1>  mov   cx, PAGE_SIZE ; 4096 ; [u.pncount]
000050A1 C3                   <1>  retn
                              <1>
                              <1> syschdir:
                              <1>  ; / makes the directory specified in the argument
                              <1>  ; / the current directory
                              <1>  ;
                              <1>  ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 19/06/2013 (Retro UNIX 8086 v1)
```

```
                             <1>  ;
                             <1>  ; 'syschdir' makes the directory specified in its argument
                             <1>  ; the current working directory.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     syschdir; name
                             <1>  ; Arguments:
                             <1>  ;     name - address of the path name of a directory
                             <1>  ;            terminated by nul byte.
                             <1>  ; Inputs: -
                             <1>  ; Outputs: -
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;      The user/application program puts address of
                             <1>  ;      the path name in BX register as 'syschdir'
                             <1>  ;      system call argument.
                             <1>
000050A2 891D[98740000]      <1>  mov   [u.namep], ebx
                             <1>        ;jsr r0,arg; u.namep / u.namep points to path name
000050A8 E87EFEFFFF          <1>  call  namei
                             <1>        ; jsr r0,namei / find its i-number
                             <1>  ;jc   error
                             <1>        ; br error3
000050AD 730F                <1>  jnc   short syschdir0
                             <1>  ; 'directory not found !' error
000050AF C705[CF740000]0C00- <1>  mov   dword [u.error], ERR_DIR_NOT_FOUND ; 12
000050B7 0000                <1>
000050B9 E9BEEFFFFF          <1>  jmp   error
                             <1> syschdir0:
000050BE E832070000          <1>  call  access
                             <1>        ; jsr r0,access; 2 / get i-node into core
000050C3 66F705[56710000]00- <1>  test  word [i.flgs], 4000h
000050CB 40                  <1>
                             <1>        ; bit $40000,i.flgs / is it a directory?
                             <1>  ;jz   error
                             <1>        ; beq error3 / no error
000050CC 750F                <1>  jnz   short syschdir1
000050CE C705[CF740000]1300- <1>  mov   dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
000050D6 0000                <1>
000050D8 E99FEFFFFF          <1>  jmp   error
                             <1> syschdir1:
000050DD 66A3[84740000]      <1>  mov   [u.cdir], ax
                             <1>        ; mov r1,u.cdir / move i-number to users
                             <1>                   ; / current directory
000050E3 66A1[66740000]      <1>  mov   ax, [cdev]
000050E9 66A3[C4740000]      <1>  mov   [u.cdrv], ax
                             <1>        ; mov cdev,u.cdev / move its device to users
                             <1>                     ; / current device
000050EF E9A8EFFFFF          <1>  jmp   sysret
                             <1>        ; br sysret3
                             <1>
                             <1> syschmod: ; < change mode of file >
                             <1>  ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 20/06/2013 - 07/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'syschmod' changes mode of the file whose name is given as
                             <1>  ; null terminated string pointed to by 'name' has it's mode
                             <1>  ; changed to 'mode'.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     syschmod; name; mode
                             <1>  ; Arguments:
                             <1>  ;     name - address of the file name
                             <1>  ;            terminated by null byte.
                             <1>  ;     mode - (new) mode/flags < attributes >
                             <1>  ;
                             <1>  ; Inputs: -
                             <1>  ; Outputs: -
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       'syschmod' system call has two arguments; so,
                             <1>  ;     * 1st argument, name is pointed to by BX register
                             <1>  ;     * 2nd argument, mode is in CX register
                             <1>  ;
                             <1>  ; Mode bits (Flags):
                             <1>  ;     bit 0 - write permission for non-owner (1)
                             <1>  ;     bit 1 - read permission for non-owner (2)
                             <1>  ;     bit 2 - write permission for owner (4)
                             <1>  ;     bit 3 - read permission for owner (8)
                             <1>  ;     bit 4 - executable flag (16)
                             <1>  ;     bit 5 - set user ID on execution flag (32)
                             <1>  ;     bit 6,7,8,9,10,11 are not used (undefined)
                             <1>  ;     bit 12 - large file flag (4096)
                             <1>  ;     bit 13 - file has modified flag (always on) (8192)
                             <1>  ;     bit 14 - directory flag (16384)
                             <1>  ;     bit 15 - 'i-node is allocated' flag (32768)
                             <1>
                             <1>  ; / name; mode
000050F4 E814000000          <1>  call  isown
                             <1>        ;jsr r0,isown / get the i-node and check user status
```

```
000050F9 66F705[56710000]00- <1>  test  word [i.flgs], 4000h
00005101 40                  <1>
                             <1>        ; bit $40000,i.flgs / directory?
00005102 7402                <1>  jz    short syschmod1
                             <1>        ; beq 2f / no
                             <1>  ; AL = (new) mode
00005104 24CF                <1>  and   al, 0CFh ; 11001111b (clears bit 4 & 5)
                             <1>        ; bic $60,r2 / su & ex / yes, clear set user id and
                             <1>                 ; / executable modes
                             <1> syschmod1: ; 2:
00005106 A2[56710000]        <1>  mov   [i.flgs], al
                             <1>        ; movb r2,i.flgs / move remaining mode to i.flgs
0000510B EB42                <1>  jmp   short isown1
                             <1>        ; br 1f
                             <1>
                             <1> isown:
                             <1>  ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 04/05/2013 - 07/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'isown' is given a file name (the 1st argument).
                             <1>  ;  It find the i-number of that file via 'namei'
                             <1>  ;  then gets the i-node into core via 'iget'.
                             <1>  ;  It then tests to see if the user is super user.
                             <1>  ;  If not, it cheks to see if the user is owner of
                             <1>  ;  the file. If he is not an error occurs.
                             <1>  ;  If user is the owner 'setimod' is called to indicate
                             <1>  ;  the inode has been modificed and the 2nd argument of
                             <1>  ;  the call is put in r2.
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    arguments of syschmod and syschown calls
                             <1>  ; OUTPUTS ->
                             <1>  ;    u.uid - id of user
                             <1>  ;    imod - set to a 1
                             <1>  ;    r2 - contains second argument of the system call
                             <1>  ;
                             <1>  ;   ((AX=R2) output as 2nd argument)
                             <1>  ;
                             <1>         ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
                             <1>  ;
                             <1>        ; jsr r0,arg2 / u.namep points to file name
                             <1>  ;; ! 2nd argument on top of stack !
                             <1>  ;; 22/06/2015 - 32 bit modifications
                             <1>  ;; 07/07/2013
0000510D 891D[98740000]      <1>  mov   [u.namep], ebx ;; 1st argument
00005113 51                  <1>  push  ecx ;; 2nd argument
                             <1>  ;;
00005114 E812FEFFFF          <1>  call  namei
                             <1>        ; jsr r0,namei / get its i-number
                             <1>        ; Retro UNIX 8086 v1 modification !
                             <1>        ; ax = 0 -> file not found
                             <1>  ;and  ax, ax
                             <1>  ;jz   error
                             <1>  ;jc   error ; 27/05/2013
                             <1>        ; br error3
00005119 730F                <1>  jnc   short isown0
                             <1>  ; 'file not found !' error
0000511B C705[CF740000]0C00- <1>  mov   dword [u.error], ERR_FILE_NOT_FOUND ; 12
00005123 0000                <1>
00005125 E952EFFFFF          <1>  jmp   error
                             <1> isown0:
0000512A E8EB050000          <1>  call  iget
                             <1>        ; jsr r0,iget / get i-node into core
0000512F A0[C6740000]        <1>  mov   al, [u.uid] ; 02/08/2013
00005134 08C0                <1>  or    al, al
                             <1>        ; tstb u.uid / super user?
00005136 7417                <1>  jz    short isown1
                             <1>        ; beq 1f / yes, branch
00005138 3A05[59710000]      <1>  cmp   al, [i.uid]
                             <1>        ; cmpb i.uid,u.uid / no, is this the owner of
                             <1>                     ; / the file
                             <1>  ;jne  error
                             <1>        ; beq 1f / yes
                             <1>        ; jmp error3 / no, error
0000513E 740F                <1>  je    short isown1
                             <1>
00005140 C705[CF740000]0B00- <1>  mov   dword [u.error], ERR_NOT_OWNER  ; 11
00005148 0000                <1>
                             <1>              ;  'permission denied !' error
0000514A E92DEFFFFF          <1>  jmp   error
                             <1> isown1: ; 1:
0000514F E8D9060000          <1>  call  setimod
                             <1>        ; jsr r0,setimod / indicates
                             <1>        ;             ; / i-node has been modified
00005154 58                  <1>  pop   eax ; 2nd argument
                             <1>        ; mov (sp)+,r2 / mode is put in r2
                             <1>               ; / (u.off put on stack with 2nd arg)
00005155 C3                  <1>  retn
                             <1>        ; rts r0
                             <1>
                             <1> ;;arg:  ; < get system call arguments >
                             <1>  ; 'arg' extracts an argument for a routine whose call is
```

```
                            <1>  ; of form:
                            <1>  ;     sys 'routine' ; arg1
                            <1>  ;           or
                            <1>  ;     sys 'routine' ; arg1 ; arg2
                            <1>  ;           or
                            <1>  ;     sys 'routine' ; arg1;...;arg10 (sys exec)
                            <1>  ;
                            <1>  ; INPUTS ->
                            <1>  ;    u.sp+18 - contains a pointer to one of arg1..argn
                            <1>  ;     This pointers's value is actually the value of
                            <1>  ;     update pc at the the trap to sysent (unkni) is
                            <1>  ;     made to process the sys instruction
                            <1>  ;    r0 - contains the return address for the routine
                            <1>  ;     that called arg. The data in the word pointer
                            <1>  ;     to by the return address is used as address
                            <1>  ;     in which the extracted argument is stored
                            <1>  ;
                            <1>  ; OUTPUTS ->
                            <1>  ;    'address' - contains the extracted argument
                            <1>  ;    u.sp+18 - is incremented by 2
                            <1>  ;    r1 - contains the extracted argument
                            <1>  ;    r0 - points to the next instruction to be
                            <1>  ;      executed in the calling routine.
                            <1>  ;
                            <1>
                            <1>  ; mov u.sp,r1
                            <1>  ; mov *18.(r1),*(r0)+ / put argument of system call
                            <1>              ; / into argument of arg2
                            <1>  ; add $2,18.(r1) / point pc on stack
                            <1>                    ; / to next system argument
                            <1>  ; rts r0
                            <1>
                            <1> ;;arg2: ; < get system calls arguments - with file name pointer>
                            <1>  ; 'arg2' takes first argument in system call
                            <1>  ;  (pointer to name of the file) and puts it in location
                            <1>  ;  u.namep; takes second argument and puts it in u.off
                            <1>  ;  and on top of the stack
                            <1>  ;
                            <1>  ; INPUTS ->
                            <1>  ;    u.sp, r0
                            <1>  ;
                            <1>  ; OUTPUTS ->
                            <1>  ;    u.namep
                            <1>  ;    u.off
                            <1>  ;    u.off pushed on stack
                            <1>  ;    r1
                            <1>  ;
                            <1>
                            <1>  ; jsr r0,arg; u.namep / u.namep contains value of
                            <1>                    ; / first arg in sys call
                            <1>  ; jsr r0,arg; u.off / u.off contains value of
                            <1>                    ; / second arg in sys call
                            <1>  ; mov r0,r1 / r0 points to calling routine
                            <1>  ; mov (sp),r0 / put operation code back in r0
                            <1>  ; mov u.off,(sp) / put pointer to second argument
                            <1>              ; / on stack
                            <1>  ; jmp (r1) / return to calling routine
                            <1>
                            <1> syschown: ; < change owner of file >
                            <1>  ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                            <1>  ; 20/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
                            <1>  ;
                            <1>  ; 'syschown' changes the owner of the file whose name is given
                            <1>  ; as null terminated string pointed to by 'name' has it's owner
                            <1>  ; changed to 'owner'
                            <1>  ;
                            <1>  ; Calling sequence:
                            <1>  ;     syschown; name; owner
                            <1>  ; Arguments:
                            <1>  ;     name - address of the file name
                            <1>  ;            terminated by null byte.
                            <1>  ;     owner - (new) owner (number/ID)
                            <1>  ;
                            <1>  ; Inputs: -
                            <1>  ; Outputs: -
                            <1>  ; ..............................................................
                            <1>  ;
                            <1>  ; Retro UNIX 8086 v1 modification:
                            <1>  ;       'syschown' system call has two arguments; so,
                            <1>  ;     * 1st argument, name is pointed to by BX register
                            <1>  ;     * 2nd argument, owner number is in CX register
                            <1>  ;
                            <1>  ; / name; owner
00005156 E8B2FFFFFF         <1>  call  isown
                            <1>        ; jsr r0,isown / get the i-node and check user status
0000515B 803D[C6740000]00   <1>  cmp   byte [u.uid], 0 ; 02/08/2013
                            <1>        ; tstb u.uid / super user
00005162 7418               <1>  jz    short syschown1
                            <1>        ; beq 2f / yes, 2f
00005164 F605[56710000]20   <1>         test    byte [i.flgs], 20h ; 32
                            <1>        ; bit $40,i.flgs / no, set userid on execution?
                            <1>  ;jnz  error
```

```
                             <1>        ; bne 3f / yes error, could create Trojan Horses
0000516B 740F                <1>  jz    short syschown1
                             <1>  ; 'permission denied !'
0000516D C705[CF740000]0B00- <1>  mov   dword [u.error], ERR_FILE_ACCESS  ; 11
00005175 0000                <1>
00005177 E900EFFFFF          <1>  jmp   error
                             <1> syschown1: ; 2:
                             <1>  ; AL = owner (number/ID)
0000517C A2[59710000]        <1>  mov   [i.uid], al ; 23/06/2015
                             <1>        ;  movb     r2,i.uid / no, put the new owners id
                             <1>                     ; / in the i-node
00005181 E916EFFFFF          <1>  jmp   sysret
                             <1>  ; 1:
                             <1>        ; jmp sysret4
                             <1>  ; 3:
                             <1>        ; jmp error
                             <1>
                             <1> systime: ; / get time of year
                             <1>  ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 20/06/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 20/06/2013
                             <1>  ; 'systime' gets the time of the year.
                             <1>  ; The present time is put on the stack.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     systime
                             <1>  ; Arguments: -
                             <1>  ;
                             <1>  ; Inputs: -
                             <1>  ; Outputs: sp+2, sp+4 - present time
                             <1>  ; ..............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       'systime' system call will return to the user
                             <1>  ;     with unix time (epoch) in DX:AX register pair
                             <1>  ;
                             <1>  ;     !! Major modification on original Unix v1 'systime'
                             <1>  ;     system call for PC compatibility !!
                             <1>
00005186 E82CEAFFFF          <1>  call  epoch
0000518B A3[80740000]        <1>  mov   [u.r0], eax
                             <1>        ; mov s.time,4(sp)
                             <1>        ; mov s.time+2,2(sp) / put the present time
                             <1>                       ; / on the stack
                             <1>        ; br sysret4
00005190 E907EFFFFF          <1>  jmp   sysret
                             <1>
                             <1> sysstime: ; / set time
                             <1>  ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 20/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysstime' sets the time. Only super user can use this call.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysstime
                             <1>  ; Arguments: -
                             <1>  ;
                             <1>  ; Inputs: sp+2, sp+4 - time system is to be set to.
                             <1>  ; Outputs: -
                             <1>  ; ..............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;     the user calls 'sysstime' with unix (epoch) time
                             <1>  ;     (to be set) is in CX:BX register pair as two arguments.
                             <1>  ;
                             <1>  ;     Retro UNIX 8086 v1 argument transfer method 2 is used
                             <1>  ;     to get sysstime system call arguments from the user;
                             <1>  ;     * 1st argument, lowword of unix time is in BX register
                             <1>  ;     * 2nd argument, highword of unix time is in CX register
                             <1>  ;
                             <1>  ;     !! Major modification on original Unix v1 'sysstime'
                             <1>  ;     system call for PC compatibility !!
                             <1>
00005195 803D[C6740000]00    <1>  cmp   byte [u.uid], 0
                             <1>        ; tstb u.uid / is user the super user
                             <1>  ;ja   error
                             <1>        ; bne error4 / no, error
0000519C 760F                <1>  jna   short systime1
                             <1>  ; 'permission denied !'
0000519E C705[CF740000]0B00- <1>  mov   dword [u.error], ERR_NOT_SUPERUSER  ; 11
000051A6 0000                <1>
000051A8 E9CFEEFFFF          <1>  jmp   error
                             <1> systime1:
                             <1>  ; 23/06/2015 (Retro UNIX 386 v1 - 32 bit version)
                             <1>  ; EBX = unix (epoch) time (from user)
000051AD 89D8                <1>  mov   eax, ebx
000051AF E885EBFFFF          <1>  call  set_date_time
                             <1>        ; mov 4(sp),s.time
                             <1>        ; mov 2(sp),s.time+2 / set the system time
000051B4 E9E3EEFFFF          <1>  jmp   sysret
                             <1>        ; br sysret4
                             <1>
                             <1> sysbreak:
                             <1>  ; 18/10/2015
                             <1>  ; 07/10/2015
                             <1>  ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 20/06/2013 - 24/03/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysbreak' sets the programs break points.
                             <1>  ; It checks the current break point (u.break) to see if it is
                             <1>  ; between "core" and the stack (sp). If it is, it is made an
                             <1>  ; even address (if it was odd) and the area between u.break
                             <1>  ; and the stack is cleared. The new breakpoint is then put
                             <1>  ; in u.break and control is passed to 'sysret'.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysbreak; addr
                             <1>  ; Arguments: -
                             <1>  ;
                             <1>  ; Inputs: u.break - current breakpoint
                             <1>  ; Outputs: u.break - new breakpoint
                             <1>  ;     area between old u.break and the stack (sp) is cleared.
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;     The user/application program puts breakpoint address
                             <1>  ;       in BX register as 'sysbreak' system call argument.
                             <1>  ;     (argument transfer method 1)
                             <1>  ;
                             <1>  ;  NOTE: Beginning of core is 0 in Retro UNIX 8086 v1 !
                             <1>  ;     ((!'sysbreak' is not needed in Retro UNIX 8086 v1!))
                             <1>  ;  NOTE:
                             <1>  ;     'sysbreak' clears extended part (beyond of previous
                             <1>  ;     'u.break' address) of user's memory for original unix's
                             <1>  ;     'bss' compatibility with Retro UNIX 8086 v1 (19/11/2013)
                             <1>
                             <1>        ; mov u.break,r1 / move users break point to r1
                             <1>        ; cmp r1,$core / is it the same or lower than core?
                             <1>        ; blos 1f / yes, 1f
                             <1>  ; 23/06/2015
000051B9 8B2D[AC740000]      <1>  mov   ebp, [u.break] ; virtual address (offset)
                             <1>  ;and  ebp, ebp
                             <1>  ;jz   short sysbreak_3
                             <1>  ; Retro UNIX 386 v1 NOTE: u.break points to virtual address !!!
                             <1>  ; (Even break point address is not needed for Retro UNIX 386 v1)
000051BF 8B15[78740000]      <1>  mov   edx, [u.sp] ; kernel stack at the beginning of sys call
000051C5 83C20C              <1>  add   edx, 12 ; EIP -4-> CS -4-> EFLAGS -4-> ESP (user)
                             <1>  ; 07/10/2015
000051C8 891D[AC740000]      <1>  mov   [u.break], ebx ; virtual address !!!
                             <1>  ;
000051CE 3B1A                <1>  cmp   ebx, [edx] ; compare new break point with
                             <1>                    ; with top of user's stack (virtual!)
000051D0 7327                <1>  jnb   short sysbreak_3
                             <1>        ; cmp r1,sp / is it the same or higher
                             <1>                ; / than the stack?
                             <1>        ; bhis 1f / yes, 1f
000051D2 89DE                <1>  mov   esi, ebx
000051D4 29EE                <1>  sub   esi, ebp ; new break point - old break point
000051D6 7621                <1>  jna   short sysbreak_3
                             <1>  ;push ebx
                             <1> sysbreak_1:
000051D8 89EB                <1>  mov   ebx, ebp
000051DA E8EBE4FFFF          <1>  call  get_physical_addr ; get physical address
000051DF 0F82A8FEFFFF        <1>  jc    tr_addr_nm_err
                             <1>  ; 18/10/2015
000051E5 89C7                <1>  mov   edi, eax
000051E7 29C0                <1>  sub   eax, eax ; 0
                             <1>         ; ECX = remain byte count in page (1-4096)
000051E9 39CE                <1>  cmp   esi, ecx
000051EB 7302                <1>  jnb   short sysbreak_2
000051ED 89F1                <1>  mov   ecx, esi
                             <1> sysbreak_2:
000051EF 29CE                <1>  sub   esi, ecx
000051F1 01CD                <1>  add   ebp, ecx
000051F3 F3AA                <1>  rep   stosb
000051F5 09F6                <1>  or    esi, esi
000051F7 75DF                <1>  jnz   short sysbreak_1
                             <1>  ;
                             <1>        ; bit $1,r1 / is it an odd address
                             <1>        ; beq 2f / no, its even
                             <1>        ; clrb (r1)+ / yes, make it even
                             <1>  ; 2: / clear area between the break point and the stack
                             <1>        ; cmp r1,sp / is it higher or same than the stack
                             <1>        ; bhis 1f / yes, quit
                             <1>        ; clr (r1)+ / clear word
                             <1>        ; br 2b / go back
                             <1>  ;pop  ebx
                             <1> sysbreak_3: ; 1:
                             <1>  ;mov  [u.break], ebx ; virtual address !!!
                             <1>        ; jsr r0,arg; u.break / put the "address"
                             <1>              ; / in u.break (set new break point)
                             <1>        ; br sysret4 / br sysret
000051F9 E99EEEFFFF          <1>  jmp   sysret
```

```
                             <1>
                             <1>
                             <1> maknod:
                             <1>  ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 02/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'maknod' creates an i-node and makes a directory entry
                             <1>  ; for this i-node in the current directory.
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - contains mode
                             <1>  ;    ii - current directory's i-number
                             <1>  ;
                             <1>  ; OUTPUTS ->
                             <1>  ;    u.dirbuf - contains i-number of free i-node
                             <1>  ;    i.flgs - flags in new i-node
                             <1>  ;    i.uid - filled with u.uid
                             <1>  ;    i.nlks - 1 is put in the number of links
                             <1>  ;    i.ctim - creation time
                             <1>  ;    i.ctim+2 - modification time
                             <1>  ;    imod - set via call to setimod
                             <1>  ;
                             <1>  ; ((AX = R1)) input
                             <1>  ;
                             <1>  ; (Retro UNIX Prototype :
                             <1>  ;     30/10/2012 - 01/03/2013, UNIXCOPY.ASM)
                             <1>         ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
                             <1>
                             <1>  ; / r1 contains the mode
000051FE 80CC80              <1>  or    ah, 80h  ; 10000000b
                             <1>        ; bis $100000,r1 / allocate flag set
00005201 6650                <1>  push  ax
                             <1>        ; mov r1,-(sp) / put mode on stack
                             <1>  ; 31/07/2013
00005203 66A1[62740000]      <1>  mov   ax, [ii] ; move current i-number to AX/r1
                             <1>        ; mov ii,r1 / move current i-number to r1
00005209 B201                <1>  mov   dl, 1 ; owner flag mask
0000520B E8E5050000          <1>  call  access
                             <1>        ; jsr r0,access; 1 / get its i-node into core
00005210 6650                <1>  push  ax
                             <1>        ; mov r1,-(sp) / put i-number on stack
00005212 66B82800            <1>  mov   ax, 40
                             <1>        ; mov $40.,r1 / r1 = 40
                             <1> maknod1: ; 1: / scan for a free i-node (next 4 instructions)
00005216 6640                <1>  inc   ax
                             <1>        ; inc r1 / r1 = r1 + 1
00005218 E8A9060000          <1>  call  imap
                             <1>        ; jsr r0,imap / get byte address and bit position in
                             <1>                  ; /     inode map in r2 & m
                             <1>          ; DX (MQ) has a 1 in the calculated bit position
                             <1>          ; eBX (R2) has byte address of the byte with allocation bit
                             <1>  ; 22/06/2015 - NOTE for next Retro UNIX version:
                             <1>  ;            Inode count must be checked here
                             <1>  ; (Original UNIX v1 did not check inode count here !?)
0000521D 8413                <1>  test  [ebx], dl
                             <1>        ; bitb mq,(r2) / is the i-node active
0000521F 75F5                <1>  jnz   short maknod1
                             <1>        ; bne 1b / yes, try the next one
00005221 0813                <1>  or    [ebx], dl
                             <1>        ; bisb mq,(r2) / no, make it active
                             <1>                   ; / (put a 1 in the bit map)
00005223 E8F2040000          <1>  call  iget
                             <1>        ; jsr r0,iget / get i-node into core
00005228 66F705[56710000]00- <1>  test  word [i.flgs], 8000h
00005230 80                  <1>
                             <1>        ; tst i.flgs / is i-node already allocated
00005231 75E3                <1>  jnz   short maknod1
                             <1>        ; blt 1b / yes, look for another one
00005233 66A3[B2740000]      <1>  mov   [u.dirbuf], ax
                             <1>        ; mov r1,u.dirbuf / no, put i-number in u.dirbuf
00005239 6658                <1>  pop   ax
                             <1>        ; mov (sp)+,r1 / get current i-number back
0000523B E8DA040000          <1>  call  iget
                             <1>        ; jsr r0,iget / get i-node in core
00005240 E880F7FFFF          <1>  call  mkdir
                             <1>        ; jsr r0,mkdir / make a directory entry
                             <1>                   ; / in current directory
00005245 66A1[B2740000]      <1>  mov   ax, [u.dirbuf]
                             <1>        ; mov u.dirbuf,r1 / r1 = new inode number
0000524B E8CA040000          <1>  call  iget
                             <1>        ; jsr r0,iget / get it into core
                             <1>        ; jsr r0,copyz; inode; inode+32. / 0 it out
00005250 B908000000          <1>  mov   ecx, 8
00005255 31C0                <1>  xor   eax, eax ; 0
00005257 BF[56710000]        <1>  mov   edi, inode
0000525C F3AB                <1>  rep   stosd
                             <1>  ;
0000525E 668F05[56710000]    <1>  pop   word [i.flgs]
                             <1>        ; mov (sp)+,i.flgs / fill flags
00005265 8A0D[C6740000]      <1>  mov   cl, [u.uid] ; 02/08/2013
0000526B 880D[59710000]      <1>  mov   [i.uid], cl
                             <1>        ; movb u.uid,i.uid / user id
```

```
00005271 C605[58710000]01    <1>  mov     byte [i.nlks], 1
                             <1>         ; movb $1,i.nlks / 1 link
                             <1>  ;call epoch ; Retro UNIX 8086 v1 modification !
                             <1>  ;mov  eax, [s.time]
                             <1>  ;mov  [i.ctim], eax
                             <1>         ; mov s.time,i.ctim / time created
                             <1>         ; mov s.time+2,i.ctim+2 / time modified
                             <1>  ; Retro UNIX 8086 v1 modification !
                             <1>  ; i.ctime=0, i.ctime+2=0 and
                             <1>           ; 'setimod' will set ctime of file via 'epoch'
00005278 E8B0050000          <1>  call setimod
                             <1>         ; jsr r0,setimod / set modified flag
0000527D C3                  <1>  retn
                             <1>         ; rts r0 / return
                             <1>
                             <1> sysseek: ; / moves read write pointer in an fsp entry
                             <1>  ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysseek' changes the r/w pointer of (3rd word of in an
                             <1>  ; fsp entry) of an open file whose file descriptor is in u.r0.
                             <1>  ; The file descriptor refers to a file open for reading or
                             <1>  ; writing. The read (or write) pointer is set as follows:
                             <1>  ;     * if 'ptrname' is 0, the pointer is set to offset.
                             <1>  ;     * if 'ptrname' is 1, the pointer is set to its
                             <1>  ;       current location plus offset.
                             <1>  ;     * if 'ptrname' is 2, the pointer is set to the
                             <1>  ;       size of file plus offset.
                             <1>  ; The error bit (e-bit) is set for an undefined descriptor.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysseek; offset; ptrname
                             <1>  ; Arguments:
                             <1>  ;     offset - number of bytes desired to move
                             <1>  ;            the r/w pointer
                             <1>  ;     ptrname - a switch indicated above
                             <1>  ;
                             <1>  ; Inputs: r0 - file descriptor
                             <1>  ; Outputs: -
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;        'sysseek' system call has three arguments; so,
                             <1>  ;     * 1st argument, file descriptor is in BX (BL) register
                             <1>  ;     * 2nd argument, offset is in CX register
                             <1>  ;     * 3rd argument, ptrname/switch is in DX (DL) register
                             <1>  ;
                             <1>
0000527E E823000000          <1>  call  seektell
                             <1>  ; AX = u.count
                             <1>  ; BX = *u.fofp
                             <1>         ; jsr r0,seektell / get proper value in u.count
                             <1>         ; add u.base,u.count / add u.base to it
00005283 0305[A0740000]      <1>  add   eax, [u.base] ; add offset (u.base) to base
00005289 8903                <1>  mov   [ebx], eax
                             <1>         ; mov u.count,*u.fofp / put result into r/w pointer
0000528B E90CEEFFFF          <1>  jmp   sysret
                             <1>         ; br sysret4
                             <1>
                             <1> systell: ; / get the r/w pointer
                             <1>  ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ; ! 'systell' does not work in original UNIX v1,
                             <1>  ;          it returns with error !
                             <1>  ; Inputs: r0 - file descriptor
                             <1>  ; Outputs: r0 - file r/w pointer
                             <1>
                             <1>  ;xor  ecx, ecx ; 0
00005290 BA01000000          <1>  mov   edx, 1 ; 05/08/2013
                             <1>  ;call       seektell
00005295 E812000000          <1>  call  seektell0 ; 05/08/2013
                             <1>  ;mov  ebx, [u.fofp]
0000529A 8B03                <1>  mov   eax, [ebx]
0000529C A3[80740000]        <1>  mov   [u.r0], eax
000052A1 E9F6EDFFFF          <1>  jmp   sysret
                             <1>
                             <1> ; Original unix v1 'systell' system call:
                             <1>         ; jsr r0,seektell
                             <1>         ; br error4
                             <1>
                             <1> seektell:
                             <1>  ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'seektell' puts the arguments from sysseek and systell
                             <1>  ; call in u.base and u.count. It then gets the i-number of
                             <1>  ; the file from the file descriptor in u.r0 and by calling
                             <1>  ; getf. The i-node is brought into core and then u.count
                             <1>  ; is checked to see it is a 0, 1, or 2.
                             <1>  ; If it is 0 - u.count stays the same
                             <1>  ;          1 - u.count = offset (u.fofp)
                             <1>  ;        2 - u.count = i.size (size of file)
                             <1>  ;
                             <1>  ; !! Retro UNIX 8086 v1 modification:
                             <1>  ;     Argument 1, file descriptor is in BX;
                             <1>  ;     Argument 2, offset is in CX;
                             <1>  ;     Argument 3, ptrname/switch is in DX register.
                             <1>  ;
                             <1>  ; mov       ax, 3 ; Argument transfer method 3 (three arguments)
                             <1>  ; call      arg
                             <1>  ;
                             <1>  ; ((Return -> ax = base for offset (position= base+offset))
                             <1>  ;
000052A6 890D[A0740000]      <1>  mov   [u.base], ecx ; offset
                             <1>        ; jsr r0,arg; u.base / puts offset in u.base
                             <1> seektell0:
000052AC 8915[A4740000]      <1>  mov   [u.count], edx
                             <1>        ; jsr r0,arg; u.count / put ptr name in u.count
                             <1>  ; mov ax, bx
                             <1>        ; mov *u.r0,r1 / file descriptor in r1
                             <1>                   ; / (index in u.fp list)
                             <1>  ; call      getf
                             <1>        ; jsr r0,getf / u.fofp points to 3rd word in fsp entry
                             <1>  ; BX = file descriptor (file number)
000052B2 E83DFCFFFF          <1>  call  getf1
000052B7 6609C0              <1>  or    ax, ax ; i-number of the file
                             <1>        ; mov r1,-(sp) / r1 has i-number of file,
                             <1>                     ; / put it on the stack
                             <1>  ;jz   error
                             <1>        ; beq error4 / if i-number is 0, not active so error
000052BA 750F                <1>  jnz   short seektell1
000052BC C705[CF740000]0A00- <1>  mov   dword [u.error], ERR_FILE_NOT_OPEN  ; 'file not open !'
000052C4 0000                <1>
000052C6 E9B1EDFFFF          <1>  jmp   error
                             <1> seektell1:
                             <1>  ;push eax
000052CB 80FC80              <1>  cmp   ah, 80h
000052CE 7203                <1>  jb    short seektell2
                             <1>        ; bgt .+4 / if its positive jump
000052D0 66F7D8              <1>  neg   ax
                             <1>        ; neg r1 / if not make it positive
                             <1> seektell2:
000052D3 E842040000          <1>  call  iget
                             <1>        ; jsr r0,iget / get its i-node into core
000052D8 8B1D[90740000]      <1>         mov     ebx, [u.fofp] ; 05/08/2013
000052DE 803D[A4740000]01    <1>  cmp   byte [u.count], 1
                             <1>        ; cmp u.count,$1 / is ptr name =1
000052E5 7705                <1>  ja    short seektell3
                             <1>        ; blt 2f / no its zero
000052E7 7409                <1>  je    short seektell_4
                             <1>        ; beq 1f / yes its 1
000052E9 31C0                <1>  xor   eax, eax
                             <1>  ;jmp  short seektell_5
000052EB C3                  <1>  retn
                             <1> seektell3:
000052EC A1[5A710000]        <1>         mov     eax, [i.size]
                             <1>                 ; mov i.size,u.count /  put number of bytes
                             <1>                                     ; / in file in u.count
                             <1>  ;jmp  short seektell_5
                             <1>        ; br 2f
000052F1 C3                  <1>  retn
                             <1> seektell_4: ; 1: / ptrname =1
                             <1>  ;mov  ebx, [u.fofp]
000052F2 8B03                <1>  mov   eax, [ebx]
                             <1>        ; mov *u.fofp,u.count / put offset in u.count
                             <1> ;seektell_5: ; 2: / ptrname =0
                             <1>  ;mov  [u.count], eax
                             <1>  ;pop  eax
                             <1>        ; mov (sp)+,r1 / i-number on stack  r1
000052F4 C3                  <1>  retn
                             <1>        ; rts r0
                             <1>
                             <1> sysintr: ; / set interrupt handling
                             <1>  ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 07/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysintr' sets the interrupt handling value. It puts
                             <1>  ; argument of its call in u.intr then branches into 'sysquit'
                             <1>  ; routine. u.tty is checked if to see if a control tty exists.
                             <1>  ; If one does the interrupt character in the tty buffer is
                             <1>  ; cleared and 'sysret'is called. If one does not exits
                             <1>  ; 'sysret' is just called.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysintr; arg
                             <1>  ; Argument:
                             <1>  ;     arg - if 0, interrupts (ASCII DELETE) are ignored.
                             <1>  ;         - if 1, intterupts cause their normal result
                             <1>  ;            i.e force an exit.
                             <1>  ;         - if arg is a location within the program,
                             <1>  ;           control is passed to that location when
                             <1>  ;           an interrupt occurs.
                             <1>  ; Inputs: -
                             <1>  ; Outputs: -
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       'sysintr' system call sets u.intr to value of BX
                             <1>  ;     then branches into sysquit.
                             <1>  ;
000052F5 66891D[BE740000]    <1>  mov   [u.intr], bx
                             <1>        ; jsr r0,arg; u.intr / put the argument in u.intr
                             <1>        ; br 1f / go into quit routine
000052FC E99BEDFFFF          <1>  jmp   sysret
                             <1>
                             <1> sysquit:
                             <1>  ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 07/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysquit' turns off the quit signal. it puts the argument of
                             <1>  ; the call in u.quit. u.tty is checked if to see if a control
                             <1>  ; tty exists. If one does the interrupt character in the tty
                             <1>  ; buffer is cleared and 'sysret'is called. If one does not exits
                             <1>  ; 'sysret' is just called.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysquit; arg
                             <1>  ; Argument:
                             <1>  ;     arg - if 0, this call diables quit signals from the
                             <1>  ;           typewriter (ASCII FS)
                             <1>  ;         - if 1, quits are re-enabled and cause execution to
                             <1>  ;           cease and a core image to be produced.
                             <1>  ;            i.e force an exit.
                             <1>  ;         - if arg is an addres in the program,
                             <1>  ;           a quit causes control to sent to that
                             <1>  ;           location.
                             <1>  ; Inputs: -
                             <1>  ; Outputs: -
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       'sysquit' system call sets u.quit to value of BX
                             <1>  ;     then branches into 'sysret'.
                             <1>  ;
00005301 66891D[C0740000]    <1>  mov   [u.quit], bx
00005308 E98FEDFFFF          <1>  jmp   sysret
                             <1>        ; jsr r0,arg; u.quit / put argument in u.quit
                             <1>  ;1:
                             <1>        ; mov u.ttyp,r1 / move pointer to control tty buffer
                             <1>                    ; / to r1
                             <1>        ; beq sysret4 / return to user
                             <1>        ; clrb 6(r1) / clear the interrupt character
                             <1>                 ; / in the tty buffer
                             <1>        ; br sysret4 / return to user
                             <1>
                             <1> syssetuid: ; / set process id
                             <1>  ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 07/07/2013 - 02/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'syssetuid' sets the user id (u.uid) of the current process
                             <1>  ; to the process id in (u.r0). Both the effective user and
                             <1>  ; u.uid and the real user u.ruid are set to this.
                             <1>  ; Only the super user can make this call.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     syssetuid
                             <1>  ; Arguments: -
                             <1>  ;
                             <1>  ; Inputs: (u.r0) - contains the process id.
                             <1>  ; Outputs: -
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       BL contains the (new) user ID of the current process
                             <1>
                             <1>        ; movb *u.r0,r1 / move process id (number) to r1
0000530D 3A1D[C7740000]      <1>  cmp   bl, [u.ruid]
                             <1>        ; cmpb r1,u.ruid / is it equal to the real user
                             <1>                     ; / id number
00005313 741E                <1>  je    short setuid1
                             <1>        ; beq 1f / yes
00005315 803D[C6740000]00    <1>  cmp   byte [u.uid], 0 ; 02/08/2013
                             <1>        ; tstb u.uid / no, is current user the super user?
                             <1>  ;ja   error
                             <1>        ; bne error4 / no, error
0000531C 760F                <1>  jna   short setuid0
0000531E C705[CF740000]0B00- <1>  mov   dword [u.error], ERR_NOT_SUPERUSER  ; 11
00005326 0000                <1>
                             <1>                    ;  'permission denied !' error
00005328 E94FEDFFFF          <1>  jmp   error
                             <1> setuid0:
0000532D 881D[C7740000]      <1>  mov   [u.ruid], bl
                             <1> setuid1: ; 1:
00005333 881D[C6740000]      <1>  mov   [u.uid], bl ; 02/08/2013
```

```
                             <1>        ; movb r1,u.uid / put process id in u.uid
                             <1>        ; movb r1,u.ruid / put process id in u.ruid
00005339 E95EEDFFFF          <1>  jmp   sysret
                             <1>        ; br sysret4 / system return
                             <1> 
                             <1> sysgetuid: ; < get user id >
                             <1>  ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 07/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysgetuid' returns the real user ID of the current process.
                             <1>  ; The real user ID identifies the person who is logged in,
                             <1>  ; in contradistinction to the effective user ID, which
                             <1>  ; determines his access permission at each moment. It is thus
                             <1>  ; useful to programs which operate using the 'set user ID'
                             <1>  ; mode, to find out who invoked them.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     syssetuid
                             <1>  ; Arguments: -
                             <1>  ;
                             <1>  ; Inputs: -
                             <1>  ; Outputs: (u.r0) - contains the real user's id.
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       AL contains the real user ID at return.
                             <1>  ;
0000533E 0FB605[C7740000]    <1>  movzx       eax, byte [u.ruid]
00005345 A3[80740000]        <1>  mov   [u.r0], eax
                             <1>        ; movb      u.ruid,*u.r0 / move the real user id to (u.r0)
0000534A E94DEDFFFF          <1>  jmp   sysret
                             <1>        ; br sysret4 / systerm return, sysret
                             <1> 
                             <1> anyi:
                             <1>  ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 25/04/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'anyi' is called if a file deleted while open.
                             <1>  ; "anyi" checks to see if someone else has opened this file.
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - contains an i-number
                             <1>  ;    fsp - start of table containing open files
                             <1>  ;
                             <1>  ; OUTPUTS ->
                             <1>  ;    "deleted" flag set in fsp entry of another occurrence of
                             <1>  ;        this file and r2 points 1st word of this fsp entry.
                             <1>  ;    if file not found - bit in i-node map is cleared
                             <1>  ;                  (i-node is freed)
                             <1>  ;               all blocks related to i-node are freed
                             <1>  ;             all flags in i-node are cleared
                             <1>  ; ((AX = R1)) input
                             <1>  ;
                             <1>  ;    (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)
                             <1>         ;    ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
                             <1>  ;
                             <1>        ; / r1 contains an i-number
0000534F BB[56720000]        <1>  mov   ebx, fsp
                             <1>        ; mov $fsp,r2 / move start of fsp table to r2
                             <1> anyi_1: ; 1:
00005354 663B03              <1>  cmp   ax, [ebx]
                             <1>        ; cmp r1,(r2) / do i-numbers match?
00005357 7433                <1>  je    short anyi_3
                             <1>        ; beq 1f / yes, 1f
00005359 66F7D8              <1>  neg   ax
                             <1>        ; neg r1 / no complement r1
0000535C 663B03              <1>  cmp   ax, [ebx]
                             <1>        ; cmp r1,(r2) / do they match now?
0000535F 742B                <1>  je    short anyi_3
                             <1>        ; beq 1f / yes, transfer
                             <1>        ; / i-numbers do not match
00005361 83C30A              <1>  add   ebx, 10 ; fsp table size is 10 bytes
                             <1>              ; in Retro UNIX 386 v1 (22/06/2015)
                             <1>        ; add $8,r2 / no, bump to next entry in fsp table
00005364 81FB[4A740000]      <1>  cmp   ebx, fsp + (nfiles*10) ; 22/06/2015
                             <1>        ; cmp r2,$fsp+[nfiles*8]
                             <1>                   ; / are we at last entry in the table
0000536A 72E8                <1>  jb    short anyi_1
                             <1>        ; blt 1b / no, check next entries i-number
                             <1>  ;cmp  ax, 32768
0000536C 80FC80              <1>  cmp   ah, 80h ; negative number check
                             <1>        ; tst r1 / yes, no match
                             <1>        ; bge .+4
0000536F 7203                <1>  jb    short anyi_2
00005371 66F7D8              <1>  neg   ax
                             <1>        ; neg r1 / make i-number positive
                             <1> anyi_2:
00005374 E84D050000          <1>  call  imap
                             <1>        ; jsr r0,imap / get address of allocation bit
                             <1>                  ; / in the i-map in r2
                             <1>  ;; DL/DX (MQ) has a 1 in the calculated bit position
                             <1>         ;; eBX (R2) has address of the byte with allocation bit
```

```
                              <1>  ; not dx
00005379 F6D2                 <1>  not   dl ;; 0 at calculated bit position, other bits are 1
                              <1>         ;and [ebx], dx
0000537B 2013                 <1>  and   [ebx], dl
                              <1>        ; bicb mq,(r2) / clear bit for i-node in the imap
0000537D E8CD040000           <1>  call  itrunc
                              <1>        ; jsr r0,itrunc / free all blocks related to i-node
00005382 66C705[56710000]00-  <1>  mov   word [i.flgs], 0
0000538A 00                   <1>
                              <1>        ; clr i.flgs / clear all flags in the i-node
0000538B C3                   <1>  retn
                              <1>        ;rts  r0 / return
                              <1> anyi_3: ; 1: / i-numbers match
0000538C FE4309               <1>  inc   byte [ebx+9] ; 22/06/2015
                              <1>        ;incb 7(r2) / increment upper byte of the 4th word
                              <1>           ; / in that fsp entry (deleted flag of fsp entry)
0000538F C3                   <1>  retn
                              <1>        ; rts r0
                                  %include 'u3.s'        ; 10/05/2015
                              <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS3.INC
                              <1> ; Last Modification: 15/09/2015
                              <1> ; ----------------------------------------------------------------------------
                              <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
                              <1> ; (v0.1 - Beginning: 11/07/2012)
                              <1> ;
                              <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                              <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                              <1> ; <Bell Laboratories (17/3/1972)>
                              <1> ; <Preliminary Release of UNIX Implementation Document>
                              <1> ;
                              <1> ; Retro UNIX 8086 v1 - U3.ASM (08/03/2014) //// UNIX v1 -> u3.s
                              <1> ;
                              <1> ; ****************************************************************************
                              <1>
                              <1> tswitch: ; Retro UNIX 386 v1
                              <1> tswap:
                              <1>  ; 01/09/2015
                              <1>  ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 14/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
                              <1>  ; time out swap, called when a user times out.
                              <1>  ; the user is put on the low priority queue.
                              <1>  ; This is done by making a link from the last user
                              <1>  ; on the low priority queue to him via a call to 'putlu'.
                              <1>  ; then he is swapped out.
                              <1>  ;
                              <1>  ; Retro UNIX 386 v1 modification ->
                              <1>  ;       swap (software task switch) is performed by changing
                              <1>  ;     user's page directory (u.pgdir) instead of segment change
                              <1>  ;     as in Retro UNIX 8086 v1.
                              <1>  ;
                              <1>  ; RETRO UNIX 8086 v1 modification ->
                              <1>  ;       'swap to disk' is replaced with 'change running segment'
                              <1>  ;     according to 8086 cpu (x86 real mode) architecture.
                              <1>  ;     pdp-11 was using 64KB uniform memory while IBM PC
                              <1>  ;     compatibles was using 1MB segmented memory
                              <1>  ;     in 8086/8088 times.
                              <1>  ;
                              <1>  ; INPUTS ->
                              <1>  ;    u.uno - users process number
                              <1>  ;    runq+4 - lowest priority queue
                              <1>  ; OUTPUTS ->
                              <1>  ;    r0 - users process number
                              <1>  ;    r2 - lowest priority queue address
                              <1>  ;
                              <1>  ; ((AX = R0, BX = R2)) output
                              <1>  ; ((Modified registers: EDX, EBX, ECX, ESI, EDI))
                              <1>  ;
00005390 A0[C9740000]         <1>  mov   al, [u.uno]
                              <1>              ; movb u.uno,r1 / move users process number to r1
                              <1>        ; mov  $runq+4,r2
                              <1>              ; / move lowest priority queue address to r2
00005395 E8CE000000           <1>         call putlu
                              <1>        ; jsr r0,putlu / create link from last user on Q to
                              <1>                     ; / u.uno's user
                              <1>
                              <1> switch: ; Retro UNIX 386 v1
                              <1> swap:
                              <1>  ; 02/09/2015
                              <1>  ; 01/09/2015
                              <1>  ; 31/08/2015
                              <1>  ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 14/04/2013 - 08/03/2014 (Retro UNIX 8086 v1)
                              <1>  ; 'swap' is routine that controls the swapping of processes
                              <1>  ; in and out of core.
                              <1>  ;
                              <1>  ; Retro UNIX 386 v1 modification ->
                              <1>  ;       swap (software task switch) is performed by changing
                              <1>  ;     user's page directory (u.pgdir) instead of segment change
                              <1>  ;     as in Retro UNIX 8086 v1.
                              <1>  ;
                              <1>  ; RETRO UNIX 8086 v1 modification ->
                              <1>  ;       'swap to disk' is replaced with 'change running segment'
```

```
                            <1>  ;     according to 8086 cpu (x86 real mode) architecture.
                            <1>  ;     pdp-11 was using 64KB uniform memory while IBM PC
                            <1>  ;     compatibles was using 1MB segmented memory
                            <1>  ;     in 8086/8088 times.
                            <1>  ;
                            <1>  ; INPUTS ->
                            <1>  ;    runq table - contains processes to run.
                            <1>  ;    p.link - contains next process in line to be run.
                            <1>  ;    u.uno - process number of process in core
                            <1>  ;    s.stack - swap stack used as an internal stack for swapping.
                            <1>  ; OUTPUTS ->
                            <1>  ;    (original unix v1 -> present process to its disk block)
                            <1>  ;    (original unix v1 -> new process into core ->
                            <1>  ;        Retro Unix 8086 v1 -> segment registers changed
                            <1>  ;        for new process)
                            <1>  ;    u.quant = 3 (Time quantum for a process)
                            <1>  ;     ((INT 1Ch count down speed -> 18.2 times per second)
                            <1>  ;    RETRO UNIX 8086 v1 will use INT 1Ch (18.2 times per second)
                            <1>  ;      for now, it will swap the process if there is not
                            <1>  ;      a keyboard event (keystroke) (Int 15h, function 4Fh)
                            <1>  ;      or will count down from 3 to 0 even if there is a
                            <1>  ;        keyboard event locking due to repetitive key strokes.
                            <1>  ;      u.quant will be reset to 3 for RETRO UNIX 8086 v1.
                            <1>  ;
                            <1>  ;    u.pri -points to highest priority run Q.
                            <1>  ;    r2 - points to the run queue.
                            <1>  ;    r1 - contains new process number
                            <1>  ;    r0 - points to place in routine or process that called
                            <1>  ;       swap all user parameters
                            <1>  ;
                            <1>  ; ((Modified registers: EAX, EDX, EBX, ECX, ESI, EDI))
                            <1>  ;
                            <1> swap_0:
                            <1>        ;mov $300,*$ps / processor priority = 6
0000539A BE[72740000]       <1>  mov   esi, runq
                            <1>        ; mov $runq,r2 / r2 points to runq table
                            <1> swap_1: ; 1: / search runq table for highest priority process
0000539F 668B06             <1>  mov   ax, [esi]
000053A2 6621C0             <1>  and   ax, ax
                            <1>              ; tst (r2)+ / are there any processes to run
                            <1>                ; / in this Q entry
000053A5 7507               <1>  jnz   short swap_2
                            <1>                ; bne 1f / yes, process 1f
                            <1>        ; cmp r2,$runq+6 / if zero compare address
                            <1>                     ; / to end of table
                            <1>        ; bne 1b / if not at end, go back
000053A7 E8E1000000         <1>  call  idle
                            <1>        ; jsr r0,idle; s.idlet+2 / wait for interrupt;
                            <1>                           ; / all queues are empty
000053AC EBF1               <1>  jmp   short swap_1
                            <1>        ; br swap
                            <1> swap_2: ; 1:
000053AE 0FB6D8             <1>  movzx ebx, al ; 02/09/2015
                            <1>        ; tst -(r2) / restore pointer to right Q entry
                            <1>        ; mov r2,u.pri / set present user to this run queue
                            <1>          ; movb (r2)+,r1 / move 1st process in queue to r1
000053B1 38E0               <1>  cmp   al, ah
                            <1>        ; cmpb r1,(r2)+ / is there only 1 process
                            <1>                    ; / in this Q to be run
000053B3 740A               <1>  je    short swap_3
                            <1>                ; beq 1f / yes
                            <1>        ; tst -(r2) / no, pt r2 back to this Q entry
                            <1>  ;movzx      ebx, al
000053B5 8AA3[F5710000]     <1>  mov   ah, [ebx+p.link-1]
000053BB 8826               <1>        mov   [esi], ah
                            <1>        ; movb p.link-1(r1),(r2) / move next process
                            <1>                           ; / in line into run queue
000053BD EB06               <1>  jmp   short swap_4
                            <1>                 ; br 2f
                            <1> swap_3: ; 1:
000053BF 6631D2             <1>  xor   dx, dx
000053C2 668916             <1>  mov   [esi], dx
                            <1>        ; clr -(r2) / zero the entry; no processes on the Q
                            <1> swap_4: ; / write out core to appropriate disk area and read
                            <1>       ; / in new process if required
                            <1>              ; clr *$ps / clear processor status
000053C5 8A25[C9740000]     <1>  mov   ah, [u.uno]
000053CB 38C4               <1>  cmp   ah, al
                            <1>        ; cmpb r1,u.uno / is this process the same as
                            <1>                    ; / the process in core?
000053CD 743B               <1>        je    short swap_8
                            <1>                ; beq 2f / yes, don't have to swap
                            <1>                ; mov r0,-(sp) / no, write out core; save r0
                            <1>                 ; / (address in routine that called swap)
                            <1>        ; mov r1,-(sp) / put r1 (new process #) on the stack
                            <1>  ; 01/09/2015
                            <1>  ;mov  [u.usp], esp
                            <1>                ; mov sp,u.usp / save stack pointer
                            <1>        ; mov $sstack,sp / move swap stack pointer
                            <1>                     ; / to the stack pointer
000053CF 08E4               <1>  or    ah, ah
                            <1>                ; tstb u.uno / is the process # = 0
```

```
000053D1 740D               <1>        jz    short swap_6 ; 'sysexit'
                            <1>        ; beq  1f / yes, kill process by overwriting
                            <1>  ; 02/09/2015
000053D3 8925[7C740000]     <1>  mov   [u.usp], esp ; return  address for 'syswait' & 'sleep'
                            <1>  ;
000053D9 E834000000         <1>  call  wswap
                            <1>        ;jsr r0,wswap / write out core to disk
                            <1>   ; 31/08/2015
                            <1>  ;movzx      ebx, al ; New (running) process number
000053DE EB1C               <1>  jmp   short swap_7
                            <1> swap_6:
                            <1>  ; 31/08/2015
                            <1>  ; Deallocate memory pages belong to the process
                            <1>  ; which is being terminated
                            <1>  ; 14/05/2015 ('sysexit')
                            <1>  ; Deallocate memory pages of the process
                            <1>  ; (Retro UNIX 386 v1 modification !)
                            <1>  ;
                            <1>  ; movzx ebx, al
000053E0 53                 <1>  push  ebx
000053E1 A1[D3740000]       <1>  mov   eax, [u.pgdir]  ; page directory of the process
000053E6 8B1D[D7740000]     <1>  mov   ebx, [u.ppgdir] ; page directory of the parent process
000053EC E87FDDFFFF         <1>  call  deallocate_page_dir
000053F1 A1[CA740000]       <1>  mov   eax, [u.upage] ; 'user' structure page of the process
000053F6 E814DEFFFF         <1>  call  deallocate_page
000053FB 5B                 <1>  pop   ebx
                            <1> swap_7: ;1:
                            <1>  ; 02/09/2015
                            <1>  ; 31/08/2015
                            <1>  ; 14/05/2015
000053FC C0E302             <1>  shl   bl, 2 ; * 4
000053FF 8B83[12720000]     <1>  mov   eax, [ebx+p.upage-4] ; the 'u' page of the new process
                            <1>  ;cli
00005405 E831000000         <1>  call  rswap
                            <1>        ; mov (sp)+,r1 / restore r1 to new process number
                            <1>        ; jsr r0,rswap / read new process into core
                            <1>              ; jsr r0,unpack / unpack the users stack from next
                            <1>                    ; / to his program to its normal
                            <1>  ; 01/09/2015
                            <1>  ;mov  esp, [u.usp]
                            <1>        ; mov u.usp,sp / location; restore stack pointer to
                            <1>                   ; / new process stack
                            <1>        ; mov (sp)+,r0 / put address of where the process
                            <1>                   ; / that just got swapped in, left off.,
                            <1>                   ; / i.e., transfer control to new process
                            <1>  ;sti
                            <1> swap_8: ;2:
                            <1>  ; RETRO UNIX 8086 v1 modification !
0000540A C605[BC740000]04   <1>  mov   byte [u.quant], time_count
                            <1>        ; movb    $30.,uquant / initialize process time quantum
00005411 C3                 <1>  retn
                            <1>        ; rts r0 / return
                            <1>
                            <1> wswap:  ; < swap out, swap to disk >
                            <1>  ; 09/05/2015 (Retro UNIX 386 v1 - Beginning)
                            <1>  ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
                            <1>  ; 'wswap' writes out the process that is in core onto its
                            <1>  ; appropriate disk area.
                            <1>  ;
                            <1>  ; Retro UNIX 386 v1 modification ->
                            <1>  ;       User (u) structure content and the user's register content
                            <1>  ;     will be copied to the process's/user's UPAGE (a page for
                            <1>  ;     saving 'u' structure and user registers for task switching).
                            <1>  ;     u.usp - points to kernel stack address which contains
                            <1>  ;           user's registers while entering system call.
                            <1>  ;     u.sp  - points to kernel stack address
                            <1>  ;           to return from system call -for IRET-.
                            <1>  ;     [u.usp]+32+16 = [u.sp]
                            <1>  ;     [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
                            <1>  ;           edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
                            <1>  ;
                            <1>  ; Retro UNIX 8086 v1 modification ->
                            <1>  ;       'swap to disk' is replaced with 'change running segment'
                            <1>  ;     according to 8086 cpu (x86 real mode) architecture.
                            <1>  ;     pdp-11 was using 64KB uniform memory while IBM PC
                            <1>  ;     compatibles was using 1MB segmented memory
                            <1>  ;     in 8086/8088 times.
                            <1>  ;
                            <1>  ; INPUTS ->
                            <1>  ;    u.break - points to end of program
                            <1>  ;    u.usp - stack pointer at the moment of swap
                            <1>  ;    core - beginning of process program
                            <1>  ;    ecore - end of core
                            <1>  ;    user - start of user parameter area
                            <1>  ;    u.uno - user process number
                            <1>  ;    p.dska - holds block number of process
                            <1>  ; OUTPUTS ->
                            <1>  ;    swp I/O queue
                            <1>  ;    p.break - negative word count of process
                            <1>  ;    r1 - process disk address
                            <1>  ;    r2 - negative word count
                            <1>  ;
                            <1>  ; RETRO UNIX 8086 v1 input/output:
                            <1>  ;
                            <1>  ; INPUTS ->
                            <1>  ;    u.uno - process number (to be swapped out)
                            <1>  ; OUTPUTS ->
                            <1>  ;    none
                            <1>  ;
                            <1>  ;   ((Modified registers: ECX, ESI, EDI))
                            <1>  ;
00005412 8B3D[CA740000]     <1>  mov   edi, [u.upage] ; process's user (u) structure page addr
00005418 B91C000000         <1>  mov   ecx, (U_SIZE + 3) / 4
0000541D BE[78740000]       <1>  mov   esi, user ; active user (u) structure
00005422 F3A5               <1>  rep   movsd
                            <1>  ;
00005424 8B35[7C740000]     <1>  mov   esi, [u.usp] ; esp (system stack pointer,
                            <1>                   ;      points to user registers)
0000542A 8B0D[78740000]     <1>  mov   ecx, [u.sp]  ; return address from the system call
                            <1>                   ; (for IRET)
                            <1>                   ; [u.sp] -> EIP (user)
                            <1>                   ; [u.sp+4]-> CS (user)
                            <1>                   ; [u.sp+8] -> EFLAGS (user)
                            <1>                   ; [u.sp+12] -> ESP (user)
                            <1>                   ; [u.sp+16] -> SS (user)
00005430 29F1               <1>  sub   ecx, esi     ; required space for user registers
00005432 83C114             <1>  add   ecx, 20          ; +5 dwords to return from system call
                            <1>                   ; (for IRET)
00005435 C1E902             <1>  shr   ecx, 2
00005438 F3A5               <1>  rep   movsd
0000543A C3                 <1>  retn
                            <1>
                            <1>  ; Original UNIX v1 'wswap' routine:
                            <1>  ; wswap:
                            <1>        ; mov *$30,u.emt / determines handling of emts
                            <1>              ; mov *$10,u.ilgins / determines handling of
                            <1>                    ; / illegal instructions
                            <1>        ; mov u.break,r2 / put process program break address in r2
                            <1>        ; inc r2 / add 1 to it
                            <1>        ; bic $1,r2 / make it even
                            <1>        ; mov r2,u.break / set break to an even location
                            <1>        ; mov u.usp,r3 / put users stack pointer
                            <1>                   ; / at moment of swap in r3
                            <1>        ; cmp r2,$core / is u.break less than $core
                            <1>        ; blos 2f / yes
                            <1>        ; cmp r2,r3 / no, is (u.break) greater than stack ptr.
                            <1>              ; bhis 2f / yes
                            <1>  ; 1:
                            <1>              ; mov (r3)+,(r2)+ / no, pack stack next to users program
                            <1>        ; cmp r3,$ecore / has stack reached end of core
                            <1>        ; bne 1b / no, keep packing
                            <1>        ; br 1f / yes
                            <1>  ; 2:
                            <1>              ; mov $ecore,r2 / put end of core in r2
                            <1>  ; 1:
                            <1>              ; sub  $user,r2 / get number of bytes to write out
                            <1>                 ; / (user up to end of stack gets written out)
                            <1>        ; neg r2 / make it negative
                            <1>        ; asr r2 / change bytes to words (divide by 2)
                            <1>        ; mov r2,swp+4 / word count
                            <1>        ; movb u.uno,r1 / move user process number to r1
                            <1>        ; asl r1 / x2 for index
                            <1>              ; mov r2,p.break-2(r1) / put negative of word count
                            <1>                         ; / into the p.break table
                            <1>              ; mov p.dska-2(r1),r1 / move disk address of swap area
                            <1>                        ; /     for process to r1
                            <1>              ; mov r1,swp+2 / put processes dska address in swp+2
                            <1>                   ; / (block number)
                            <1>        ; bis $1000,swp / set it up to write (set bit 9)
                            <1>              ; jsr r0,ppoke / write process out on swap area of disk
                            <1>  ; 1:
                            <1>              ; tstb swp+1 / is lt done writing?
                            <1>              ; bne 1b / no, wait
                            <1>        ; rts r0 / yes, return to swap
                            <1>
                            <1> rswap:  ; < swap in, swap from disk >
                            <1>  ; 15/09/2015
                            <1>  ; 28/08/2015
                            <1>  ; 14/05/2015
                            <1>  ; 09/05/2015 (Retro UNIX 386 v1 - Beginning)
                            <1>  ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
                            <1>  ; 'rswap' reads a process whose number is in r1,
                            <1>  ; from disk into core.
                            <1>  ;
                            <1>  ; Retro UNIX 386 v1 modification ->
                            <1>  ;       User (u) structure content and the user's register content
                            <1>  ;     will be restored from process's/user's UPAGE (a page for
                            <1>  ;     saving 'u' structure and user registers for task switching).
                            <1>  ;     u.usp - points to kernel stack address which contains
                            <1>  ;           user's registers while entering system call.
                            <1>  ;     u.sp  - points to kernel stack address
                            <1>  ;           to return from system call -for IRET-.
                            <1>  ;     [u.usp]+32+16 = [u.sp]
                            <1>  ;     [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
                            <1>  ;           edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
                            <1>  ;
                            <1>  ; RETRO UNIX 8086 v1 modification ->
                            <1>  ;       'swap to disk' is replaced with 'change running segment'
                            <1>  ;     according to 8086 cpu (x86 real mode) architecture.
                            <1>  ;     pdp-11 was using 64KB uniform memory while IBM PC
                            <1>  ;     compatibles was using 1MB segmented memory
                            <1>  ;     in 8086/8088 times.
                            <1>  ;
                            <1>  ; INPUTS ->
                            <1>  ;    r1 - process number of process to be read in
                            <1>  ;    p.break - negative of word count of process
                            <1>  ;    p.dska - disk address of the process
                            <1>  ;    u.emt - determines handling of emt's
                            <1>  ;    u.ilgins - determines handling of illegal instructions
                            <1>  ; OUTPUTS ->
                            <1>  ;    8 = (u.ilgins)
                            <1>  ;    24 = (u.emt)
                            <1>  ;    swp - bit 10 is set to indicate read
                            <1>  ;           (bit 15=0 when reading is done)
                            <1>  ;    swp+2 - disk block address
                            <1>  ;    swp+4 - negative word count
                            <1>  ;      ((swp+6 - address of user structure))
                            <1>  ;
                            <1>  ; RETRO UNIX 8086 v1 input/output:
                            <1>  ;
                            <1>  ; INPUTS ->
                            <1>  ;    AL     - new process number (to be swapped in)
                            <1>  ; OUTPUTS ->
                            <1>  ;    none
                            <1>  ;
                            <1>  ;   ((Modified registers: EAX, ECX, ESI, EDI, ESP))
                            <1>  ;
                            <1>  ; Retro UNIX 386 v1 - modification ! 14/05/2015
0000543B 89C6                 <1>  mov   esi, eax  ; process's user (u) structure page addr
0000543D B91C000000           <1>  mov   ecx, (U_SIZE + 3) / 4
00005442 BF[78740000]         <1>  mov   edi, user ; active user (u) structure
00005447 F3A5                 <1>  rep   movsd
00005449 58                   <1>  pop   eax ; 15/09/2015, 'rswap' return address
0000544A 8B3D[7C740000]       <1>  mov   edi, [u.usp] ; esp (system stack pointer,
                            <1>                  ;      points to user registers)
00005450 8B0D[78740000]       <1>  mov   ecx, [u.sp]  ; return address from the system call
                            <1>                  ; (for IRET)
                            <1>                  ; [u.sp] -> EIP (user)
                            <1>                  ; [u.sp+4]-> CS (user)
                            <1>                  ; [u.sp+8] -> EFLAGS (user)
                            <1>                  ; [u.sp+12] -> ESP (user)
                            <1>                  ; [u.sp+16] -> SS (user)
                            <1>  ; 28/08/2015
00005456 29F9                 <1>  sub   ecx, edi     ; required space for user registers
00005458 83C114               <1>  add   ecx, 20          ; +5 dwords to return from system call
                            <1>                  ; (for IRET)
0000545B C1E902               <1>  shr   ecx, 2
0000545E F3A5                 <1>  rep   movsd
00005460 8B25[7C740000]       <1>  mov   esp, [u.usp] ; 15/09/2015
00005466 50                   <1>  push  eax ; 15/09/2015 'rswap' return address
00005467 C3                   <1>  retn
                            <1>
                            <1>  ; Original UNIX v1 'rswap'  and 'unpack' routines:
                            <1>  ;rswap:
                            <1>             ; asl r1 / process number x2 for index
                            <1>             ; mov p.break-2(r1), swp+4 / word count
                            <1>             ; mov p.dska-2(r1),swp+2 / disk address
                            <1>             ; bis $2000,swp / read
                            <1>             ; jsr r0,ppoke / read it in
                            <1>  ; 1:
                            <1>             ; tstb swp+1 / done
                            <1>             ; bne 1b / no, wait for bit 15 to clear (inhibit bit)
                            <1>             ; mov u.emt,*$30 / yes move these
                            <1>             ; mov u.ilgins,*$10 / back
                            <1>             ; rts r0 / return
                            <1>
                            <1>  ;unpack: ; / move stack back to its normal place
                            <1>        ; mov u.break,r2 / r2 points to end of user program
                            <1>                ; cmp r2,$core / at beginning of user program yet?
                            <1>        ; blos 2f / yes, return
                            <1>        ; cmp r2,u.usp / is break_above the stack pointer
                            <1>                   ; / before swapping
                            <1>        ; bhis 2f / yes, return
                            <1>        ; mov $ecore,r3 / r3 points to end of core
                            <1>        ; add r3,r2
                            <1>        ; sub u.usp,r2 / end of users stack is in r2
                            <1>  ; 1:
                            <1>        ; mov -(r2),-(r3) / move stack back to its normal place
                            <1>        ; cmp r2,u.break / in core
                            <1>        ; bne 1b
                            <1>  ; 2:
                            <1>             ; rts r0
                            <1>
                            <1> putlu:
                            <1>  ; 12/09/2015
                            <1>  ; 02/09/2015
                            <1>  ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
                            <1>  ; 15/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
                            <1>  ; 'putlu' is called with a process number in r1 and a pointer
                            <1>  ; to lowest priority Q (runq+4) in r2. A link is created from
                            <1>  ; the last process on the queue to process in r1 by putting
                            <1>  ; the process number in r1 into the last process's link.
                            <1>  ;
                            <1>  ; INPUTS ->
                            <1>  ;    r1 - user process number
                            <1>  ;    r2 - points to lowest priority queue
                            <1>  ;    p.dska - disk address of the process
                            <1>  ;    u.emt - determines handling of emt's
                            <1>  ;    u.ilgins - determines handling of illegal instructions
                            <1>  ; OUTPUTS ->
                            <1>  ;    r3 - process number of last process on the queue upon
                            <1>  ;       entering putlu
                            <1>  ;    p.link-1 + r3 - process number in r1
                            <1>  ;    r2 - points to lowest priority queue
                            <1>  ;
                            <1>  ; ((Modified registers: EDX, EBX))
                            <1>  ;
                            <1>  ; / r1 = user process no.; r2 points to lowest priority queue
                            <1>
                            <1>  ; eBX = r2
                            <1>  ; eAX = r1 (AL=r1b)
                            <1>
00005468 BB[72740000]         <1>  mov   ebx, runq
0000546D 0FB613               <1>  movzx       edx, byte [ebx]
00005470 43                   <1>  inc   ebx
00005471 20D2                 <1>  and   dl, dl
                            <1>        ; tstb (r2)+ / is queue empty?
00005473 740A                 <1>        jz    short putlu_1
                            <1>        ; beq 1f / yes, branch
00005475 8A13                 <1>  mov   dl, [ebx] ; 12/09/2015
                            <1>        ; movb (r2),r3 / no, save the "last user" process number
                            <1>                   ; / in r3
00005477 8882[F5710000]       <1>        mov   [edx+p.link-1], al
                            <1>        ; movb r1,p.link-1(r3) / put pointer to user on
                            <1>                   ; / "last users" link
0000547D EB03                 <1>  jmp   short putlu_2
                            <1>        ; br 2f /
                            <1> putlu_1: ; 1:
0000547F 8843FF               <1>  mov   [ebx-1], al
                            <1>              ; movb r1,-1(r2) / user is only user;
                            <1>                  ; / put process no. at beginning and at end
                            <1> putlu_2: ; 2:
00005482 8803                 <1>  mov   [ebx], al
                            <1>              ; movb r1,(r2) / user process in r1 is now the last entry
                            <1>                   ; / on the queue
00005484 88C2                 <1>  mov   dl, al
00005486 88B2[F5710000]       <1>         mov     [edx+p.link-1], dh ; 0
                            <1>        ; dec r2 / restore r2
0000548C C3                   <1>         retn
                            <1>        ; rts r0
                            <1>
                            <1> ;copyz:
                            <1> ;       mov     r1,-(sp) / put r1 on stack
                            <1> ;       mov     r2,-(sp) / put r2 on stack
                            <1> ;       mov     (r0)+,r1
                            <1> ;       mov     (r0)+,r2
                            <1> ;1:
                            <1> ;       clr     (r1)+ / clear all locations between r1 and r2
                            <1> ;       cmp     r1,r2
                            <1> ;       blo     1b
                            <1> ;       mov     (sp)+,r2 / restore r2
                            <1> ;       mov     (sp)+,r1 / restore r1
                            <1> ;       rts     r0
                            <1>
                            <1> idle:
                            <1>  ; 01/09/2015
                            <1>  ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
                            <1>  ; 10/04/2013 - 23/10/2013 (Retro UNIX 8086 v1)
                            <1>  ; (idle & wait loop)
                            <1>  ; Retro Unix 8086 v1 modification on original UNIX v1
                            <1>  ; idle procedure!
                            <1>        ;
                            <1>        ; 01/09/2015
0000548D FB                   <1>  sti
                            <1>        ; 29/07/2013
0000548E F4                   <1>        hlt
0000548F 90                   <1>        nop ; 10/10/2013
00005490 90                   <1>        nop
00005491 90                   <1>        nop
                            <1>        ; 23/10/2013
00005492 90                   <1>        nop
00005493 90                   <1>        nop
00005494 90                   <1>        nop
00005495 90                   <1>        nop
00005496 C3                   <1>        retn
                            <1>
                            <1>  ;mov *$ps,-(sp) / save ps on stack
                            <1>  ;clr *$ps / clear ps
```

```
                             <1>  ;mov clockp,-(sp) / save clockp on stack
                             <1>  ;mov (r0)+,clockp / arg to idle in clockp
                             <1>  ;1 / wait for interrupt
                             <1>  ;mov (sp)+,clockp / restore clockp, ps
                             <1>  ;mov (sp)+,*$ps
                             <1>  ;rts r0
                             <1>
                             <1> clear:
                             <1>  ; 10/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 09/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)
                             <1>  ; 'clear' zero's out of a block (whose block number is in r1)
                             <1>  ; on the current device (cdev)
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - block number of block to be zeroed
                             <1>  ;    cdev - current device number
                             <1>  ; OUTPUTS ->
                             <1>  ;    a zeroed I/O buffer onto the current device
                             <1>  ;    r1 - points to last entry in the I/O buffer
                             <1>  ;
                             <1>  ; ((AX = R1)) input/output
                             <1>  ;    (Retro UNIX Prototype : 18/11/2012 - 14/11/2012, UNIXCOPY.ASM)
                             <1>        ;    ((Modified registers: EDX, ECX, EBX, ESI, EDI, EBP))
                             <1>
00005497 E8AB0D0000          <1>  call  wslot
                             <1>        ; jsr r0,wslot / get an I/O buffer set bits 9 and 15 in first
                             <1>                  ; / word of I/O queue r5 points to first data word in buffer
0000549C 89DF                <1>  mov   edi, ebx ; r5
0000549E 89C2                <1>  mov   edx, eax
000054A0 B980000000          <1>  mov   ecx, 128
                             <1>        ; mov $256.,r3
000054A5 31C0                <1>  xor   eax, eax
000054A7 F3AB                <1>  rep   stosd
000054A9 89D0                <1>  mov   eax, edx
                             <1> ; 1:
                             <1>             ; clr (r5)+ / zero data word in buffer
                             <1>             ; dec r3
                             <1>             ; bgt 1b / branch until all data words in buffer are zero
000054AB E8B30D0000          <1>  call  dskwr
                             <1>        ; jsr r0,dskwr / write zeroed buffer area out onto physical
                             <1>                            ; / block specified in r1
                             <1>  ; eAX (r1) = block number
000054B0 C3                  <1>  retn
                             <1>        ; rts r0
                                 %include 'u4.s'        ; 15/04/2015
                             <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS4.INC
                             <1> ; Last Modification: 14/10/2015
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
                             <1> ; (v0.1 - Beginning: 11/07/2012)
                             <1> ;
                             <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                             <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                             <1> ; <Bell Laboratories (17/3/1972)>
                             <1> ; <Preliminary Release of UNIX Implementation Document>
                             <1> ;
                             <1> ; Retro UNIX 8086 v1 - U4.ASM (04/07/2014) //// UNIX v1 -> u4.s
                             <1> ;
                             <1> ; ****************************************************************************
                             <1>
                             <1> ;setisp:
                             <1>        ;mov     r1,-(sp)
                             <1>        ;mov     r2,-(sp)
                             <1>        ;mov     r3,-(sp)
                             <1>        ;mov     clockp,-(sp)
                             <1>        ;mov     $s.syst+2,clockp
                             <1>        ;jmp     (r0)
                             <1>
                             <1> clock: ; / interrupt from 60 cycle clock
                             <1>
                             <1>  ; 14/10/2015
                             <1>  ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 07/12/2013 - 10/04/2014 (Retro UNIX 8086 v1)
                             <1>
                             <1>        ;mov     r0,-(sp) / save r0
                             <1>        ;tst     *$lks / restart clock?
                             <1>        ;mov     $s.time+2,r0 / increment the time of day
                             <1>        ;inc     (r0)
                             <1>        ;bne     1f
                             <1>        ;inc     -(r0)
                             <1> ;1:
                             <1>        ;mov     clockp,r0 / increment appropriate time category
                             <1>        ;inc     (r0)
                             <1>        ;bne     1f
                             <1>        ;inc     -(r0)
                             <1> ;1:
                             <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             <1>
000054B1 803D[BC740000]00    <1>  cmp   byte [u.quant], 0
000054B8 772C                <1>  ja    short clk_1
                             <1>  ;
000054BA 803D[77740000]FF    <1>         cmp     byte [sysflg], 0FFh ; user or system space ?
```

```
000054C1 7529                  <1>  jne   short clk_2 ; system space (sysflg <> 0FFh)
000054C3 803D[C9740000]01      <1>  cmp     byte [u.uno], 1 ; /etc/init ?
000054CA 761A                  <1>  jna   short clk_1 ; yes, do not swap out
000054CC 66833D[BE740000]00    <1>  cmp   word [u.intr], 0
000054D4 7616                  <1>  jna   short clk_2
                               <1> clk_0:
                               <1>  ; 14/10/2015
000054D6 FE05[77740000]        <1>  inc   byte [sysflg]     ; Now, we are in system space
000054DC 58                    <1>  pop   eax ; return address to the timer interrupt
                               <1>  ;
000054DD B020                  <1>  MOV   AL,EOI                  ; GET END OF INTERRUPT MASK
                               <1>  ;CLI                   ; DISABLE INTERRUPTS TILL STACK CLEARED
000054DF E620                  <1>  OUT   INTA00,AL         ; END OF INTERRUPT TO 8259 - 1
                               <1>  ;
000054E1 E912ECFFFF            <1>  jmp     sysrelease ; 'sys release' by clock/timer
                               <1> clk_1:
000054E6 FE0D[BC740000]        <1>  dec   byte [u.quant]
                               <1> clk_2:
000054EC C3                    <1>  retn   ; return to (hardware) timer interrupt routine
                               <1>
                               <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               <1>
                               <1>       ;mov     $uquant,r0 / decrement user time quantum
                               <1>       ;decb    (r0)
                               <1>       ;bge     1f / if less than 0
                               <1>       ;clrb    (r0) / make it 0
                               <1> ;1: / decrement time out counts return now if priority was not 0
                               <1>       ;cmp     4(sp),$200 / ps greater than or equal to 200
                               <1>       ;bge     2f / yes, check time outs
                               <1>       ;tstb    (r0) / no, user timed out?
                               <1>       ;bne     1f / no
                               <1>       ;cmpb    sysflg,$-1 / yes, are we outside the system?
                               <1>       ;bne     1f / no, 1f
                               <1>       ;mov     (sp)+,r0 / yes, put users r0 in r0
                               <1>       ;sys     0 / sysrele
                               <1>       ;rti
                               <1> ;2: / priority is high so just decrement time out counts
                               <1>       ;mov     $toutt,r0 / r0 points to beginning of time out table
                               <1> ;2:
                               <1>       ;tstb    (r0) / is the time out?
                               <1>       ;beq     3f / yes, 3f (get next entry)
                               <1>       ;decb    (r0) / no, decrement the time
                               <1>       ;bne     3f / isit zero now?
                               <1>       ;incb    (r0) / yes, increment the time
                               <1> ;3:
                               <1>       ;inc     r0 / next entry
                               <1>       ;cmp     r0,$touts / end of toutt table?
                               <1>       ;blo     2b / no, check this entry
                               <1>       ;mov     (sp)+,r0 / yes, restore r0
                               <1>       ;rti / return from interrupt
                               <1> ;1: / decrement time out counts; if 0 call subroutine
                               <1>       ;mov     (sp)+,r0 / restore r0
                               <1>       ;mov     $240,*$ps / set processor priority to 5
                               <1>       ;jsr     r0,setisp / save registers
                               <1>       ;mov     $touts-toutt-1,r0 / set up r0 as index to decrement thru
                               <1>                               ;  / the table
                               <1> ;1:
                               <1>       ;tstb    toutt(r0) / is the time out for this entry
                               <1>       ;beq     2f / yes
                               <1>       ;decb    toutt(r0) / no, decrement the time
                               <1>       ;bne     2f / is the time 0, now
                               <1>       ;asl     r0 / yes, 2 x r0 to get word index for tout entry
                               <1>       ;jsr     r0,*touts(r0) / go to appropriate routine specified in this
                               <1>       ;asr     r0 / touts entry; set r0 back to toutt index
                               <1> ;2:
                               <1>       ;dec     r0 / set up r0 for next entry
                               <1>       ;bge     1b / finished? , no, go back
                               <1>       ;br      retisp / yes, restore registers and do a rti
                               <1>
                               <1> ;retisp:
                               <1>       ;mov     (sp)+,clockp / pop values before interrupt off the stack
                               <1>       ;mov     (sp)+,r3
                               <1>       ;mov     (sp)+,r2
                               <1>       ;mov     (sp)+,r1
                               <1>       ;mov     (sp)+,r0
                               <1>       ;rti     / return from interrupt
                               <1>
                               <1>
                               <1> wakeup: ; / wakeup processes waiting for an event
                               <1>  ; / by linking them to the queue
                               <1>  ;
                               <1>  ; 15/09/2015
                               <1>  ; 29/06/2015
                               <1>  ; 15/04/2015 (Retro UNIX 386 v1 - Beginning)
                               <1>  ;
                               <1>  ; 15/05/2013 - 02/06/2014
                               <1>  ; Retro UNIX 8086 v1 modification !
                               <1>  ; (Process/task switching routine by using
                               <1>  ; Retro UNIX 8086 v1 keyboard interrupt output.)
                               <1>  ;
                               <1>  ; In original UNIX v1, 'wakeup' is called to wake the process
                               <1>  ; sleeping in the specified wait channel by creating a link
```

```
                              <1>  ; to it from the last user process on the run queue.
                              <1>  ; If there is no process to wake up, nothing happens.
                              <1>  ;
                              <1>  ; In Retro UNIX 8086 v1, Int 09h keyboard interrupt will set
                              <1>  ; 'switching' status of the current process (owns current tty)
                              <1>  ; (via alt + function keys) to a process which has highest
                              <1>  ; priority (on run queue) on the requested tty (0 to 7, except
                              <1>  ; 8 and 9 which are tty identifiers of COM1, COM2 serial ports)
                              <1>  ; as it's console tty. (NOTE: 'p.ttyc' is used to set console
                              <1>  ; tty for tty switching by keyboard.)
                              <1>  ;
                              <1>  ; INPUT ->
                              <1>  ;        AL = wait channel (r3) ('tty number' for now)
                              <1>  ;        ;;EBX = Run queue (r2) offset
                              <1>  ;
                              <1>  ; ((modified registers: EAX, EBX))
                              <1>  ;
000054ED 0FB6D8               <1>  movzx ebx, al ; 29/06/2015
000054F0 81C3[08710000]       <1>  add   ebx, wlist
000054F6 8A03                 <1>  mov   al, [ebx] ; waiting list (waiting process number)
000054F8 20C0                 <1>  and   al, al
000054FA 7424                 <1>  jz    short wa0 ; nothing to wakeup
                              <1>  ;
000054FC 30E4                 <1>  xor   ah, ah
000054FE 8825[BC740000]       <1>  mov   [u.quant], ah ; 0 ; time quantum = 0
00005504 8823                 <1>  mov   [ebx], ah ; 0 ; zero wait channel entry
                              <1>  ; 15/09/2015
00005506 0FB6D8               <1>  movzx ebx, al
00005509 88A3[E5710000]       <1>  mov   [ebx+p.waitc-1], ah ; 0
0000550F FEC4                 <1>  inc   ah
00005511 88A3[05720000]       <1>  mov   byte [ebx+p.stat-1], ah ; 1 ; SRUN
                              <1>  ;
00005517 57                   <1>  push  edi
00005518 52                   <1>  push  edx
00005519 E84AFFFFFF           <1>  call  putlu
0000551E 5A                   <1>  pop   edx
0000551F 5F                   <1>  pop   edi
                              <1> wa0:
00005520 C3                   <1>  retn
                              <1>
                              <1> sleep:
                              <1>  ; 15/09/2015
                              <1>  ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ;
                              <1>  ; 09/05/2013 - 20/03/2014
                              <1>  ;
                              <1>  ; Retro UNIX 8086 v1 modification !
                              <1>  ; (Process/task switching and quit routine by using
                              <1>  ; Retro UNIX 8086 v1 keyboard interrupt output.))
                              <1>  ;
                              <1>  ; In original UNIX v1, 'sleep' is called to wait for
                              <1>  ; tty and tape output or input becomes available
                              <1>  ; and process is put on waiting channel and swapped out,
                              <1>  ; then -when the tty or tape is ready to write or read-
                              <1>  ; 'wakeup' gets process back to active swapped-in status.)
                              <1>  ;
                              <1>  ; In Retro UNIX 8086 v1, Int 1Bh ctrl+brk interrupt and
                              <1>  ; Int 09h keyboard interrupt will set 'quit' or 'switching'
                              <1>  ; status of the current process also INT 1Ch will count down
                              <1>  ; 'uquant' value and INT 09h will redirect scancode of keystroke
                              <1>  ; to tty buffer of the current process and kernel will get
                              <1>  ; user input by using tty buffer of the current process
                              <1>  ; (instead of standard INT 16h interrupt).
                              <1>  ; TTY output will be redirected to related video page of text mode
                              <1>  ; (INT 10h will be called with different video page depending
                              <1>  ; on tty assignment of the active process: 0 to 7 for
                              <1>  ; pseudo screens.)
                              <1>  ;
                              <1>  ; In Retro UNIX 8086 v1, 'sleep' will be called to wait for
                              <1>  ; a keystroke from keyboard or wait for reading or writing
                              <1>  ; characters/data on serial port(s).
                              <1>  ;
                              <1>  ; Character/Terminal input/output through COM1 and COM2 will be
                              <1>  ; performed by related routines in addition to pseudo TTY routines.
                              <1>  ;
                              <1>  ; R1 = AH = wait channel (0-9 for TTYs) ; 05/10/2013 (22/09/2013)
                              <1>  ;
                              <1>  ;; 05/10/2013
                              <1>         ;10/12/2013
                              <1>  ;cmp   byte [u.uno], 1
                              <1>         ;ja    short sleep0
                              <1>  ;retn
                              <1>
                              <1>  ; 20/03/2014
                              <1>  ;mov  bx, [runq]
                              <1>  ;cmp  bl, bh
                              <1>  ;jne  short sleep0
                              <1>  ; 25/02/2014
                              <1>  ;cmp word ptr [runq], 0
                              <1>  ;ja short sleep0
                              <1>  ;retn
                              <1> sleep0:
```

```
                             <1>  ;
00005521 E854000000          <1>  call  isintr
00005526 0F8570EBFFFF        <1>  jnz   sysret
                             <1>        ; / wait for event
                             <1>              ; jsr r0,isintr / check to see if interrupt
                             <1>                    ; / or quit from user
                             <1>                          ; br 2f / something happened
                             <1>                    ; / yes, his interrupt so return
                             <1>                                ;     / to user
                             <1>
                             <1>  ; 30/06/2015
0000552C 0FB6DC              <1>        movzx ebx, ah ; 30/06/2015
0000552F 81C3[08710000]      <1>  add   ebx, wlist
00005535 8A03                <1>  mov   al, [ebx]
00005537 20C0                <1>  and   al, al
00005539 7407                <1>  jz    short sleep1
0000553B 53                  <1>  push  ebx
0000553C E827FFFFFF          <1>  call  putlu
00005541 5B                  <1>  pop   ebx
                             <1> sleep1:
00005542 A0[C9740000]        <1>  mov   al, [u.uno]
00005547 8803                <1>        mov   [ebx], al   ; put the process number
                             <1>                    ; in the wait channel
                             <1>        ; mov (r0)+,r1 / put number of wait channel in r1
                             <1>        ; movb wlist(r1),-(sp) / put old process number in there,
                             <1>                         ; / on the stack
                             <1>              ; movb u.uno,wlist(r1) / put process number of process
                             <1>                         ; / to put to sleep in there
                             <1>         ; 15/09/2015
00005549 0FB6D8              <1>  movzx ebx, al
0000554C C683[05720000]04    <1>         mov     byte [ebx+p.stat-1], 4 ; SSLEEP
00005553 FEC4                <1>  inc   ah
00005555 88A3[E5710000]      <1>  mov   [ebx+p.waitc-1], ah ; wait channel + 1
                             <1>  ;
0000555B 66FF35[66740000]    <1>  push    word [cdev]
                             <1>        ; mov cdev,-(sp) / nothing happened in isintr so
00005562 E833FEFFFF          <1>  call  swap
                             <1>              ; jsr r0,swap / swap out process that needs to sleep
00005567 668F05[66740000]    <1>         pop     word [cdev]
                             <1>        ; mov (sp)+,cdev / restore device
0000556E E807000000          <1>  call  isintr
                             <1>  ; 22/09/2013
00005573 0F8523EBFFFF        <1>  jnz   sysret
                             <1>        ; jsr r0,isintr / check for interrupt of new process
                             <1>                          ; br 2f / yes, return to new user
                             <1>        ; movb (sp)+,r1 / no, r1 = old process number that was
                             <1>                    ; / originally on the wait channel
                             <1>              ; beq 1f / if 0 branch
                             <1>              ; mov $runq+4,r2 / r2 points to lowest priority queue
                             <1>              ; mov $300,*$ps / processor priority = 6
                             <1>        ; jsr r0,putlu / create link to old process number
                             <1>              ; clr *$ps / clear the status; process priority = 0
                             <1>    ;1:
00005579 C3                  <1>  retn
                             <1>        ; rts r0 / return
                             <1>      ;2:
                             <1>         ;;jmpsysret
                             <1>        ; jmp sysret / return to user
                             <1>
                             <1> isintr:
                             <1>  ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ;
                             <1>  ; 09/05/2013 - 30/05/2014
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification !
                             <1>  ; (Process/task switching and quit routine by using
                             <1>  ; Retro UNIX 8086 v1 keyboard interrupt output.))
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ; 'isintr' checks if user interrupt request is enabled
                             <1>  ;  and there is a 'quit' request by user;
                             <1>  ;  otherwise, 'isintr' will return with zf=1 that means
                             <1>  ;  "nothing to do". (20/10/2013)
                             <1>  ;
                             <1>  ; 20/10/2013
0000557A 66833D[B0740000]00  <1>  cmp   word [u.ttyp], 0 ; has process got a tty ?
00005582 7622                <1>  jna   short isintr2 ; retn
                             <1>  ; 03/09/2013
                             <1>  ; (nothing to do)
                             <1>  ;retn
                             <1>  ; 22/09/2013
00005584 66833D[BE740000]00  <1>  cmp   word [u.intr], 0
0000558C 7618                <1>  jna   short isintr2 ; retn
                             <1>  ; 30/05/2014
0000558E 6650                <1>  push  ax
00005590 66A1[C0740000]      <1>  mov   ax, [u.quit]
00005596 6609C0              <1>  or    ax, ax ; 0 ?
00005599 7409                <1>  jz    short isintr1 ; zf = 1
0000559B 6683F8FE            <1>  cmp   ax, 0FFFEh  ; 'ctrl + brk' check
0000559F 7703                <1>  ja    short isintr1 ; 0FFFFh, zf = 0
000055A1 6631C0              <1>  xor   ax, ax ; zf = 1
                             <1> isintr1:
```

```
000055A4 6658                <1>  pop   ax
                             <1> isintr2: ; 22/09/2013
                             <1>  ; zf=1 -> nothing to do
000055A6 C3                  <1>  retn
                             <1>
                             <1>  ; UNIX v1 original 'isintr' routine...
                             <1>        ;mov     r1,-(sp) / put number of wait channel on the stack
                             <1>        ;mov     r2,-(sp) / save r2
                             <1>        ;mov     u.ttyp,r1 / r1 = pointer to buffer of process control
                             <1>         ;                 / typewriter
                             <1>        ;beq     1f / if 0, do nothing except skip return
                             <1>        ;movb    6(r1),r1 / put interrupt char in the tty buffer in r1
                             <1>        ;beq     1f / if its 0 do nothing except skip return
                             <1>        ;cmp     r1,$177 / is interrupt char = delete?
                             <1>        ;bne     3f / no, so it must be a quit (fs)
                             <1>        ;tst     u.intr / yes, value of u.intr determines handling
                             <1>         ;              / of interrupts
                             <1>        ;bne     2f / if not 0, 2f. If zero do nothing.
                             <1>      ;1:
                             <1>        ;tst     (r0)+ / bump r0 past system return (skip)
                             <1>      ;4:
                             <1>        ;mov     (sp)+,r2 / restore r1 and r2
                             <1>        ;mov     (sp)+,r1
                             <1>        ;rts     r0
                             <1>      ;3: / interrupt char = quit (fs)
                             <1>        ;tst     u.quit / value of u.quit determines handling of quits
                             <1>        ;beq     1b / u.quit = 0 means do nothing
                             <1>      ;2: / get here because either u.intr <> 0 or u.qult <> O
                             <1>        ;mov     $tty+6,r1 / move pointer to tty block into r1
                             <1>      ;1: / find process control tty entry in tty block
                             <1>        ;cmp     (r1),u.ttyp / is this the process control tty buffer?
                             <1>        ;beq     1f / block found go to 1f
                             <1>        ;add     $8,r1 / look at next tty block
                             <1>        ;cmp     r1,$tty+[ntty*8]+6 / are we at end of tty blocks
                             <1>        ;blo     1b / no
                             <1>        ;br      4b / no process control tty found so go to 4b
                             <1>      ;1:
                             <1>        ;mov     $240,*$ps / set processor priority to 5
                             <1>        ;movb    -3(r1),0f / load getc call argument; character llst
                             <1>         ;                  / identifier
                             <1>        ;inc     0f / increment
                             <1>      ;1:
                             <1>        ;jsr     r0,getc; 0:.. / erase output char list for control
                             <1>         ;        br 4b / process tty. This prevents a line of stuff
                             <1>         ;             / being typed out after you hit the interrupt
                             <1>         ;             / key
                             <1>        ;br      1b
                                 %include 'u5.s'        ; 03/06/2015
                             <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS5.INC
                             <1> ; Last Modification: 14/11/2015
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
                             <1> ; (v0.1 - Beginning: 11/07/2012)
                             <1> ;
                             <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                             <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                             <1> ; <Bell Laboratories (17/3/1972)>
                             <1> ; <Preliminary Release of UNIX Implementation Document>
                             <1> ;
                             <1> ; Retro UNIX 8086 v1 - U5.ASM (07/08/2013) //// UNIX v1 -> u5.s
                             <1> ;
                             <1> ; ****************************************************************************
                             <1>
                             <1> mget:
                             <1>  ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Get existing or (allocate) a new disk block for file
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    u.fofp (file offset pointer)
                             <1>  ;    inode
                             <1>  ;    u.off (file offset)
                             <1>  ; OUTPUTS ->
                             <1>  ;    r1 (physical block number)
                             <1>  ;    r2, r3, r5 (internal)
                             <1>  ;
                             <1>  ; ((AX = R1)) output
                             <1>  ;    (Retro UNIX Prototype : 05/03/2013 - 14/11/2012, UNIXCOPY.ASM)
                             <1>         ;    ((Modified registers: eDX, eBX, eCX, eSI, eDI, eBP))
                             <1>
                             <1>        ; mov *u.fofp,mq / file offset in mq
                             <1>        ; clr ac / later to be high sig
                             <1>        ; mov $-8,lsh   / divide ac/mq by 256.
                             <1>        ; mov mq,r2
                             <1>        ; bit $10000,i.flgs / lg/sm is this a large or small file
                             <1>        ; bne 4f / branch for large file
                             <1> mget_0:
000055A7 8B35[90740000]      <1>         mov     esi, [u.fofp]
000055AD 0FB65E01            <1>         movzx   ebx, byte [esi+1]
                             <1>         ; BX = r2
000055B1 66F705[56710000]00- <1>         test word [i.flgs], 4096 ; 1000h
```

```
000055B9 10                  <1>
                             <1>                          ; is this a large or small file
000055BA 756F                <1>  jnz   short mget_5 ; 4f ; large file
                             <1>
000055BC F6C3F0              <1>         test    bl, 0F0h ; !0Fh
                             <1>        ; bit $!17,r2
000055BF 7526                <1>  jnz   short mget_2
                             <1>        ; bne 3f / branch if r2 greater than or equal to 16
000055C1 80E30E              <1>         and     bl, 0Eh
                             <1>        ; bic $!16,r2 / clear all bits but bits 1,2,3
000055C4 0FB783[5C710000]    <1>  movzx       eax, word [ebx+i.dskp] ; AX = R1, physical block number
                             <1>        ; mov i.dskp(r2),r1 / r1 has physical block number
000055CB 6609C0              <1>  or    ax, ax
000055CE 7516                <1>  jnz   short mget_1
                             <1>        ; bne 2f / if physical block num is zero then need a new block
                             <1>              ; / for file
000055D0 E8AB000000          <1>  call  alloc
                             <1>        ; jsr r0,alloc / allocate a new block
                             <1>          ; eAX (r1) = Physical block number
000055D5 668983[5C710000]    <1>  mov   [ebx+i.dskp], ax
                             <1>        ; mov r1,i.dskp(r2) / physical block number stored in i-node
000055DC E84C020000          <1>  call  setimod
                             <1>        ; jsr r0,setimod / set inode modified byte (imod)
000055E1 E8B1FEFFFF          <1>  call  clear
                             <1>        ; jsr r0,clear / zero out disk/drum block just allocated
                             <1> mget_1: ; 2:
                             <1>         ; eAX (r1) = Physical block number
000055E6 C3                  <1>  retn
                             <1>        ; rts r0
                             <1> mget_2: ; 3: / adding on block which changes small file to a large file
000055E7 E894000000          <1>  call  alloc
                             <1>        ; jsr r0,alloc / allocate a new block for this file;
                             <1>                      ; / block number in r1
                             <1>         ; eAX (r1) = Physical block number
000055EC E8560C0000          <1>  call  wslot
                             <1>        ; jsr r0,wslot / set up I/O buffer for write, r5 points to
                             <1>                  ; / first data word in buffer
                             <1>         ; eAX (r1) = Physical block number
000055F1 B908000000          <1>  mov   ecx, 8  ; R3, transfer old physical block pointers
                             <1>           ; into new indirect block area for the new
                             <1>           ; large file
000055F6 89DF                <1>  mov   edi, ebx ; r5
000055F8 BE[5C710000]        <1>  mov   esi, i.dskp
                             <1>        ; mov $8.,r3 / next 6 instructions transfer old physical
                             <1>                ; / block pointers
                             <1>        ; mov $i.dskp,r2 / into new indirect block for the new
                             <1>               ; / large file
000055FD 6631C0              <1>  xor   ax, ax ; mov ax, 0
                             <1> mget_3: ;1:
00005600 66A5                <1>  movsw
                             <1>        ; mov (r2),(r5)+
00005602 668946FE            <1>  mov   [esi-2], ax
                             <1>        ; clr (r2)+
00005606 E2F8                <1>  loop  mget_3 ; 1b
                             <1>        ; dec r3
                             <1>        ; bgt 1b
                             <1>
00005608 B1F8                <1>  mov   cl, 256-8
                             <1>        ; mov $256.-8.,r3 / clear rest of data buffer
                             <1> mget_4:; 1
0000560A F366AB              <1>  rep   stosw
                             <1>        ; clr (r5)+
                             <1>        ; dec r3
                             <1>        ; bgt 1b
                             <1>  ; 24/03/2013
                             <1>         ; AX (r1) = Physical block number
0000560D E8510C0000          <1>  call  dskwr
                             <1>        ; jsr r0,dskwr / write new indirect block on disk
                             <1>         ; eAX (r1) = Physical block number
00005612 66A3[5C710000]      <1>  mov   [i.dskp], ax
                             <1>        ; mov r1,i.dskp / put pointer to indirect block in i-node
00005618 66810D[56710000]00- <1>  or    word [i.flgs], 4096 ; 1000h
00005620 10                  <1>
                             <1>        ; bis $10000,i.flgs / set large file bit
                             <1>                         ; / in i.flgs word of i-node
00005621 E807020000          <1>  call  setimod
                             <1>        ; jsr r0,setimod / set i-node modified flag
00005626 E97CFFFFFF          <1>         jmp     mget_0
                             <1>        ; br mget
                             <1>
                             <1> mget_5:  ; 4 ; large file
                             <1>        ; mov $-8,lsh / divide byte number by 256.
                             <1>        ; bic $!776,r2 / zero all bits but 1,2,3,4,5,6,7,8; gives offset
                             <1>                  ; / in indirect block
                             <1>        ; mov r2,-(sp) / save on stack (*)
                             <1>        ; mov mq,r2 / calculate offset in i-node for pointer to proper
                             <1>                        ; / indirect block
                             <1>        ; bic $!16,r2
0000562B 80E3FE              <1>         and     bl, 0FEh ; bh = 0
0000562E 53                  <1>         push    ebx  ; i-node pointer offset in indirect block  (*)
                             <1>         ; 01/03/2013 Max. possible BX (offset) value is 127 (65535/512)
                             <1>  ;          for this file system (offset 128 to 255 not in use)
                             <1>  ; There is always 1 indirect block for this file system
0000562F 0FB705[5C710000]    <1>  movzx       eax, word [i.dskp] ; i.dskp[0]
                             <1>        ; mov i.dskp(r2),r1
00005636 6609C0              <1>  or    ax, ax ; R1
00005639 7515                <1>  jnz   short mget_6 ; 2f
                             <1>        ; bne 2f / if no indirect block exists
0000563B E840000000          <1>  call  alloc
                             <1>        ; jsr r0,alloc / allocate a new block
00005640 66A3[5C710000]      <1>  mov   [i.dskp], ax  ; 03/03/2013
                             <1>        ; mov r1,i.dskp(r2) / put block number of new block in i-node
00005646 E8E2010000          <1>  call  setimod
                             <1>        ; jsr r0,setimod / set i-node modified byte
                             <1>  ; eAX = new block number
0000564B E847FEFFFF          <1>  call  clear
                             <1>        ; jsr r0,clear / clear new block
                             <1> mget_6: ;2
                             <1>  ; 05/03/2013
                             <1>  ; eAX = r1, physical block number (of indirect block)
00005650 E8920B0000          <1>  call  dskrd ; read indirect block
                             <1>        ; jsr r0,dskrd / read in indirect block
00005655 5A                  <1>  pop   edx  ; R2, get offset (*)
                             <1>        ; mov (sp)+,r2 / get offset
                             <1>  ; eAX = r1, physical block number (of indirect block)
00005656 50                  <1>  push  eax ; ** ; 24/03/2013
                             <1>        ; mov r1,-(sp) / save block number of indirect block on stack
                             <1>  ; eBX (r5) = pointer to buffer (indirect block)
00005657 01D3                <1>  add   ebx, edx ; / r5 points to first word in indirect block, r2
                             <1>        ; add r5,r2 / r5 points to first word in indirect block, r2
                             <1>                    ; / points to location of inter
00005659 0FB703              <1>  movzx       eax, word [ebx] ; put physical block no of block
                             <1>                    ; in file sought in R1 (AX)
                             <1>        ; mov (r2),r1 / put physical block no of block in file
                             <1>                        ; / sought in r1
0000565C 6609C0              <1>  or    ax, ax
0000565F 751D                <1>         jnz  short mget_7 ; 2f
                             <1>        ; bne 2f / if no block exists
00005661 E81A000000          <1>  call  alloc
                             <1>        ; jsr r0,alloc / allocate a new block
00005666 668903              <1>  mov   [ebx], ax ; R1
                             <1>        ; mov r1,(r2) / put new block number into proper location in
                             <1>                      ; / indirect block
00005669 5A                  <1>  pop   edx ; ** ; 24/03/2013
                             <1>        ; mov (sp)+,r1 / get block number of indirect block
0000566A 52                  <1>  push  edx ; ** ; 31/07/2013
0000566B 50                  <1>  push  eax ; * ; 24/03/2013, 31/07/2013 (new block number)
0000566C 89D0                <1>  mov   eax, edx ; 24/03/2013
                             <1>        ; mov (r2),-(sp) / save block number of new block
                             <1>  ; eAX (r1) = physical block number (of indirect block)
0000566E E8D40B0000          <1>  call  wslot
                             <1>        ; jsr r0,wslot
                             <1>         ; eAX (r1) = physical block number
                             <1>  ; eBX (r5) = pointer to buffer (indirect block)
00005673 E8EB0B0000          <1>  call  dskwr
                             <1>  ; eAX = r1 = physical block number (of indirect block)
                             <1>        ; jsr r0,dskwr / write newly modified indirect block
                             <1>                   ; / back out on disk
00005678 58                  <1>  pop   eax ; *  ; 31/07/2013
                             <1>        ; mov (sp),r1 / restore block number of new block
                             <1>  ; eAX (r1) = physical block number of new block
00005679 E819FEFFFF          <1>  call  clear
                             <1>        ; jsr r0,clear / clear new block
                             <1> mget_7: ; 2
0000567E 5A                  <1>  pop   edx ; **
                             <1>        ; tst (sp)+ / bump stack pointer
                             <1>  ; eAX (r1) = Block number of new block
0000567F C3                  <1>  retn
                             <1>        ; rts r0
                             <1>
                             <1> alloc:
                             <1>  ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 01/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; get a free block and
                             <1>  ; set the corresponding bit in the free storage map
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    cdev (current device)
                             <1>  ;    r2
                             <1>  ;    r3
                             <1>  ; OUTPUTS ->
                             <1>  ;    r1 (physical block number of block assigned)
                             <1>  ;    smod, mmod, systm (super block), mount (mountable super block)
                             <1>  ;
                             <1>  ; ((AX = R1)) output
                             <1>  ;     (Retro UNIX Prototype : 14/11/2012 - 21/07/2012, UNIXCOPY.ASM)
                             <1>         ;    ((Modified registers: DX, CX))
                             <1>
                             <1>        ;mov r2,-(sp) / save r2, r3 on stack
                             <1>        ;mov r3,-(sp)
                             <1>  ;push       ecx
00005680 53                  <1>  push  ebx ; R2
                             <1>  ;push       edx ; R3
```

```
00005681 BB[287D0000]        <1>  mov   ebx, systm ; SuperBlock
                             <1>        ; mov $systm,r2 / start of inode and free storage map for drum
00005686 803D[66740000]00    <1>  cmp   byte [cdev], 0
                             <1>        ; tst cdev
0000568D 7605                <1>  jna   short alloc_1
                             <1>        ; beq 1f / drum is device
0000568F BB[307F0000]        <1>  mov   ebx, mount
                             <1>        ; mov $mount,r2 / disk or tape is device, start of inode and
                             <1>                    ; / free storage map
                             <1> alloc_1: ; 1
00005694 668B0B              <1>         mov  cx, [ebx]
                             <1>        ; mov (r2)+,r1 / first word contains number of bytes in free
                             <1>                   ; / storage map
00005697 66C1E103            <1>  shl   cx, 3
                             <1>        ; asl r1 / multiply r1 by eight gives
                             <1>        ; number of blocks in device
                             <1>        ; asl r1
                             <1>        ; asl r1
                             <1>  ;; push cx ;; 01/08/2013
                             <1>        ; mov r1,-(sp) / save # of blocks in device on stack
0000569B 31C0                <1>  xor   eax, eax ; 0
                             <1>        ; clr r1 / r1 contains bit count of free storage map
                             <1> alloc_2: ; 1
0000569D 43                  <1>  inc   ebx ; 18/8/2012
0000569E 43                  <1>  inc   ebx ;
0000569F 668B13              <1>  mov   dx, [ebx]
                             <1>        ; mov (r2)+,r3 / word of free storage map in r3
000056A2 6609D2              <1>  or    dx, dx
000056A5 750E                <1>  jnz   short alloc_3 ; 1f
                             <1>        ; bne 1f / branch if any free blocks in this word
000056A7 6683C010            <1>  add   ax, 16
                             <1>        ; add $16.,r1
000056AB 6639C8              <1>  cmp   ax, cx
                             <1>        ; cmp r1 ,(sp) / have we examined all free storage bytes
000056AE 72ED                <1>  jb    short alloc_2
                             <1>        ; blo 1b
                             <1>  ; 14/11/2015
                             <1>  ; Note: If the super block buffer has wrong content (zero bytes)
                             <1>  ;     because of a (DMA or another) r/w error,
                             <1>  ;     we will be here, at 'jmp panic' code address,
                             <1>  ;     even if the (disk) file system space is not full !!!
                             <1>  ;     (cx = 0)
                             <1>  ;
000056B0 E95DE2FFFF          <1>  jmp     panic
                             <1>        ; jmp panic / found no free storage
                             <1> alloc_3: ; 1
000056B5 66D1EA              <1>  shr   dx, 1
                             <1>        ; asr r3 / find a free block
000056B8 7204                <1>  jc    short alloc_4 ; 1f
                             <1>        ; bcs 1f / branch when free block found; bit for block k
                             <1>               ; / is in byte k/8 / in bit k (mod 8)
000056BA 6640                <1>  inc   ax
                             <1>        ; inc r1 / increment bit count in bit k (mod8)
000056BC EBF7                <1>  jmp   short alloc_3
                             <1>        ; br 1b
                             <1> alloc_4: ; 1:
                             <1>  ;; pop cx ;; 01/08/2013
                             <1>        ; tst (sp)+ / bump sp
                             <1>  ; 02/04/2013
000056BE E829000000          <1>  call  free3
                             <1>        ; jsr r0,3f / have found a free block
                             <1>  ; 21/8/2012
000056C3 66F7D2              <1>  not   dx ; masking bit is '0' and others are '1'
000056C6 662113              <1>  and   [ebx], dx   ;; 0 -> allocated
                             <1>        ; bic r3,(r2) / set bit for this block
                             <1>                    ; / i.e. assign block
                             <1>        ; br 2f
000056C9 EB09                <1>  jmp   short alloc_5
                             <1>
                             <1> free:
                             <1>  ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 07/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; calculates byte address and bit position for given block number
                             <1>  ; then sets the corresponding bit in the free storage map
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - block number for a block structured device
                             <1>  ;    cdev - current device
                             <1>  ; OUTPUTS ->
                             <1>  ;    free storage map is updated
                             <1>  ;    smod is incremented if cdev is root device (fixed disk)
                             <1>  ;    mmod is incremented if cdev is a removable disk
                             <1>  ;
                             <1>  ;  (Retro UNIX Prototype : 01/12/2012, UNIXCOPY.ASM)
                             <1>         ;  ((Modified registers: DX, CX))
                             <1>
                             <1>        ;mov r2,-(sp) / save r2, r3
                             <1>        ;mov r3,-(sp)
                             <1>  ;push       ecx
000056CB 53                  <1>  push  ebx ; R2
                             <1>  ;push       edx ; R3
```

```
                              <1>
000056CC E81B000000           <1>         call    free3
                              <1>        ; jsr r0,3f  / set up bit mask and word no.
                              <1>                   ; / in free storage map for block
000056D1 660913               <1>  or    [ebx], dx
                              <1>        ; bis r3, (r2) / set free storage block bit;
                              <1>               ;  / indicates free block
                              <1>  ; 0 -> allocated, 1 -> free
                              <1>
                              <1> alloc_5:
                              <1>  ; 07/04/2013
                              <1> free_1: ; 2:
                              <1>  ; pop      edx
                              <1>        ; mov (sp)+,r3 / restore r2, r3
000056D4 5B                   <1>  pop   ebx
                              <1>        ; mov (sp)+,r2
                              <1>  ; pop ecx
000056D5 803D[66740000]00     <1>  cmp   byte [cdev], 0
                              <1>        ; tst cdev / cdev = 0, block structured, drum;
                              <1>              ; / cdev = 1, mountable device
000056DC 7707                 <1>  ja    short alloc_6 ; 1f
                              <1>        ; bne 1f
                              <1>  ;mov  byte [smod], 1
000056DE FE05[75740000]       <1>  inc   byte [smod]
                              <1>        ; incb smod / set super block modified for drum
                              <1>  ; eAX (r1) = block number
000056E4 C3                   <1>  retn
                              <1>        ; rts r0
                              <1> free_2:
                              <1> alloc_6: ; 1:
                              <1>  ;mov  byte [mmod], 1
000056E5 FE05[76740000]       <1>  inc   byte [mmod]
                              <1>        ; incb     mmod
                              <1>           ; / set super block modified for mountable device
                              <1>  ; eAX (r1) = block number
000056EB C3                   <1>  retn
                              <1>        ; rts r0
                              <1> free3:
                              <1>  ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 02/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
                              <1>  ;
                              <1>  ; free3 is called from 'alloc' and 'free' procedures
                              <1>  ;
                              <1> alloc_free_3: ; 3
000056EC 66BA0100             <1>  mov   dx, 1
000056F0 88C1                 <1>  mov   cl, al
                              <1>        ; mov r1,r2 / block number, k, = 1
000056F2 80E10F               <1>  and   cl, 0Fh  ; 0Fh <-- (k) mod 16
                              <1>        ; bic $!7,r2 / clear all bits but 0,1,2; r2 = (k) mod (8)
000056F5 7403                 <1>  jz    short free4
                              <1>        ; bisb 2f(r2),r3 / use mask to set bit in r3 corresponding to
                              <1>                   ; / (k) mod 8
000056F7 66D3E2               <1>  shl   dx, cl
                              <1> free4:
000056FA 0FB7D8               <1>  movzx      ebx, ax
                              <1>        ; mov r1,r2 / divide block number by 16
000056FD 66C1EB04             <1>  shr   bx, 4
                              <1>        ; asr r2
                              <1>        ; asr r2
                              <1>        ; asr r2
                              <1>        ; asr r2
                              <1>        ; bcc 1f / branch if bit 3 in r1 was 0 i.e.,
                              <1>              ; / bit for block is in lower half of word
                              <1>        ; swab r3 / swap bytes in r3; bit in upper half of word in free
                              <1>                ; / storage map
                              <1> alloc_free_4: ; 1
00005701 66D1E3               <1>  shl   bx, 1
                              <1>        ; asl r2 / multiply block number by 2; r2 = k/8
00005704 81C3[2A7D0000]       <1>  add   ebx, systm+2 ; SuperBlock+2
                              <1>        ; add $systm+2,r2 / address of word of free storage map for drum
                              <1>                    ; / with block bit in it
0000570A 803D[66740000]00     <1>  cmp   byte [cdev], 0
                              <1>        ; tst cdev
00005711 7606                 <1>  jna   short alloc_free_5
                              <1>        ; beq 1f / cdev = 0 indicates device is drum
00005713 81C308020000         <1>  add   ebx, mount - systm
                              <1>        ; add $mount-systm,r2 / address of word of free storage map for
                              <1>                      ; / mountable device with bit of block to be
                              <1>                      ; / freed
                              <1> alloc_free_5: ; 1
00005719 C3                   <1>  retn
                              <1>        ; rts r0 / return to 'free'
                              <1>        ; 2
                              <1>          ; .byte   1,2,4,10,20,40,100,200 / masks for bits 0,...,7
                              <1>
                              <1> iget:
                              <1>  ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 07/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)
                              <1>  ;
                              <1>  ; get a new i-node whose i-number in r1 and whose device is in cdev
                              <1>  ;
                              <1>  ; ('iget' returns current i-number in r1, if input value of r1 is 0)
                              <1>  ;
                              <1>  ; INPUTS ->
                              <1>  ;    ii - current i-number, rootdir
                              <1>  ;    cdev - new i-node device
                              <1>  ;    idev - current i-node device
                              <1>  ;    imod - current i-node modified flag
                              <1>  ;    mnti - cross device file i-number
                              <1>  ;    r1 - i-numbe rof new i-node
                              <1>  ;    mntd - mountable device number
                              <1>  ;
                              <1>  ; OUTPUTS ->
                              <1>  ;    cdev, idev, imod, ii, r1
                              <1>  ;
                              <1>  ; ((AX = R1)) input/output
                              <1>  ;
                              <1>  ;  (Retro UNIX Prototype : 14/07/2012 - 18/11/2012, UNIXCOPY.ASM)
                              <1>        ;  ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
                              <1>
0000571A 8A15[66740000]       <1>  mov   dl, [cdev] ; 18/07/2013
00005720 8A35[64740000]       <1>  mov   dh, [idev] ; 07/08/2013
                              <1>  ;
00005726 663B05[62740000]     <1>  cmp   ax, [ii]
                              <1>        ; cmp r1,ii / r1 = i-number of current file
0000572D 7504                 <1>  jne   short iget_1
                              <1>        ; bne 1f
0000572F 38F2                 <1>  cmp   dl, dh
                              <1>        ; cmp idev,cdev
                              <1>                ; / is device number of i-node = current device
00005731 7476                 <1>         je      short iget_5
                              <1>        ; beq 2f
                              <1> iget_1: ; 1:
00005733 30DB                 <1>  xor   bl, bl
00005735 381D[74740000]       <1>  cmp   [imod], bl ; 0
                              <1>        ; tstb imod / has i-node of current file
                              <1>                ; / been modified i.e., imod set
0000573B 762D                 <1>  jna   short iget_2
                              <1>        ; beq 1f
0000573D 881D[74740000]       <1>  mov   [imod], bl ; 0
                              <1>        ;  clrb     imod / if it has,
                              <1>                 ; / we must write the new i-node out on disk
00005743 6650                 <1>  push  ax
                              <1>        ; mov r1,-(sp)
                              <1>  ;mov  dl, [cdev]
00005745 6652                 <1>  push  dx
                              <1>        ; mov cdev,-(sp)
00005747 66A1[62740000]       <1>  mov   ax, [ii]
                              <1>        ; mov ii,r1
                              <1>  ;mov  dh, [idev]
0000574D 8835[66740000]       <1>  mov   [cdev], dh
                              <1>        ; mov idev,cdev
00005753 FEC3                 <1>  inc   bl ; 1
                              <1>  ; 31/07/2013
00005755 881D[FC740000]       <1>  mov     [rw], bl ; 1 == write
                              <1>  ;;28/07/2013 rw -> u.rw
                              <1>         ;;mov   [u.rw], bl ; 1 == write
0000575B E84A000000           <1>  call  icalc
                              <1>        ; jsr r0,icalc; 1
00005760 665A                 <1>  pop   dx
00005762 8815[66740000]       <1>  mov   [cdev], dl
                              <1>        ; mov (sp)+,cdev
00005768 6658                 <1>  pop   ax
                              <1>        ; mov (sp)+,r1
                              <1> iget_2: ; 1:
0000576A 6621C0               <1>  and   ax, ax
                              <1>        ; tst r1 / is new i-number non zero
0000576D 7434                 <1>  jz    short iget_4 ; 2f
                              <1>        ; beq 2f / branch if r1=0
                              <1>
                              <1>  ; mov       dl, [cdev]
0000576F 08D2                 <1>  or    dl, dl
                              <1>        ; tst cdev / is the current device number non zero
                              <1>               ; / (i.e., device =/ drum)
00005771 7517                 <1>  jnz   short iget_3 ;  1f
                              <1>        ; bne 1f / branch 1f cdev =/ 0  ;; (cdev != 0)
00005773 663B05[6C740000]     <1>  cmp   ax, [mnti]
                              <1>        ; cmp r1,mnti / mnti is the i-number of the cross device
                              <1>                 ; / file (root directory of mounted device)
0000577A 750E                 <1>  jne   short iget_3 ; 1f
                              <1>        ; bne 1f
                              <1>         ;mov    bl, [mntd]
0000577C FEC2                 <1>  inc   dl ; mov dl, 1 ; 17/07/2013
0000577E 8815[66740000]       <1>         mov  [cdev], dl ; 17/07/2013 - 09/07/2013
                              <1>        ; mov mntd,cdev / make mounted device the current device
00005784 66A1[70740000]       <1>  mov   ax, [rootdir]
                              <1>        ; mov rootdir,r1
                              <1> iget_3: ; 1:
0000578A 66A3[62740000]       <1>  mov   [ii], ax
                              <1>        ; mov r1,ii
00005790 8815[64740000]       <1>  mov   [idev], dl ; cdev
                              <1>        ; mov cdev,idev
00005796 30DB                 <1>  xor   bl, bl
                              <1>         ; 31/07/2013
```

```
00005798 881D[FC740000]        <1>  mov     [rw], bl ; 0 == read
                               <1>  ;;28/07/2013 rw -> u.rw
                               <1>         ;;mov   [u.rw], bl ; 0 = read
0000579E E807000000            <1>  call  icalc
                               <1>        ; jsr r0,icalc; 0 / read in i-node ii
                               <1> iget_4: ; 2:
000057A3 66A1[62740000]        <1>  mov   ax, [ii]
                               <1>        ; mov ii,r1
                               <1> iget_5:
000057A9 C3                    <1>  retn
                               <1>        ; rts r0
                               <1>
                               <1> icalc:
                               <1>  ; 02/07/2015
                               <1>  ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
                               <1>  ; 07/04/2013 - 31/07/2013 (Retro UNIX 8086 v1)
                               <1>  ;
                               <1>  ; calculate physical block number from i-number then
                               <1>  ; read or write that block
                               <1>  ;
                               <1>  ; 'icalc' is called from 'iget'
                               <1>  ;
                               <1>  ; for original unix v1:
                               <1>  ; / i-node i is located in block (i+31.)/16. and begins 32.*
                               <1>        ; / (i+31.) mod 16. bytes from its start
                               <1>  ;
                               <1>  ; for retro unix 8086 v1:
                               <1>  ;  i-node is located in block (i+47)/16 and
                               <1>  ;  begins 32*(i+47) mod 16 bytes from its start
                               <1>  ;
                               <1>  ; INPUTS ->
                               <1>  ;    r1 - i-number of i-node
                               <1>  ;
                               <1>  ; OUTPUTS ->
                               <1>  ;    inode r/w
                               <1>  ;
                               <1>  ; ((AX = R1)) input
                               <1>  ;
                               <1>  ;  (Retro UNIX Prototype : 14/07/2012 - 18/11/2012, UNIXCOPY.ASM)
                               <1>         ;  ((Modified registers: eAX, eDX, eCX, eBX, eSI, eDI, eBP))
                               <1>  ;
000057AA 0FB7D0                <1>  movzx edx, ax
000057AD 6683C22F              <1>  add   dx, 47
000057B1 89D0                  <1>  mov   eax, edx
                               <1>  ;add  ax, 47      ; add 47 to inode number
                               <1>        ;  add      $31.,r1 / add 31. to i-number
000057B3 50                    <1>  push  eax
                               <1>        ; mov r1,-(sp) / save i+31. on stack
000057B4 66C1E804              <1>  shr   ax, 4
                               <1>        ; asr r1 / divide by 16.
                               <1>        ; asr r1
                               <1>        ; asr r1
                               <1>        ; asr r1 / r1 contains block number of block
                               <1>               ; / in which i-node exists
000057B8 E82A0A0000            <1>  call  dskrd
                               <1>        ; jsr r0,dskrd / read in block containing i-node i.
                               <1>  ; 31/07/2013
000057BD 803D[FC740000]00      <1>         cmp     byte [rw], 0 ; Retro Unix 8086 v1 feature !
                               <1>  ;; 28/07/2013 rw -> u.rw
                               <1>         ;;cmp     byte [u.rw], 0 ; Retro Unix 8086 v1 feature !
                               <1>        ; tst (r0)
000057C4 7605                  <1>  jna   short icalc_1
                               <1>        ; beq 1f / branch to wslot when argument
                               <1>               ; / in icalc call = 1
                               <1>  ; eAX = r1  = block number
000057C6 E87C0A0000            <1>  call  wslot
                               <1>        ; jsr r0,wslot / set up data buffer for write
                               <1>                   ; / (will be same buffer as dskrd got)
                               <1>  ; eBX = r5 points to first word in data area for this block
                               <1> icalc_1: ; 1:
000057CB 5A                    <1>  pop   edx
000057CC 83E20F                <1>  and   edx, 0Fh ; (i+47) mod 16
                               <1>        ; bic $!17,(sp) / zero all but last 4 bits;
                               <1>                    ; / gives (i+31.) mod 16
000057CF C1E205                <1>  shl   edx, 5
                               <1>  ; eDX = 32 * ((i+47) mod 16)
000057D2 89DE                  <1>  mov   esi, ebx  ; ebx points 1st word of the buffer
000057D4 01D6                  <1>  add   esi, edx  ; edx is inode offset in the buffer
                               <1>              ; eSI (r5) points to first word in i-node i.
                               <1>        ; mov (sp)+,mq / calculate offset in data buffer;
                               <1>                   ; / 32.*(i+31.)mod16
                               <1>        ; mov $5,lsh / for i-node i.
                               <1>        ; add mq,r5 / r5 points to first word in i-node i.
000057D6 BF[56710000]          <1>  mov   edi, inode
                               <1>        ; mov $inode,r1 / inode is address of first word
                               <1>                    ; / of current i-node
000057DB B908000000            <1>  mov   ecx, 8 ; 02/07/2015(32 bit modification)
                               <1>        ; mov $16.,r3
                               <1>        ; 31/07/2013
000057E0 382D[FC740000]        <1>        cmp     [rw], ch ; 0  ;; Retro Unix 8086 v1 feature !
                               <1>        ;;28/07/2013 rw -> u.rw
                               <1>        ;;cmp    [u.rw], ch ; 0  ;; Retro Unix 8086 v1 feature !
```

```
                             <1>        ; tst (r0)+ / branch to 2f when argument in icalc call = 0
000057E6 760A                <1>  jna   short icalc_3
                             <1>        ; beq 2f / r0 now contains proper return address
                             <1>                ; / for rts r0
                             <1> icalc_2: ; 1:
000057E8 87F7                <1>  xchg  esi, edi
                             <1>  ; overwrite old i-node (in buffer to be written)
000057EA F3A5                <1>  rep   movsd
                             <1>        ; mov (r1)+,(r5)+ / over write old i-node
                             <1>        ; dec r3
                             <1>        ; bgt 1b
000057EC E8720A0000          <1>  call  dskwr
                             <1>        ; jsr r0,dskwr / write inode out on device
000057F1 C3                  <1>  retn
                             <1>        ; rts r0
                             <1> icalc_3: ; 2:
                             <1>  ; copy new i-node into inode area of (core) memory
000057F2 F3A5                <1>  rep   movsd
                             <1>        ; mov (r5)+,(r1)+ / read new i-node into
                             <1>                         ; / "inode" area of core
                             <1>        ; dec r3
                             <1>        ; bgt 2b
000057F4 C3                  <1>  retn
                             <1>        ; rts r0
                             <1>
                             <1> access:
                             <1>  ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 24/04/2013 - 29/04/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; check whether user is owner of file or user has read or write
                             <1>  ; permission (based on i.flgs).
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - i-number of file
                             <1>  ;    u.uid
                             <1>  ; arg0 -> (owner flag mask)
                             <1>  ;     Retro UNIX 8086 v1 feature -> owner flag mask in DL (DX)
                             <1>  ; OUTPUTS ->
                             <1>  ;    inode (or jump to error)
                             <1>  ;
                             <1>  ; ((AX = R1)) input/output
                             <1>  ;
                             <1>         ;  ((Modified registers: eCX, eBX, eDX, eSI, eDI, eBP))
                             <1>  ;
000057F5 6652                <1>  push  dx  ; save flags (DL)
000057F7 E81EFFFFFF          <1>  call  iget
                             <1>        ; jsr r0,iget / read in i-node for current directory
                             <1>                   ; / (i-number passed in r1)
000057FC 8A0D[56710000]      <1>  mov   cl, [i.flgs]
                             <1>        ; mov i.flgs,r2
00005802 665A                <1>  pop   dx  ; restore flags (DL)
00005804 8A35[C6740000]      <1>  mov   dh, [u.uid]
0000580A 3A35[59710000]      <1>  cmp   dh, [i.uid]
                             <1>        ; cmpb i.uid,u.uid / is user same as owner of file
00005810 7503                <1>  jne   short access_1
                             <1>        ; bne 1f / no, then branch
00005812 C0E902              <1>  shr   cl, 2
                             <1>        ; asrb r2 / shift owner read write bits into non owner
                             <1>                ; / read/write bits
                             <1>        ; asrb r2
                             <1> access_1: ; 1:
00005815 20D1                <1>  and   cl, dl
                             <1>        ; bit r2,(r0)+ / test read-write flags against argument
                             <1>                   ; / in access call
00005817 7513                <1>  jnz   short access_2
                             <1>        ; bne 1f
00005819 08F6                <1>  or    dh, dh      ; super user  (root) ?
                             <1>        ; tstb u.uid
0000581B 740F                <1>  jz    short access_2 ; yes, super user
                             <1>  ;jnz  error
                             <1>        ; beq 1f
                             <1>        ; jmp error
0000581D C705[CF740000]0B00- <1>  mov   dword [u.error], ERR_FILE_ACCESS
00005825 0000                <1>
                             <1>               ; 'permission denied !' error
00005827 E950E8FFFF          <1>  jmp   error
                             <1>
                             <1> access_2: ; 1:
                             <1>  ; DL = flags
0000582C C3                  <1>  retn
                             <1>        ; rts r0
                             <1>
                             <1> setimod:
                             <1>  ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 09/04/2013 - 31/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'setimod' sets byte at location 'imod' to 1; thus indicating that
                             <1>  ; the inode has been modified. Also puts the time of modification
                             <1>  ; into the inode.
                             <1>  ;
                             <1>  ;  (Retro UNIX Prototype : 14/07/2012 - 23/02/2013, UNIXCOPY.ASM)
                             <1>         ;  ((Modified registers: eDX, eCX, eBX))
                             <1>  ;
                             <1>
                             <1>  ; push      edx
0000582D 50                  <1>  push  eax
                             <1>
0000582E C605[74740000]01    <1>  mov   byte [imod], 1
                             <1>        ; movb $1,imod / set current i-node modified bytes
                             <1>  ; Erdogan Tan 14-7-2012
00005835 E87DE3FFFF          <1>  call  epoch
                             <1>         ; mov s.time,i.mtim
                             <1>                  ; / put present time into file modified time
                             <1>         ; mov s.time+2,i.mtim+2
                             <1>
0000583A A3[70710000]        <1>  mov   [i.mtim], eax
                             <1>
                             <1>  ; Retro UNIX 386 v1 modification ! (cmp)
                             <1>  ; Retro UNIX 8086 v1 modification ! (test)
0000583F 833D[6C710000]00    <1>  cmp   dword [i.ctim], 0
00005846 7505                <1>  jnz   short setimod_ok
                             <1>
00005848 A3[6C710000]        <1>  mov   [i.ctim], eax
                             <1>
                             <1> setimod_ok: ; 31/07/2013
0000584D 58                  <1>  pop   eax
                             <1>  ;pop  edx
                             <1>
0000584E C3                  <1>  retn
                             <1>        ; rts r0
                             <1>
                             <1> itrunc:
                             <1>  ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 23/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'itrunc' truncates a file whose i-number is given in r1
                             <1>  ;  to zero length.
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - i-number of i-node
                             <1>  ;    i.dskp - pointer to contents or indirect block in an i-node
                             <1>  ;    i.flgs - large file flag
                             <1>  ;    i.size - size of file
                             <1>  ;
                             <1>  ; OUTPUTS ->
                             <1>  ;    i.flgs - large file flag is cleared
                             <1>  ;    i.size - set to 0
                             <1>  ;    i.dskp .. i.dskp+16 - entire list is cleared
                             <1>  ;    setimod - set to indicate i-node has been modified
                             <1>  ;    r1 - i-number of i-node
                             <1>  ;
                             <1>  ; ((AX = R1)) input/output
                             <1>  ;
                             <1>  ;  (Retro UNIX Prototype : 01/12/2012 - 10/03/2013, UNIXCOPY.ASM)
                             <1>         ;  ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
                             <1>
0000584F E8C6FEFFFF          <1>  call  iget
                             <1>        ; jsr r0,iget
00005854 BE[5C710000]        <1>  mov   esi, i.dskp
                             <1>        ; mov $i.dskp,r2 / address of block pointers in r2
00005859 31C0                <1>  xor   eax, eax
                             <1> itrunc_1: ; 1:
0000585B 66AD                <1>  lodsw
                             <1>        ; mov (r2)+,r1 / move physical block number into r1
0000585D 6609C0              <1>  or    ax, ax
00005860 743B                <1>  jz    short itrunc_5
                             <1>        ; beq 5f
00005862 56                  <1>  push  esi
                             <1>        ; mov r2,-(sp)
00005863 66F705[56710000]00- <1>  test    word [i.flgs], 1000h
0000586B 10                  <1>
                             <1>        ; bit $10000,i.flgs / test large file bit?
0000586C 7429                <1>  jz    short itrunc_4
                             <1>        ; beq 4f / if clear, branch
0000586E 50                  <1>  push  eax
                             <1>        ; mov r1,-(sp) / save block number of indirect block
0000586F E873090000          <1>  call  dskrd
                             <1>        ; jsr r0,dskrd / read in block, 1st data word
                             <1>                   ; / pointed to by r5
                             <1>  ; eBX = r5 = Buffer data address (the 1st word)
00005874 B900010000          <1>  mov   ecx, 256
                             <1>        ; mov $256.,r3 / move word count into r3
00005879 89DE                <1>  mov   esi, ebx
                             <1> itrunc_2: ; 2:
0000587B 66AD                <1>  lodsw
                             <1>        ; mov (r5)+,r1 / put 1st data word in r1;
                             <1>                   ; / physical block number
0000587D 6621C0              <1>  and   ax, ax
00005880 7409                <1>  jz    short itrunc_3
                             <1>        ; beq 3f / branch if zero
                             <1>  ;push ecx
00005882 6651                <1>  push  cx
                             <1>        ; mov r3,-(sp) / save r3, r5 on stack
                             <1>  ;push esi
                             <1>        ; mov r5,-(sp)
00005884 E842FEFFFF          <1>  call  free
                             <1>        ; jsr r0,free / free block in free storage map
                             <1>  ;pop  esi
                             <1>        ; mov(sp)+,r5
00005889 6659                <1>  pop   cx
                             <1>  ;pop  ecx
                             <1>        ; mov (sp)+,r3
                             <1> itrunc_3: ; 3:
0000588B E2EE                <1>  loop  itrunc_2
                             <1>        ; dec r3 / decrement word count
                             <1>        ; bgt 2b / branch if positive
0000588D 58                  <1>  pop   eax
                             <1>        ; mov (sp)+,r1 / put physical block number of
                             <1>                   ; / indirect block
                             <1>  ; 01/08/2013
0000588E 668125[56710000]FF- <1>         and     word [i.flgs], 0EFFFh ; 1110111111111111b
00005896 EF                  <1>
                             <1> itrunc_4: ; 4:
00005897 E82FFEFFFF          <1>  call  free
                             <1>        ; jsr r0,free / free indirect block
0000589C 5E                  <1>  pop   esi
                             <1>        ; mov (sp)+,r2
                             <1> itrunc_5: ; 5:
0000589D 81FE[6C710000]      <1>  cmp   esi, i.dskp+16
                             <1>        ; cmp r2,$i.dskp+16.
000058A3 72B6                <1>  jb    short itrunc_1
                             <1>        ; bne 1b / branch until all i.dskp entries check
                             <1>  ; 01/08/2013
                             <1>  ;and     word [i.flgs], 0EFFFh ; 1110111111111111b
                             <1>        ; bic $10000,i.flgs / clear large file bit
000058A5 BF[5C710000]        <1>  mov   edi, i.dskp
000058AA 66B90800            <1>  mov   cx, 8
000058AE 6631C0              <1>  xor   ax, ax
000058B1 66A3[5A710000]      <1>  mov   [i.size], ax ; 0
                             <1>        ; clr i.size / zero file size
000058B7 F366AB              <1>  rep   stosw
                             <1>        ; jsr r0,copyz; i.dskp; i.dskp+16.
                             <1>                 ; / zero block pointers
000058BA E86EFFFFFF          <1>  call  setimod
                             <1>        ; jsr r0,setimod / set i-node modified flag
000058BF 66A1[62740000]      <1>  mov   ax, [ii]
                             <1>        ; mov ii,r1
000058C5 C3                  <1>  retn
                             <1>        ; rts r0
                             <1>
                             <1> imap:
                             <1>  ; 03/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 26/04/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'imap' finds the byte in core (superblock) containing
                             <1>  ; allocation bit for an i-node whose number in r1.
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - contains an i-number
                             <1>  ;    fsp - start of table containing open files
                             <1>  ;
                             <1>  ; OUTPUTS ->
                             <1>  ;    r2 - byte address of byte with the allocation bit
                             <1>  ;    mq - a mask to locate the bit position.
                             <1>  ;       (a 1 is in calculated bit posisiton)
                             <1>  ;
                             <1>  ; ((AX = R1)) input/output
                             <1>  ; ((DL/DX = MQ)) output
                             <1>  ; ((BX = R2)) output
                             <1>  ;
                             <1>  ;    (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)
                             <1>         ;    ((Modified registers: eDX, eCX, eBX, eSI))
                             <1>  ;
                             <1>        ; / get the byte that has the allocation bit for
                             <1>        ; / the i-number contained in r1
                             <1>  ;mov  dx, 1
000058C6 B201                <1>  mov   dl, 1
                             <1>        ; mov $1,mq / put 1 in the mq
000058C8 0FB7D8              <1>  movzx ebx, ax
                             <1>        ; mov r1,r2 / r2 now has i-number whose byte
                             <1>                  ; / in the map we must find
000058CB 6683EB29            <1>  sub   bx, 41
                             <1>        ; sub $41.,r2 / r2 has i-41
000058CF 88D9                <1>  mov   cl, bl
                             <1>        ; mov r2,r3 / r3 has i-41
000058D1 80E107              <1>  and   cl, 7
                             <1>        ; bic $!7,r3 / r3 has (i-41) mod 8 to get
                             <1>                 ; / the bit position
000058D4 7402                <1>  jz    short imap1
                             <1>  ;shl  dx, cl
000058D6 D2E2                <1>  shl   dl, cl
                             <1>        ; mov r3,lsh / move the 1 over (i-41) mod 8 positions
                             <1> imap1:                ; / to the left to mask the correct bit
000058D8 66C1EB03            <1>  shr   bx, 3
                             <1>        ; asr r2
                             <1>        ; asr r2
                             <1>        ; asr r2 / r2 has (i-41) base 8 of the byte number
                             <1>               ; / from the start of the map
                             <1>        ; mov r2,-(sp) / put (i-41) base 8 on the stack
000058DC BE[287D0000]        <1>  mov   esi, systm
                             <1>        ; mov $systm,r2 / r2 points to the in-core image of
                             <1>                    ; / the super block for drum
                             <1>  ;cmp  word [cdev], 0
000058E1 803D[66740000]00    <1>  cmp   byte [cdev], 0
                             <1>        ; tst cdev / is the device the disk
000058E8 7606                <1>  jna   short imap2
                             <1>        ; beq 1f / yes
000058EA 81C608020000        <1>  add   esi, mount - systm
                             <1>        ; add $mount-systm,r2 / for mounted device,
                             <1>              ; / r2 points to 1st word of its super block
                             <1> imap2: ; 1:
000058F0 66031E              <1>  add   bx, [esi] ;; add free map size to si
                             <1>        ; add (r2)+,(sp) / get byte address of allocation bit
000058F3 6683C304            <1>  add   bx, 4
000058F7 01F3                <1>  add   ebx, esi
                             <1>              ; add (sp)+,r2 / ?
                             <1>  ;add  ebx, 4 ;; inode map offset in superblock
                             <1>              ;; (2 + free map size + 2)
                             <1>        ; add $2,r2 / ?
                             <1>  ; DL/DX (MQ) has a 1 in the calculated bit position
                             <1>         ; BX (R2) has byte address of the byte with allocation bit
000058F9 C3                  <1>  retn
                             <1>        ; rts r0
                                 %include 'u6.s'        ; 31/05/2015
                             <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS6.INC
                             <1> ; Last Modification: 18/11/2015
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
                             <1> ; (v0.1 - Beginning: 11/07/2012)
                             <1> ;
                             <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                             <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                             <1> ; <Bell Laboratories (17/3/1972)>
                             <1> ; <Preliminary Release of UNIX Implementation Document>
                             <1> ;
                             <1> ; Retro UNIX 8086 v1 - U6.ASM (23/07/2014) //// UNIX v1 -> u6.s
                             <1> ;
                             <1> ; ****************************************************************************
                             <1>
                             <1> readi:
                             <1>  ; 20/05/2015
                             <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 11/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Reads from an inode whose number in R1
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - inode number
                             <1>  ;    u.count - byte count user desires
                             <1>  ;    u.base - points to user buffer
                             <1>  ;    u.fofp - points to word with current file offset
                             <1>  ; OUTPUTS ->
                             <1>  ;    u.count - cleared
                             <1>  ;    u.nread - accumulates total bytes passed back
                             <1>  ;
                             <1>  ; ((AX = R1)) input/output
                             <1>  ;    (Retro UNIX Prototype : 01/03/2013 - 14/12/2012, UNIXCOPY.ASM)
                             <1>         ;    ((Modified registers: edx, ebx, ecx, esi, esi, ebp))
                             <1>
000058FA 31D2                <1>  xor   edx, edx ; 0
000058FC 8915[A8740000]      <1>  mov   [u.nread], edx ; 0
                             <1>         ; clr u.nread / accumulates number of bytes transmitted
00005902 668915[DF740000]    <1>  mov   [u.pcount], dx ; 19/05/2015
00005909 3915[A4740000]      <1>  cmp   [u.count], edx ; 0
                             <1>           ; tst u.count / is number of bytes to be read greater than 0
0000590F 7701                <1>  ja    short readi_1 ; 1f
                             <1>         ; bgt 1f / yes, branch
00005911 C3                  <1>  retn
                             <1>         ; rts r0 / no, nothing to read; return to caller
                             <1> readi_1: ; 1:
                             <1>           ; mov r1,-(sp) / save i-number on stack
00005912 6683F828            <1>  cmp   ax, 40
                             <1>         ; cmp r1,$40. / want to read a special file
                             <1>         ;             / (i-nodes 1,...,40 are for special files)
00005916 0F87D3000000        <1>         ja     dskr
                             <1>         ; ble 1f / yes, branch
                             <1>         ; jmp dskr / no, jmp to dskr;
                             <1>         ;         / read file with i-node number (r1)
                             <1>         ;    / starting at byte ((u.fofp)), read in u.count bytes
                             <1>  ; (20/05/2015)
0000591C 50                  <1>  push  eax ; because subroutines will jump to 'ret_'
                             <1>  ; 1:
0000591D 0FB6D8              <1>  movzx ebx, al
00005920 66C1E302            <1>  shl   bx, 2
                             <1>         ; asl r1 / multiply inode number by 2
00005924 81C3[28590000]      <1>  add   ebx, readi_2 - 4
0000592A FF23                <1>  jmp   dword [ebx]
                             <1>         ; jmp *1f-2(r1)
                             <1> readi_2: ; 1:
0000592C [78590000]          <1>  dd    rtty ; tty, AX = 1 (runix)
                             <1>         ;rtty / tty; r1=2
                             <1>         ;rppt / ppt; r1=4
00005930 [CB590000]          <1>  dd    rmem ; mem, AX = 2 (runix)
                             <1>         ;rmem / mem; r1=6
                             <1>         ;rrf0 / rf0
                             <1>         ;rrk0 / rk0
                             <1>         ;rtap / tap0
                             <1>         ;rtap / tap1
                             <1>         ;rtap / tap2
                             <1>         ;rtap / tap3
                             <1>         ;rtap / tap4
                             <1>         ;rtap / tap5
                             <1>         ;rtap / tap6
                             <1>         ;rtap / tap7
00005934 [8D600000]          <1>  dd    rfd ; fd0, AX = 3 (runix only)
00005938 [8D600000]          <1>  dd    rfd ; fd1, AX = 4 (runix only)
0000593C [8D600000]          <1>  dd    rhd ; hd0, AX = 5 (runix only)
00005940 [8D600000]          <1>  dd    rhd ; hd1, AX = 6 (runix only)
00005944 [8D600000]          <1>  dd    rhd ; hd2, AX = 7 (runix only)
00005948 [8D600000]          <1>  dd    rhd ; hd3, AX = 8 (runix only)
0000594C [E0590000]          <1>  dd    rlpr ; lpr, AX = 9 (invalid, write only device !?)
00005950 [C7590000]          <1>  dd    rcvt ; tty0, AX = 10 (runix)
                             <1>         ;rcvt / tty0
00005954 [C7590000]          <1>  dd    rcvt ; tty1, AX = 11 (runix)
                             <1>         ;rcvt / tty1
00005958 [C7590000]          <1>  dd    rcvt ; tty2, AX = 12 (runix)
                             <1>         ;rcvt / tty2
0000595C [C7590000]          <1>  dd    rcvt ; tty3, AX = 13 (runix)
                             <1>         ;rcvt / tty3
00005960 [C7590000]          <1>  dd    rcvt ; tty4, AX = 14 (runix)
                             <1>         ;rcvt / tty4
00005964 [C7590000]          <1>  dd    rcvt ; tty5, AX = 15 (runix)
                             <1>         ;rcvt / tty5
00005968 [C7590000]          <1>  dd    rcvt ; tty6, AX = 16 (runix)
                             <1>         ;rcvt / tty6
0000596C [C7590000]          <1>  dd    rcvt ; tty7, AX = 17 (runix)
                             <1>         ;rcvt / tty7
00005970 [C7590000]          <1>  dd    rcvt ; COM1, AX = 18 (runix only)
                             <1>         ;rcrd / crd
00005974 [C7590000]          <1>  dd    rcvt ; COM2, AX = 19 (runix only)
                             <1>
                             <1> rtty: ; / read from console tty
                             <1>  ; 17/10/2015 - 16/07/2015 (Retro UNIX 8086 v1)
                             <1>  ;          (Only 1 byte is read, by ignoring byte count!)
                             <1>  ;          WHAT FOR: Every character from Keyboard input
                             <1>  ;          must be written immediate on video page (screen)
                             <1>  ;          when it is required.
                             <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 11/03/2013 - 19/06/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Console tty buffer is PC keyboard buffer
                             <1>  ; and keyboard-keystroke handling is different than original
                             <1>  ; unix (PDP-11) here. TTY/Keyboard procedures here are changed
                             <1>  ; according to IBM PC compatible ROM BIOS keyboard functions.
                             <1>  ;
                             <1>  ; 06/12/2013
00005978 0FB61D[C9740000]    <1>  movzx ebx, byte [u.uno] ; process number
0000597F 8A83[D5710000]      <1>  mov   al, [ebx+p.ttyc-1] ; current/console tty
                             <1> rttys:
                             <1>        ; mov tty+[8*ntty]-8+6,r5 / r5 is the address of the 4th word of
                             <1>                 ; / of the control and status block
                             <1>        ; tst 2(r5) / for the console tty; this word points to the console
                             <1>               ; / tty buffer
                             <1>  ; 28/07/2013
00005985 A2[CE740000]        <1>  mov   [u.ttyn], al
                             <1>  ; 13/01/2014
0000598A FEC0                <1>  inc   al
0000598C A2[B0740000]        <1>  mov   [u.ttyp], al ; tty number + 1
                             <1> rtty_nc: ; 01/02/2014
                             <1>  ; 29/09/2013
00005991 B90A000000          <1>  mov   ecx, 10
                             <1> rtty_1:      ; 01/02/2014
00005996 6651                <1>  push  cx ; 29/09/2013
                             <1>  ; byte [u.ttyn] = tty number (0 to 9)
00005998 B001                <1>  mov   al, 1
0000599A E8330B0000          <1>  call  getc
0000599F 6659                <1>  pop   cx ; 29/09/2013
000059A1 7516                <1>  jnz   short rtty_2
                             <1>        ; bne 1f / 2nd word of console tty buffer contains number
                             <1>                 ; / of chars. Is this number non-zero?
000059A3 E20D                <1>  loop  rtty_idle ; 01/02/2014
                             <1>  ; 05/10/2013
000059A5 8A25[CE740000]      <1>  mov   ah, [u.ttyn]
                             <1>  ; 29/09/2013
000059AB E871FBFFFF          <1>  call  sleep
                             <1>        ; jsr r0,canon; ttych / if 0, call 'canon' to get a line
                             <1>                 ;           / (120 chars.)
                             <1>  ;byte [u.ttyn] = tty number (0 to 9)
000059B0 EBDF                <1>  jmp   short rtty_nc ; 01/02/2014
                             <1>
                             <1> rtty_idle:
                             <1>  ; 29/07/2013
000059B2 E8D6FAFFFF          <1>  call  idle
000059B7 EBDD                <1>  jmp   short rtty_1 ; 01/02/2014
                             <1>  ;1:
                             <1>        ; tst 2(r5) / is the number of characters zero
                             <1>        ; beq ret1 / yes, return to caller via 'ret1'
                             <1>        ; movb *4(r5),r1 / no, put character in r1
                             <1>        ; inc 4(r5) / 3rd word of console tty buffer points to byte which
                             <1>                  ; / contains the next char.
                             <1>        ; dec 2(r5) / decrement the character count
                             <1> rtty_2:
000059B9 30C0                <1>  xor   al, al
000059BB E8120B0000          <1>  call  getc
000059C0 E892000000          <1>  call  passc
                             <1>        ; jsr r0,passc / move the character to core (user)
                             <1>  ;; 17/10/2015 - 16/07/2015
                             <1>  ; 19/06/2014
                             <1>  ;;jnz short rtty_nc
000059C5 58                  <1>  pop   eax  ; (20/05/2015)
000059C6 C3                  <1>  retn
                             <1> ;ret1:
                             <1>        ; jmp ret / return to caller via 'ret'
                             <1>
                             <1> rcvt:   ; < receive/read character from tty >
                             <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 15/05/2013 - 06/12/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification !
                             <1>  ;
                             <1>  ; In original UNIX v1, 'rcvt' routine
                             <1>  ;           (exactly different than this one)
                             <1>  ;     was in 'u9.s' file.
                             <1>  ;
000059C7 2C0A                <1>  sub   al, 10
                             <1>  ; AL = tty number (0 to 9), (COM1=8, COM2=9)
                             <1>  ; 16/07/2013
                             <1>  ; 21/05/2013
000059C9 EBBA                <1>         jmp     short rttys
                             <1>
                             <1> ;rppt: / read paper tape
                             <1> ;jsr   r0,pptic / gets next character in clist for ppt input and
                             <1> ;             / places
                             <1> ;      br ret / it in r1; if there 1s no problem with reader, it
                             <1> ;             / also enables read bit in prs
                             <1> ;jsr   r0,passc / place character in users buffer area
                             <1> ;br    rppt
                             <1>
                             <1> rmem: ; / transfer characters from memory to a user area of core
                             <1>  ; 17/10/2015
                             <1>  ; 11/06/2015
                             <1>  ; 24/05/2015
                             <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ;
000059CB 8B35[90740000]      <1>  mov     esi, [u.fofp]
                             <1> rmem_1:
000059D1 8B1E                <1>         mov     ebx, [esi]
                             <1>          ; mov *u.fofp,r1 / save file offset which points to the char
                             <1>                       ; / to be transferred to user
000059D3 FF06                <1>         inc     dword [esi] ; 17/10/2015
                             <1>        ; inc *u.fofp / increment file offset to point to 'next'
                             <1>                  ; / char in memory file
000059D5 8A03                <1>  mov   al, [ebx]
                             <1>        ; movb (r1),r1 / get character from memory file,
                             <1>                       ; / put it in r1
000059D7 E87B000000          <1>  call  passc        ; jsr r0,passc / move this character to
                             <1>                    ;  / the next byte of the users core area
                             <1>        ; br rmem / continue
000059DC 75F3                <1>  jnz   short rmem_1
                             <1> ret_:
000059DE 58                  <1>  pop   eax ; 09/06/2015
000059DF C3                  <1>  retn
                             <1>
                             <1> rlpr:
                             <1> ;1:
                             <1> ;rcrd:
000059E0 C705[CF740000]0F00- <1>         mov     dword [u.error], ERR_DEV_NOT_RDY ; 19/05/2015
000059E8 0000                <1>
000059EA E98DE6FFFF          <1>  jmp   error
                             <1>        ;jmp  error / see 'error' routine
                             <1>
                             <1> dskr:
                             <1>  ; 12/10/2015
                             <1>  ; 21/08/2015
                             <1>  ; 25/07/2015
                             <1>  ; 10/07/2015
                             <1>  ; 16/06/2015
                             <1>  ; 31/05/2015
                             <1>  ; 24/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 26/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)
                             <1> dskr_0:
000059EF 50                  <1>  push  eax
```

```
                             <1>        ; mov (sp),r1 / i-number in r1
                             <1>  ; AX = i-number
000059F0 E825FDFFFF          <1>  call  iget
                             <1>        ; jsr r0,iget / get i-node (r1) into i-node section of core
000059F5 0FB715[5A710000]    <1>         movzx   edx, word [i.size] ; 16/06/2015
                             <1>        ; mov i.size,r2 / file size in bytes in r2
000059FC 8B1D[90740000]      <1>  mov   ebx, [u.fofp]
00005A02 2B13                <1>  sub   edx, [ebx]
                             <1>        ; sub *u.fofp,r2 / subtract file offset
                             <1>         ; 12/10/2015
                             <1>  ; jna     short ret_
                             <1>        ; blos ret
00005A04 7709                <1>  ja    short dskr_1
                             <1>  ;
                             <1> dskr_retn: ; 12/10/2015
00005A06 58                  <1>  pop   eax
00005A07 C605[E1740000]00    <1>  mov   byte [u.kcall], 0
00005A0E C3                  <1>  retn
                             <1> dskr_1:
00005A0F 3B15[A4740000]      <1>  cmp     edx, [u.count]
                             <1>        ; cmp r2,u.count / are enough bytes left in file
                             <1>                      ; / to carry out read
00005A15 7306                <1>  jnb   short dskr_2
                             <1>        ; bhis 1f
00005A17 8915[A4740000]      <1>  mov   [u.count], edx
                             <1>        ; mov r2,u.count / no, just read to end of file
                             <1> dskr_2: ; 1:
                             <1>  ; AX = i-number
00005A1D E885FBFFFF          <1>  call  mget
                             <1>        ; jsr r0,mget / returns physical block number of block
                             <1>                   ; / in file where offset points
                             <1>  ; eAX = physical block number
00005A22 E8C0070000          <1>  call  dskrd
                             <1>        ; jsr r0,dskrd / read in block, r5 points to
                             <1>                     ; / 1st word of data in buffer
                             <1>  ; 09/06/2015
00005A27 803D[E1740000]00    <1>  cmp   byte [u.kcall], 0 ; the caller is 'namei' sign (=1)
00005A2E 770F                <1>  ja    short dskr_4       ; zf=0 -> the caller is 'namei'
00005A30 66833D[DF740000]00  <1>  cmp   word [u.pcount], 0
00005A38 7705                <1>  ja    short dskr_4
                             <1> dskr_3:
                             <1>  ; [u.base] = virtual address to transfer (as destination address)
00005A3A E853000000          <1>  call  trans_addr_w ; translate virtual address to physical (w)
                             <1> dskr_4:
                             <1>  ; eBX (r5) = system (I/O) buffer address -physical-
00005A3F E8C7020000          <1>  call  sioreg
                             <1>        ; jsr r0,sioreg
00005A44 87F7                <1>  xchg  esi, edi
                             <1>  ; eDI = file (user data) offset
                             <1>  ; eSI = sector (I/O) buffer offset
                             <1>  ; eCX = byte count
00005A46 F3A4                <1>  rep   movsb
                             <1>        ; movb (r2)+,(r1)+ / move data from buffer into working core
                             <1>                           ; / starting at u.base
                             <1>        ; dec r3
                                        ; bne 2b / branch until proper number of bytes are transferred
                             <1>  ; 25/07/2015
                             <1>  ; eax = remain bytes in buffer
                             <1>         ;       (check if remain bytes in the buffer > [u.pcount])
00005A48 09C0                <1>  or    eax, eax
00005A4A 75EE                <1>  jnz   short dskr_3 ; (page end before system buffer end!)
                             <1>  ; 03/08/2013
                             <1>  ;pop  eax
00005A4C 390D[A4740000]      <1>  cmp   [u.count], ecx ; 0
                             <1>        ; tst u.count / all bytes read off disk
                             <1>        ; bne dskr
                             <1>        ; br ret
                             <1>         ;ja      short dskr_0
                             <1>  ;mov  [u.kcall], cl ; 0 ; 09/06/2015
                             <1>  ;retn
                             <1>  ; 12/10/2015
00005A52 76B2                <1>  jna   short dskr_retn
00005A54 58                  <1>  pop   eax  ; (i-node number)
00005A55 EB98                <1>  jmp   short dskr_0
                             <1>
                             <1> passc:
                             <1>  ; 18/10/2015
                             <1>  ; 10/07/2015
                             <1>  ; 01/07/2015
                             <1>  ; 08/06/2015
                             <1>  ; 04/06/2015
                             <1>  ; 20/05/2015
                             <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ;
                             <1>        ;(Retro UNIX 386 v1 - translation from user's virtual address
                             <1>  ;                     to physical address
00005A57 66833D[DF740000]00  <1>  cmp   word [u.pcount], 0 ; byte count in page = 0 (initial value)
                             <1>                     ; 1-4095 --> use previous physical base address
                             <1>                     ; in [u.pbase]
00005A5F 7705                <1>  ja    short passc_3
                             <1>  ; 08/06/2015 - 10/07/2015
00005A61 E82C000000          <1>  call  trans_addr_w
```

```
                              <1> passc_3:
                              <1>  ; 19/05/2015
00005A66 66FF0D[DF740000]     <1>  dec   word [u.pcount]
                              <1>  ;
00005A6D 8B1D[DB740000]       <1>  mov   ebx, [u.pbase]
00005A73 8803                 <1>  mov   [ebx], al
                              <1>        ; movb r1,*u.base / move a character to the next byte of the
                              <1>                       ; / users buffer
00005A75 FF05[A0740000]       <1>  inc   dword [u.base]
                              <1>        ; inc u.base / increment the pointer to point to
                              <1>                ; / the next byte in users buffer
00005A7B FF05[DB740000]       <1>  inc   dword [u.pbase] ; 04/06/2015
00005A81 FF05[A8740000]       <1>  inc   dword [u.nread]
                              <1>        ; inc u.nread / increment the number of bytes read
00005A87 FF0D[A4740000]       <1>  dec   dword [u.count]
                              <1>        ; dec u.count / decrement the number of bytes to be read
                              <1>        ; bne 1f / any more bytes to read?; yes, branch
00005A8D C3                   <1>  retn
                              <1>        ; mov (sp)+,r0 / no, do a non-local return to the caller of
                              <1>                     ; / 'readi' by:
                              <1>        ;/ (1) pop the return address off the stack into r0
                              <1>        ; mov (sp)+,r1 / (2) pop the i-number off the stack into r1
                              <1>  ;1:
                              <1>        ; clr *$ps / clear processor status
                              <1>        ; rts r0 / return to address currently on top of stack
                              <1>
                              <1> trans_addr_r:
                              <1>  ; Translate virtual address to physical address
                              <1>  ; for reading from user's memory space
                              <1>  ; (Retro UNIX 386 v1 feature only !)
                              <1>  ; 18/10/2015
                              <1>  ; 10/07/2015
                              <1>  ; 09/06/2015
                              <1>  ; 08/06/2015
                              <1>  ; 04/06/2015
                              <1>  ;
                              <1>  ; 18/10/2015
00005A8E 31D2                 <1>  xor   edx, edx ; 0 (read access sign)
00005A90 EB04                 <1>  jmp   short trans_addr_rw
                              <1>
                              <1>  ;push eax
                              <1>  ;push ebx
                              <1>  ;mov  ebx, [u.base]
                              <1>  ;call get_physical_addr ; get physical address
                              <1>  ;;jnc short cpass_0
                              <1>  ;jnc  short passc_1
                              <1>  ;mov  [u.error], eax
                              <1>  ;;pop ebx
                              <1>  ;;pop eax
                              <1>  ;jmp  error
                              <1> ;cpass_0:
                              <1>  ; 18/10/2015
                              <1>  ; 20/05/2015
                              <1>  ;mov  [u.pbase], eax ; physical address
                              <1>  ;mov  [u.pcount], cx ; remain byte count in page (1-4096)
                              <1>  ;pop  ebx
                              <1>  ;pop  eax
                              <1>  ;retn ; 08/06/2015
                              <1>
                              <1> trans_addr_w:
                              <1>  ; Translate virtual address to physical address
                              <1>  ; for writing to user's memory space
                              <1>  ; (Retro UNIX 386 v1 feature only !)
                              <1>  ; 18/10/2015
                              <1>  ; 29/07/2015
                              <1>  ; 10/07/2015
                              <1>  ; 09/06/2015
                              <1>  ; 08/06/2015
                              <1>  ; 04/06/2015 (passc)
                              <1>  ;
                              <1>  ; 18/10/2015
00005A92 29D2                 <1>  sub   edx, edx
00005A94 FEC2                 <1>  inc   dl ; 1 (write access sign)
                              <1> trans_addr_rw:
00005A96 50                   <1>  push  eax
00005A97 53                   <1>  push  ebx
                              <1>  ; 18/10/2015
00005A98 52                   <1>  push  edx ; r/w sign (in DL)
                              <1>  ;
00005A99 8B1D[A0740000]       <1>  mov   ebx, [u.base]
00005A9F E826DCFFFF           <1>  call  get_physical_addr ; get physical address
00005AA4 730A                 <1>  jnc   short passc_0
00005AA6 A3[CF740000]         <1>  mov   [u.error], eax
                              <1>  ;pop  edx
                              <1>  ;pop  ebx
                              <1>  ;pop  eax
00005AAB E9CCE5FFFF           <1>  jmp   error
                              <1> passc_0:
00005AB0 F6C202               <1>  test  dl, PTE_A_WRITE ; writable page ; 18/10/2015
00005AB3 5A                   <1>  pop   edx ; 18/10/2015
00005AB4 751C                 <1>  jnz   short passc_1
                              <1>  ; 18/10/2015
```

```
00005AB6 20D2                <1>  and   dl, dl
00005AB8 7418                <1>  jz    short passc_1
                             <1>  ; 20/05/2015
                             <1>  ; read only (duplicated) page -must be copied to a new page-
                             <1>  ; EBX = linear address
00005ABA 51                  <1>  push  ecx
00005ABB E81FD9FFFF          <1>  call  copy_page
00005AC0 59                  <1>  pop   ecx
00005AC1 721E                <1>  jc    short passc_2
00005AC3 50                  <1>  push  eax ; physical address of the new/allocated page
00005AC4 E82BDBFFFF          <1>  call  add_to_swap_queue
00005AC9 58                  <1>  pop   eax
                             <1>  ; 18/10/2015
00005ACA 81E3FF0F0000        <1>  and   ebx, PAGE_OFF ; 0FFFh
                             <1>  ;mov  ecx, PAGE_SIZE
                             <1>  ;sub  ecx, ebx
00005AD0 01D8                <1>  add   eax, ebx
                             <1> passc_1:
                             <1>  ; 18/10/2015
                             <1>  ; 20/05/2015
00005AD2 A3[DB740000]        <1>  mov   [u.pbase], eax ; physical address
00005AD7 66890D[DF740000]    <1>  mov   [u.pcount], cx ; remain byte count in page (1-4096)
00005ADE 5B                  <1>  pop   ebx
00005ADF 58                  <1>  pop   eax
00005AE0 C3                  <1>  retn  ; 08/06/2015
                             <1> passc_2:
00005AE1 C705[CF740000]0100- <1>  mov   dword [u.error], ERR_MINOR_IM ; "Insufficient memory !" error
00005AE9 0000                <1>
                             <1>  ;pop  ebx
                             <1>  ;pop  eax
00005AEB E98CE5FFFF          <1>  jmp   error
                             <1>
                             <1> writei:
                             <1>  ; 20/05/2015
                             <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 12/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Write data to file with inode number in R1
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - inode number
                             <1>  ;    u.count - byte count to be written
                             <1>  ;    u.base - points to user buffer
                             <1>  ;    u.fofp - points to word with current file offset
                             <1>  ; OUTPUTS ->
                             <1>  ;    u.count - cleared
                             <1>  ;    u.nread - accumulates total bytes passed back
                             <1>  ; ((AX = R1))
                             <1>  ;     (Retro UNIX Prototype : 18/11/2012 - 11/11/2012, UNIXCOPY.ASM)
                             <1>  ;     ((Modified registers: DX, BX, CX, SI, DI, BP))
                             <1>
00005AF0 31C9                <1>  xor   ecx, ecx
00005AF2 890D[A8740000]      <1>  mov   [u.nread], ecx  ; 0
                             <1>        ; clr u.nread / clear the number of bytes transmitted during
                             <1>                   ; / read or write calls
00005AF8 66890D[DF740000]    <1>  mov   [u.pcount], cx ; 19/05/2015
00005AFF 390D[A4740000]      <1>  cmp   [u.count], ecx
                             <1>  ;     ; tst u.count / test the byte count specified by the user
00005B05 7701                <1>  ja    short writei_1 ; 1f
                             <1>        ; bgt 1f / any bytes to output; yes, branch
00005B07 C3                  <1>  retn
                             <1>  ;     ; rts r0 / no, return - no writing to do
                             <1> writei_1: ;1:
                             <1>        ; mov r1 ,-(sp) / save the i-node number on the stack
00005B08 6683F828            <1>  cmp   ax, 40
                             <1>        ; cmp r1,$40.
                             <1>        ; / does the i-node number indicate a special file?
00005B0C 0F87F2000000        <1>         ja      dskw
                             <1>        ; bgt dskw / no, branch to standard file output
                             <1>  ; (20/05/2015)
00005B12 50                  <1>  push  eax ; because subroutines will jump to 'ret_'
00005B13 0FB6D8              <1>  movzx ebx, al
00005B16 66C1E302            <1>  shl   bx, 2
                             <1>        ; asl r1 / yes, calculate the index into the special file
00005B1A 81C3[1E5B0000]      <1>  add   ebx, writei_2 - 4
00005B20 FF23                <1>  jmp   dword [ebx]
                             <1>        ; jmp *1f-2(r1)
                             <1>        ; / jump table and jump to the appropriate routine
                             <1> writei_2: ;1:
00005B22 [6E5B0000]          <1>  dd    wtty ; tty, AX = 1 (runix)
                             <1>         ;wtty / tty; r1=2
                             <1>         ;wppt / ppt; r1=4
00005B26 [D45B0000]          <1>  dd    wmem ; mem, AX = 2 (runix)
                             <1>         ;wmem / mem; r1=6
                             <1>         ;wrf0 / rf0
                             <1>         ;wrk0 / rk0
                             <1>         ;wtap / tap0
                             <1>         ;wtap / tap1
                             <1>         ;wtap / tap2
                             <1>         ;wtap / tap3
                             <1>         ;wtap / tap4
                             <1>         ;wtap / tap5
                             <1>        ;wtap / tap6
                             <1>        ;wtap / tap7
00005B2A [0F610000]          <1>  dd    wfd ; fd0, AX = 3 (runix only)
00005B2E [0F610000]          <1>  dd    wfd ; fd1, AX = 4 (runix only)
00005B32 [0F610000]          <1>  dd    whd ; hd0, AX = 5 (runix only)
00005B36 [0F610000]          <1>  dd    whd ; hd1, AX = 6 (runix only)
00005B3A [0F610000]          <1>  dd    whd ; hd2, AX = 7 (runix only)
00005B3E [0F610000]          <1>  dd    whd ; hd3, AX = 8 (runix only)
00005B42 [C55B0000]          <1>  dd    wlpr ; lpr, AX = 9   (runix)
00005B46 [BF5B0000]          <1>  dd    xmtt ; tty0, AX = 10 (runix)
                             <1>        ;xmtt / tty0
00005B4A [BF5B0000]          <1>  dd    xmtt ; tty1, AX = 11 (runix)
                             <1>        ;xmtt / tty1
00005B4E [BF5B0000]          <1>  dd    xmtt ; tty2, AX = 12 (runix)
                             <1>        ;xmtt / tty2
00005B52 [BF5B0000]          <1>  dd    xmtt ; tty3, AX = 13 (runix)
                             <1>        ;xmtt / tty3
00005B56 [BF5B0000]          <1>  dd    xmtt ; tty4, AX = 14 (runix)
                             <1>        ;xmtt / tty4
00005B5A [BF5B0000]          <1>  dd    xmtt ; tty5, AX = 15 (runix)
                             <1>        ;xmtt / tty5
00005B5E [BF5B0000]          <1>  dd    xmtt ; tty6, AX = 16 (runix)
                             <1>        ;xmtt / tty6
00005B62 [BF5B0000]          <1>  dd    xmtt ; tty7, AX = 17 (runix)
                             <1>        ;xmtt / tty7
00005B66 [BF5B0000]          <1>  dd    xmtt ; COM1, AX = 18 (runix only)
                             <1>        ; / wlpr / lpr
00005B6A [BF5B0000]          <1>  dd    xmtt ; COM2, AX = 19 (runix only)
                             <1>
                             <1> wtty: ; write to console tty (write to screen)
                             <1>  ; 18/11/2015
                             <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 12/03/2013 - 07/07/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Console tty output is on current video page
                             <1>  ; Console tty character output procedure is changed here
                             <1>  ; according to IBM PC compatible ROM BIOS video (text mode) functions.
                             <1>  ;
00005B6E 0FB61D[C9740000]    <1>  movzx ebx, byte [u.uno] ; process number
00005B75 8AA3[D5710000]      <1>  mov   ah, [ebx+p.ttyc-1] ; current/console tty
00005B7B 88E0                <1>  mov   al, ah ; 07/07/2014
                             <1> wttys:
                             <1>  ; 10/10/2013
00005B7D 8825[CE740000]      <1>  mov   [u.ttyn], ah
                             <1>  ; 13/01/2014
00005B83 FEC0                <1>  inc   al
00005B85 A2[B1740000]        <1>  mov   [u.ttyp+1], al ; tty number + 1
                             <1> wtty_nc: ; 15/05/2013
                             <1>  ; AH = [u.ttyn] = tty number ; 28/07/2013
00005B8A E81C010000          <1>  call  cpass
                             <1>        ; jsr r0,cpass / get next character from user buffer area; if
                             <1>                    ; / none go to return address in syswrite
                             <1>        ; tst r1 / is character = null
                             <1>        ; beq wtty / yes, get next character
                             <1>  ; 10/10/2013
00005B8F 742C                <1>  jz    short wret
                             <1>  ;1 :
                             <1>        ;mov  $240,*$ps / no, set processor priority to five
                             <1>        ;cmpb cc+1,$20. / is character count for console tty greater
                             <1>        ;               / than 20
                             <1>        ;bhis 2f / yes; branch to put process to sleep
                             <1>  ; 27/06/2014
                             <1> wtty_1:
                             <1>  ; AH = tty number
                             <1>  ; AL = ASCII code of the character
                             <1>  ; 15/04/2014
00005B91 6650                <1>  push  ax
00005B93 E8AF0A0000          <1>  call  putc ; 14/05/2013
00005B98 731F                <1>  jnc   short wtty_2
                             <1>  ; 18/11/2015
00005B9A E8EEF8FFFF          <1>  call  idle
00005B9F 668B0424            <1>  mov   ax, [esp]
00005BA3 E89F0A0000          <1>  call  putc
00005BA8 730F                <1>  jnc   short wtty_2
                             <1>  ; 02/06/2014
00005BAA 8A25[CE740000]      <1>  mov   ah, [u.ttyn]
00005BB0 E86CF9FFFF          <1>  call  sleep
00005BB5 6658                <1>  pop   ax
00005BB7 EBD8                <1>  jmp   short wtty_1
                             <1>        ; jc  error ; 15/05/2013 (COM1 or COM2 serial port error)
                             <1>        ; jsr       r0,putc; 1 / find place in freelist to assign to
                             <1>                   ; / console tty and
                             <1>        ; br  2f / place character in list; if none available
                             <1>               ; / branch to put process to sleep
                             <1>        ; jsr r0,startty / attempt to output character on tty
                             <1> wtty_2:
                             <1>  ; 15/04/2014
00005BB9 6658                <1>  pop   ax
00005BBB EBCD                <1>  jmp   short wtty_nc
                             <1>        ; br wtty
                             <1> wret:  ; 10/10/2013 (20/05/2015)
00005BBD 58                  <1>  pop   eax
```

```
00005BBE C3                  <1>  retn
                             <1>  ;2:
                             <1>        ;mov  r1,-(sp) / place character on stack
                             <1>        ;jsr  r0,sleep; 1 / put process to sleep
                             <1>        ;mov  (sp)+,r1 / remove character from stack
                             <1>        ;br   1b / try again to place character in clist and output
                             <1>
                             <1> xmtt:   ; < send/write character to tty >
                             <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 15/05/2013 - 06/12/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification !
                             <1>  ;
                             <1>  ; In original UNIX v1, 'xmtt' routine
                             <1>  ;           (exactly different than this one)
                             <1>  ;     was in 'u9.s' file.
                             <1>  ;
00005BBF 2C0A                <1>  sub   al, 10
                             <1>  ; AL = tty number (0 to 9), (COM1=8, COM2=9)
                             <1>   ; 10/10/2013
00005BC1 88C4                <1>  mov   ah, al
                             <1>  ; 28/07/2013
00005BC3 EBB8                <1>  jmp   short wttys
                             <1>
                             <1> ;wppt:
                             <1> ;jsr   r0,cpass / get next character from user buffer area,
                             <1> ;               / if none return to writei's calling routine
                             <1> ;jsr   r0,pptoc / output character on ppt
                             <1> ;br    wppt
                             <1> wlpr:
00005BC5 C705[CF740000]0F00- <1>         mov     dword [u.error], ERR_DEV_NOT_RDY ; 19/05/2015
00005BCD 0000                <1>
00005BCF E9A8E4FFFF          <1>  jmp   error   ; ... Printing procedure will be located here ...
                             <1>        ;/    jsr   r0,cpass
                             <1>        ;/    cmp   r0,$'a
                             <1>        ;/    blo   1f
                             <1>        ;/    cmp   r1,$'z
                             <1>        ;/    bhi   1f
                             <1>        ;/    sub   $40,r1
                             <1>        ;/1:
                             <1>        ;/    jsr   r0,lptoc
                             <1>        ;/    br    wlpr
                             <1>        ; br rmem / continue
                             <1>
                             <1> wmem: ; / transfer characters from a user area of core to memory file
                             <1>  ; 17/10/2015
                             <1>  ; 11/06/2015
                             <1>  ; 24/05/2015
                             <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ;
00005BD4 813D[1B080000]-     <1>  cmp   dword [x_timer], clock ; multi tasking clock/timer
00005BDA [B1540000]          <1>
00005BDE 7415                <1>         je      short wmem_acc_err
                             <1>  ;
00005BE0 8B35[90740000]      <1>         mov     esi, [u.fofp]
                             <1> wmem_1:
00005BE6 E8C0000000          <1>  call  cpass
                             <1>        ; jsr r0,cpass / get next character from users area of
                             <1>                  ; / core and put it in r1
                             <1>             ; mov r1,-(sp) / put character on the stack
                             <1>  ; 20/09/2013
00005BEB 74D0                <1>  jz    short wret ; wmem_2
00005BED 8B1E                <1>         mov     ebx, [esi]
                             <1>        ; mov *u.fofp,r1 / save file offset in r1
00005BEF FF06                <1>         inc     dword [esi] ; 17/10/2015
                             <1>        ; inc *u.fofp / increment file offset to point to next
                             <1>                  ; / available location in file
00005BF1 8803                <1>  mov   [ebx], al
                             <1>        ; movb (sp)+,(r1) / pop char off stack, put in memory loc
                             <1>                      ; / assigned to it
00005BF3 EBF1                <1>  jmp   short wmem_1
                             <1>        ; br wmem / continue
                             <1>  ;1:
                             <1>  ;jmp  error / ?
                             <1> ;wmem_2:
                             <1> ;; 20/09/2013
                             <1> ;pop   ax
                             <1> ;retn
                             <1>
                             <1> wmem_acc_err:
00005BF5 C705[CF740000]0B00- <1>  mov   dword [u.error], ERR_FILE_ACCESS ; permission denied !
00005BFD 0000                <1>
00005BFF E978E4FFFF          <1>  jmp   error
                             <1>
                             <1>
                             <1> dskw: ; / write routine for non-special files
                             <1>  ;
                             <1>  ; 25/07/2015
                             <1>  ; 16/06/2015
                             <1>  ; 09/06/2015
                             <1>  ; 31/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 26/04/2013 - 20/09/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 01/08/2013 (mkdir_w check)
00005C04 6650                <1>  push  ax ; 26/04/2013
                             <1>        ; mov (sp),r1 / get an i-node number from the stack into r1
                             <1>  ; AX = inode number
00005C06 E80FFBFFFF          <1>  call  iget
                             <1>        ; jsr r0,iget / write i-node out (if modified),
                             <1>                     ; / read i-node 'r1' into i-node area of core
00005C0B 8B1D[90740000]      <1>         mov     ebx, [u.fofp]
00005C11 8B13                <1>  mov   edx, [ebx]
                             <1>        ; mov *u.fofp,r2 / put the file offset [(u.off) or the offset
                             <1>                      ; / in the fsp entry for this file] in r2
00005C13 0315[A4740000]      <1>  add   edx, [u.count]
                             <1>        ; add u.count,r2 / no. of bytes to be written
                             <1>                      ; / + file offset is put in r2
                             <1>  ; 16/06/2015
00005C19 81FAFFFF0000        <1>  cmp   edx, 65535 ; file size limit (for UNIX v1 file system)
00005C1F 760F                <1>  jna   short dskw_0
00005C21 C705[CF740000]1400- <1>  mov   dword [u.error], ERR_FILE_SIZE ; 'file size error !'
00005C29 0000                <1>
00005C2B E94CE4FFFF          <1>  jmp   error
                             <1> dskw_0:
00005C30 663B15[5A710000]    <1>  cmp     dx, [i.size]
                             <1>        ; cmp r2,i.size / is this greater than the present size of
                             <1>                        ; / the file?
00005C37 760C                <1>  jna   short dskw_1
                             <1>        ; blos 1f / no, branch
00005C39 668915[5A710000]    <1>         mov     [i.size], dx
                             <1>        ; mov r2,i.size / yes, increase the file size to
                             <1>                    ; / file offset + no. of data bytes
00005C40 E8E8FBFFFF          <1>  call  setimod
                             <1>        ; jsr r0,setimod / set imod=1 (i.e., core inode has been
                             <1>                   ; / modified), stuff time of modification into
                             <1>                 ; / core image of i-node
                             <1> dskw_1: ; 1:
00005C45 E85DF9FFFF          <1>  call  mget
                             <1>  ; eAX = Block number
                             <1>        ; jsr r0,mget / get the block no. in which to write
                             <1>                   ; /     the next data byte
                             <1>  ; eax = block number
00005C4A 8B1D[90740000]      <1>  mov     ebx, [u.fofp]
00005C50 8B13                <1>  mov   edx, [ebx]
00005C52 81E2FF010000        <1>  and   edx, 1FFh
                             <1>        ; bit *u.fofp,$777 / test the lower 9 bits of the file offset
00005C58 750C                <1>  jnz   short dskw_2
                             <1>        ; bne 2f / if its non-zero, branch; if zero, file offset = 0,
                             <1>               ; / 512, 1024,...(i.e., start of new block)
00005C5A 813D[A4740000]0002- <1>  cmp   dword [u.count], 512
00005C62 0000                <1>
                             <1>        ; cmp u.count,$512. / if zero, is there enough data to fill
                             <1>                       ; / an entire block? (i.e., no. of
00005C64 7305                <1>  jnb   short dskw_3
                             <1>        ; bhis 3f / bytes to be written greater than 512.?
                             <1>               ; / Yes, branch. Don't have to read block
                             <1> dskw_2: ; 2: / in as no past info. is to be saved (the entire block will be
                             <1>               ; / overwritten).
00005C66 E87C050000          <1>  call  dskrd
                             <1>        ; jsr r0,dskrd / no, must retain old info..
                             <1>                    ; / Hence, read block 'r1' into an I/O buffer
                             <1> dskw_3: ; 3:
                             <1>  ; eAX (r1) = block/sector number
00005C6B E8D7050000          <1>  call  wslot
                             <1>        ; jsr r0,wslot / set write and inhibit bits in I/O queue,
                             <1>                  ; / proc. status=0, r5 points to 1st word of data
00005C70 803D[E1740000]00    <1>  cmp   byte [u.kcall], 0
00005C77 770F                <1>  ja    short dskw_5 ; zf=0 -> the caller is 'mkdir'
                             <1>  ;
00005C79 66833D[DF740000]00  <1>  cmp   word [u.pcount], 0
00005C81 7705                <1>  ja    short dskw_5
                             <1> dskw_4:
                             <1>  ; [u.base] = virtual address to transfer (as source address)
00005C83 E806FEFFFF          <1>  call  trans_addr_r ; translate virtual address to physical (r)
                             <1> dskw_5:
                             <1>  ; eBX (r5) = system (I/O) buffer address
00005C88 E87E000000          <1>  call  sioreg
                             <1>        ; jsr r0,sioreg / r3 = no. of bytes of data,
                             <1>                    ; / r1 = address of data, r2 points to location
                             <1>                    ; / in buffer in which to start writing data
                             <1>  ; eSI = file (user data) offset
                             <1>  ; eDI = sector (I/O) buffer offset
                             <1>  ; eCX = byte count
                             <1>  ;
00005C8D F3A4                <1>        rep   movsb
                             <1>        ; movb (r1 )+,(r2)+
                             <1>                  ; / transfer a byte of data to the I/O buffer
                             <1>        ; dec r3 / decrement no. of bytes to be written
                             <1>        ; bne 2b / have all bytes been transferred? No, branch
                             <1>  ; 25/07/2015
                             <1>  ; eax = remain bytes in buffer
                             <1>         ;       (check if remain bytes in the buffer > [u.pcount])
00005C8F 09C0                <1>  or    eax, eax
00005C91 75F0                <1>  jnz   short dskw_4 ; (page end before system buffer end!)
```

```
                             <1> dskw_6:
00005C93 E8CB050000          <1>  call  dskwr
                             <1>        ; jsr r0,dskwr / yes, write the block and the i-node
00005C98 833D[A4740000]00    <1>         cmp     dword [u.count], 0
                             <1>        ; tst u.count / any more data to write?
00005C9F 77A4                <1>  ja    short dskw_1
                             <1>        ; bne 1b / yes, branch
                             <1>  ; 03/08/2013
00005CA1 C605[E1740000]00    <1>  mov   byte [u.kcall], 0
                             <1>  ; 20/09/2013 (;;)
00005CA8 6658                <1>  pop   ax
00005CAA C3                  <1>  retn
                             <1>  ;;jmp       short dskw_ret
                             <1>          ; jmp ret / no, return to the caller via 'ret'
                             <1>
                             <1> cpass: ; / get next character from user area of core and put it in r1
                             <1>  ; 18/10/2015
                             <1>  ; 10/10/2015
                             <1>  ; 10/07/2015
                             <1>  ; 02/07/2015
                             <1>  ; 01/07/2015
                             <1>  ; 24/06/2015
                             <1>  ; 08/06/2015
                             <1>  ; 04/06/2015
                             <1>  ; 20/05/2015
                             <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;      [u.base] = virtual address in user area
                             <1>  ;      [u.count] = byte count (max.)
                             <1>  ;      [u.pcount] = byte count in page (0 = reset)
                             <1>  ; OUTPUTS ->
                             <1>  ;      AL = the character which is pointed by [u.base]
                             <1>  ;      zf = 1 -> transfer count has been completed
                             <1>         ;
                             <1>  ; ((Modified registers:  EAX, EDX, ECX))
                             <1>  ;
                             <1>  ;
00005CAB 833D[A4740000]00    <1>  cmp   dword [u.count], 0  ; 14/08/2013
                             <1>        ; tst u.count / have all the characters been transferred
                             <1>                    ; / (i.e., u.count, # of chars. left
00005CB2 763F                <1>  jna   short cpass_3
                             <1>        ; beq 1f / to be transferred = 0?) yes, branch
00005CB4 FF0D[A4740000]      <1>  dec   dword [u.count]
                             <1>        ; dec u.count / no, decrement u.count
                             <1>         ; 19/05/2015
                             <1>  ;(Retro UNIX 386 v1 - translation from user's virtual address
                             <1>  ;                     to physical address
00005CBA 66833D[DF740000]00  <1>  cmp   word [u.pcount], 0 ; byte count in page = 0 (initial value)
                             <1>                    ; 1-4095 --> use previous physical base address
                             <1>                    ; in [u.pbase]
00005CC2 770E                <1>  ja    short cpass_1
                             <1>  ; 02/07/2015
00005CC4 833D[D7740000]00    <1>         cmp     dword [u.ppgdir], 0  ; is the caller os kernel
00005CCB 7427                            je      short cpass_k       ; (sysexec, '/etc/init') ?
                             <1>  ; 08/06/2015 - 10/07/2015
00005CCD E8BCFDFFFF          <1>  call  trans_addr_r
                             <1> cpass_1:
                             <1>  ; 02/07/2015
                             <1>  ; 24/06/2015
00005CD2 66FF0D[DF740000]    <1>  dec   word [u.pcount]
                             <1> cpass_2:
                             <1>   ;10/10/2015
                             <1>  ; 02/07/2015
00005CD9 8B15[DB740000]      <1>  mov   edx, [u.pbase]
00005CDF 8A02                <1>  mov   al, [edx] ; 10/10/2015
                             <1>        ; movb *u.base,r1 / take the character pointed to
                             <1>                      ; / by u.base and put it in r1
00005CE1 FF05[A8740000]      <1>  inc   dword [u.nread]
                             <1>        ; inc u.nread / increment no. of bytes transferred
00005CE7 FF05[A0740000]      <1>  inc   dword [u.base]
                             <1>        ; inc u.base / increment the buffer address to point to the
                             <1>                 ; / next byte
00005CED FF05[DB740000]      <1>  inc   dword [u.pbase] ; 04/06/2015
                             <1> cpass_3:
00005CF3 C3                  <1>  retn
                             <1>        ; rts r0 / next byte
                             <1>  ; 1:
                             <1>        ; mov (sp)+,r0
                             <1>                 ; / put return address of calling routine into r0
                             <1>        ; mov (sp)+,r1 / i-number in r1
                             <1>        ; rts r0 / non-local return
                             <1> cpass_k:
                             <1>  ; 02/07/2015
                             <1>  ; The caller is os kernel
                             <1>  ; (get sysexec arguments from kernel's memory space)
                             <1>  ;
00005CF4 8B1D[A0740000]      <1>  mov   ebx, [u.base]
00005CFA 66C705[DF740000]00- <1>         mov     word [u.pcount], PAGE_SIZE ; 4096
00005D02 10                  <1>
00005D03 891D[DB740000]      <1>  mov   [u.pbase], ebx
00005D09 EBCE                <1>  jmp   short cpass_2
```

```
                             <1> 
                             <1> sioreg:
                             <1>  ; 25/07/2015
                             <1>  ; 18/07/2015
                             <1>  ; 02/07/2015
                             <1>  ; 17/06/2015
                             <1>  ; 09/06/2015
                             <1>  ; 19/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 12/03/2013 - 22/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;     eBX = system buffer (data) address (r5)
                             <1>  ;     [u.fofp] = pointer to file offset pointer
                             <1>  ;     [u.base] = virtual address of the user buffer
                             <1>  ;     [u.pbase] = physical address of the user buffer
                             <1>  ;     [u.count] = byte count
                             <1>  ;     [u.pcount] = byte count within page frame
                             <1>  ; OUTPUTS ->
                             <1>  ;     eSI = user data offset (r1)
                             <1>  ;     eDI = system (I/O) buffer offset (r2)
                             <1>  ;     eCX = byte count (r3)
                             <1>  ;     EAX = remain bytes after byte count within page frame
                             <1>  ;     (If EAX > 0, transfer will continue from the next page)
                             <1>         ;
                             <1>  ; ((Modified registers:  EDX))
                             <1> 
00005D0B 8B35[90740000]      <1>         mov     esi, [u.fofp]
00005D11 8B3E                <1>         mov     edi, [esi]
                             <1>        ; mov *u.fofp,r2 / file offset (in bytes) is moved to r2
00005D13 89F9                <1>  mov   ecx, edi
                             <1>        ; mov r2,r3 / and also to r3
00005D15 81C900FEFFFF        <1>  or    ecx, 0FFFFFE00h
                             <1>        ; bis $177000,r3 / set bits 9,...,15 of file offset in r3
00005D1B 81E7FF010000        <1>  and   edi, 1FFh
                             <1>        ; bic $!777,r2 / calculate file offset mod 512.
00005D21 01DF                <1>  add   edi, ebx ; EBX = system buffer (data) address
                             <1>        ; add r5,r2 / r2 now points to 1st byte in system buffer
                             <1>               ; / where data is to be placed
                             <1>                ; mov u.base,r1 / address of data is in r1
00005D23 F7D9                <1>  neg   ecx
                             <1>        ; neg r3 / 512 - file offset (mod512.) in r3
                             <1>              ; / (i.e., the no. of free bytes in the file block)
00005D25 3B0D[A4740000]      <1>  cmp   ecx, [u.count]
                             <1>        ; cmp r3,u.count / compare this with the no. of data bytes
                             <1>                    ; / to be written to the file
00005D2B 7606                <1>  jna   short sioreg_0
                             <1>        ; blos      2f / if less than branch. Use the no. of free bytes
                             <1>              ; / in the file block as the number to be written
00005D2D 8B0D[A4740000]      <1>  mov   ecx, [u.count]
                             <1>        ; mov u.count,r3 / if greater than, use the no. of data
                             <1>                    ; / bytes as the number to be written
                             <1> sioreg_0:
                             <1>  ; 17/06/2015
00005D33 803D[E1740000]00    <1>  cmp   byte [u.kcall], 0
00005D3A 7613                <1>  jna   short sioreg_1
                             <1>  ; 25/07/2015
                             <1>   ; the caller is 'mkdir' or 'namei'
00005D3C A1[A0740000]        <1>  mov   eax, [u.base] ; 25/07/2015
00005D41 A3[DB740000]        <1>  mov   [u.pbase], eax ; physical address = virtual address
00005D46 66890D[DF740000]    <1>  mov   word [u.pcount], cx ; remain bytes in buffer (1 sector)
00005D4D EB0B                <1>  jmp   short sioreg_2
                             <1> sioreg_1:
                             <1>  ; 25/07/2015
                             <1>  ; 18/07/2015
                             <1>  ; 09/06/2015
00005D4F 0FB715[DF740000]    <1>  movzx edx, word [u.pcount]
                             <1>        ; ecx and [u.pcount] are always > 0, here
00005D56 39D1                <1>  cmp   ecx, edx
00005D58 772A                <1>  ja    short sioreg_4 ; transfer count > [u.pcount]
                             <1> sioreg_2: ; 2:
00005D5A 31C0                <1>  xor   eax, eax ; 25/07/2015
                             <1> sioreg_3:
00005D5C 010D[A8740000]      <1>  add   [u.nread], ecx
                             <1>        ; add r3,u.nread / r3 + number of bytes xmitted
                             <1>                      ; / during write is put into u.nread
00005D62 290D[A4740000]      <1>  sub   [u.count], ecx
                             <1>        ; sub r3,u.count / u.count = no. of bytes that still
                             <1>                    ; / must be written or read
00005D68 010D[A0740000]      <1>  add   [u.base], ecx
                             <1>        ; add r3,u.base / u.base points to the 1st of the remaining
                             <1>                   ; / data bytes
00005D6E 010E                <1>         add  [esi], ecx
                             <1>        ; add r3,*u.fofp / new file offset = number of bytes done
                             <1>                    ; / + old file offset
                             <1>  ; 25/07/2015
00005D70 8B35[DB740000]      <1>  mov   esi, [u.pbase]
00005D76 66290D[DF740000]    <1>  sub   [u.pcount], cx
00005D7D 010D[DB740000]      <1>  add   [u.pbase], ecx
00005D83 C3                  <1>         retn
                             <1>        ; rts r0
                             <1>        ; transfer count > [u.pcount]
                             <1> sioreg_4:
```

```
                             <1>  ; 25/07/2015
                             <1>  ; transfer count > [u.pcount]
                             <1>  ; (ecx > edx)
00005D84 89C8                <1>  mov   eax, ecx
00005D86 29D0                <1>  sub   eax, edx ; remain bytes for 1 sector (block) transfer
00005D88 89D1                <1>  mov   ecx, edx ; current transfer count = [u.pcount]
00005D8A EBD0                <1>  jmp   short sioreg_3
                                 %include 'u7.s'        ; 18/04/2015
                             <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS7.INC
                             <1> ; Last Modification: 14/11/2015
                             <1> ; -------------------------------------------------------------------------
                             <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
                             <1> ; (v0.1 - Beginning: 11/07/2012)
                             <1> ;
                             <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                             <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                             <1> ; <Bell Laboratories (17/3/1972)>
                             <1> ; <Preliminary Release of UNIX Implementation Document>
                             <1> ;
                             <1> ; Retro UNIX 8086 v1 - U7.ASM (13/07/2014) //// UNIX v1 -> u7.s
                             <1> ;
                             <1> ; *************************************************************************
                             <1>
                             <1> sysmount: ; / mount file system; args special; name
                             <1>  ; 14/11/2015
                             <1>  ; 24/10/2015
                             <1>  ; 13/10/2015
                             <1>  ; 10/07/2015
                             <1>  ; 16/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 09/07/2013 - 04/11/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'sysmount' anounces to the system that a removable
                             <1>  ; file system has been mounted on a special file.
                             <1>  ; The device number of the special file is obtained via
                             <1>  ; a call to 'getspl'. It is put in the I/O queue entry for
                             <1>  ; dismountable file system (sb1) and the I/O queue entry is
                             <1>  ; set up to read (bit 10 is set). 'ppoke' is then called to
                             <1>  ; to read file system into core, i.e. the first block on the
                             <1>  ; mountable file system is read in. This block is super block
                             <1>  ; for the file system. This call is super user restricted.
                             <1>  ;
                             <1>  ; Calling sequence:
                             <1>  ;     sysmount; special; name
                             <1>  ; Arguments:
                             <1>  ;     special - pointer to name of special file (device)
                             <1>  ;     name -  pointer to name of the root directory of the
                             <1>  ;           newly mounted file system. 'name' should
                             <1>  ;           always be a directory.
                             <1>  ; Inputs: -
                             <1>  ; Outputs: -
                             <1>  ; .............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;      'sysmount' system call has two arguments; so,
                             <1>  ;     * 1st argument, special is pointed to by BX register
                             <1>  ;     * 2nd argument, name is in CX register
                             <1>  ;
                             <1>  ;     NOTE: Device numbers, names and related procedures are
                             <1>  ;            already modified for IBM PC compatibility and
                             <1>  ;            Retro UNIX 8086 v1 device configuration.
                             <1>
                             <1>  ;call arg2
                             <1>        ; jsr r0,arg2 / get arguments special and name
00005D8C 891D[98740000]      <1>  mov   [u.namep], ebx
00005D92 51                  <1>  push  ecx ; directory name
00005D93 66833D[6C740000]00  <1>  cmp   word [mnti], 0
                             <1>        ; tst mnti / is the i-number of the cross device file
                             <1>               ; / zero?
                             <1>  ;ja   error
                             <1>              ; bne errora / no, error
00005D9B 0F87E9000000        <1>  ja    sysmnt_err0
                             <1>  ;
00005DA1 E8CC000000          <1>  call  getspl
                             <1>        ; jsr r0,getspl / get special files device number in r1
                             <1>  ; 13/10/2015
00005DA6 0FB7D8              <1>  movzx ebx, ax ; ; Retro UNIX 8086 v1 device number (0 to 5)
00005DA9 F683[246E0000]80    <1>         test    byte [ebx+drv.status], 80h ; 24/10/2015
00005DB0 750F                <1>  jnz   short sysmnt_1
                             <1> sysmnt_err1:
00005DB2 C705[CF740000]0F00- <1>         mov     dword [u.error], ERR_DRV_NOT_RDY ; drive not ready !
00005DBA 0000                <1>
00005DBC E9BBE2FFFF          <1>  jmp   error
                             <1> sysmnt_1:
00005DC1 8F05[98740000]      <1>  pop   dword [u.namep]
                             <1>                ; mov (sp)+,u.namep / put the name of file to be placed
                             <1>                      ; / on the device
                             <1>  ; 14/11/2015
00005DC7 53                  <1>  push  ebx ; 13/10/2015
                             <1>        ; mov r1,-(sp) / save the device number
                             <1>         ;
00005DC8 E85EF1FFFF          <1>  call  namei
                             <1>  ;or   ax, ax ; Retro UNIX 8086 v1 modification !
                             <1>               ; ax = 0 -> file not found
                             <1>  ;jz   error
                             <1>  ;jc   error
                             <1>        ; jsr r0,namei / get the i-number of the file
                             <1>                    ; br errora
00005DCD 730F                <1>  jnc   short sysmnt_2
                             <1> sysmnt_err2:
00005DCF C705[CF740000]0C00- <1>         mov     dword [u.error], ERR_FILE_NOT_FOUND ; drive not ready !
00005DD7 0000                <1>
00005DD9 E99EE2FFFF          <1>  jmp   error
                             <1> sysmnt_2:
00005DDE 66A3[6C740000]      <1>  mov   [mnti], ax
                             <1>              ; mov r1,mnti / put it in mnti
                             <1> ;mov   ebx, sb1 ; super block buffer (of mounted disk)
                             <1> sysmnt_3: ;1:
                             <1>         ;cmp byte [ebx+1], 0
                             <1>        ; tstb sb1+1 / is 15th bit of I/O queue entry for
                             <1>                 ; / dismountable device set?
                             <1>         ;jna short sysmnt_4
                             <1>        ; bne 1b / (inhibit bit) yes, skip writing
                             <1>  ;call idle  ; (wait for hardware interrupt)
                             <1>  ;jmp  short sysmnt_3
                             <1> sysmnt_4:
00005DE4 58                  <1>  pop   eax ; Retro UNIX 8086 v1 device number/ID (0 to 5)
00005DE5 A2[69740000]        <1>  mov   [mdev], al
                             <1>        ; mov  (sp),mntd / no, put the device number in mntd
00005DEA 8803                <1>  mov   [ebx], al
                             <1>              ; movb (sp),sb1 / put the device number in the lower byte
                             <1>                    ; / of the I/O queue entry
                             <1>  ;mov  byte [cdev], 1 ; mounted device/drive
                             <1>              ; mov (sp)+,cdev / put device number in cdev
00005DEC 66810B0004          <1>         or   word [ebx], 400h ; Bit 10, 'read' flag/bit
                             <1>        ; bis $2000,sb1 / set the read bit
                             <1>  ; Retro UNIX 386 v1 modification :
                             <1>  ;     32 bit block number at buffer header offset 4
00005DF1 C7430401000000      <1>  mov   dword [ebx+4], 1 ; physical block number = 1
00005DF8 E8A3050000          <1>  call  diskio
00005DFD 731C                <1>  jnc   short sysmnt_5
00005DFF 31C0                <1>  xor   eax, eax
00005E01 66A3[6C740000]      <1>  mov   [mnti], ax ; 0
00005E07 A2[69740000]        <1>  mov   [mdev], al ; 0
                             <1>  ;mov  [cdev], al ; 0
                             <1> sysmnt_invd:
                             <1>  ; 14/11/2015
00005E0C FEC8                <1>  dec   al
00005E0E 8903                <1>  mov   [ebx], eax ; 000000FFh
00005E10 FEC0                <1>  inc   al
00005E12 48                  <1>  dec   eax
00005E13 894304              <1>  mov   [ebx+4], eax ; 0FFFFFFFFh
00005E16 E961E2FFFF          <1>  jmp   error
                             <1> sysmnt_5:
                             <1>  ; 14/11/2015 (Retro UNIX 386 v1 modification)
                             <1>  ; (Following check is needed to prevent mounting an
                             <1>  ; in valid valid file system (in valid super block).
                             <1>  ;
00005E1B 0FB603              <1>  movzx eax, byte [ebx] ; device number
00005E1E C0E002              <1>  shl   al, 2 ; 4*index
00005E21 8B88[086E0000]      <1>  mov   ecx, [eax+drv.size] ; volume (fs) size
00005E27 C1E103              <1>  shl   ecx, 3
00005E2A 0FB715[2C7F0000]    <1>  movzx edx, word [sb1+4] ; the 1st data word
00005E31 39D1                <1>  cmp   ecx, edx ; compare free map bits and volume size
                             <1>               ; (in sectors), if they are not equal
                             <1>               ; the disk to be mounted is an...
00005E33 75D7                <1>  jne   short sysmnt_invd ; invalid disk !
                             <1>               ; (which has not got a valid super block)
                             <1>  ;
00005E35 C6430100            <1>  mov   byte [ebx+1], 0
                             <1>              ; jsr r0,ppoke / read in entire file system
                             <1> ;sysmnt_6: ;1:
                             <1>  ;;cmp byte [sb1+1], 0
                             <1>        ; tstb   sb1+1 / done reading?
                             <1>        ;;jna sysret
                             <1>  ;;call      idle ; (wait for hardware interrupt)
                             <1>  ;;jmp short sysmnt_6
                             <1>        ;bne 1b / no, wait
                             <1>              ;br sysreta / yes
00005E39 E95EE2FFFF          <1>  jmp   sysret
                             <1>
                             <1> sysumount: ; / special dismount file system
                             <1>  ; 16/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 09/07/2013 - 04/11/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 04/11/2013
                             <1>  ; 09/07/2013
                             <1>  ; 'sysumount' anounces to the system that the special file,
                             <1>  ; indicated as an argument is no longer contain a removable
                             <1>  ; file system. 'getspl' gets the device number of the special
                             <1>  ; file. If no file system was mounted on that device an error
                             <1>  ; occurs. 'mntd' and 'mnti' are cleared and control is passed
                             <1>  ; to 'sysret'.
                             <1>  ;
                             <1>  ; Calling sequence:
```

```
                             <1>  ;     sysmount; special
                             <1>  ; Arguments:
                             <1>  ;     special - special file to dismount (device)
                             <1>  ;
                             <1>  ; Inputs: -
                             <1>  ; Outputs: -
                             <1>  ; ...............................................................
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;       'sysumount' system call has one argument; so,
                             <1>  ;     * Single argument, special is pointed to by BX register
                             <1>  ;
                             <1>
                             <1>  ;mov  ax, 1 ; one/single argument, put argument in BX
                             <1>  ;call arg
                             <1>         ; jsr r0,arg; u.namep / point u.namep to special
00005E3E 891D[98740000]      <1>          mov  [u.namep], ebx
00005E44 E829000000          <1>  call  getspl
                             <1>         ; jsr r0,getspl / get the device number in r1
00005E49 3A05[69740000]      <1>  cmp   al, [mdev]
                             <1>         ; cmp r1,mntd / is it equal to the last device mounted?
00005E4F 7539                <1>  jne   short sysmnt_err0 ; 'permission denied !' error
                             <1>  ;jne  error
                             <1>                ; bne errora / no error
00005E51 30C0                <1>  xor   al, al ; ah = 0
                             <1> sysumnt_0: ;1:
00005E53 3805[297F0000]      <1>         cmp   [sb1+1], al ; 0
                             <1>         ; tstb sb1+1 / yes, is the device still doing I/O
                             <1>                   ; / (inhibit bit set)?
00005E59 7607                <1>  jna   short sysumnt_1
                             <1>         ; bne 1b / yes, wait
00005E5B E82DF6FFFF          <1>  call  idle ; (wait for hardware interrupt)
00005E60 EBF1                <1>  jmp   short sysumnt_0
                             <1> sysumnt_1:
00005E62 A2[69740000]        <1>  mov   [mdev], al
                             <1>         ; clr mntd / no, clear these
00005E67 66A3[6C740000]      <1>         mov   [mnti], ax
                             <1>                ; clr mnti
00005E6D E92AE2FFFF          <1>          jmp  sysret
                             <1>         ; br sysreta / return
                             <1>
                             <1> getspl: ; / get device number from a special file name
00005E72 E8B4F0FFFF          <1>  call  namei
                             <1>  ;or   ax, ax ; Retro UNIX 8086 v1 modification !
                             <1>                 ; ax = 0 -> file not found
00005E77 0F8252FFFFFF        <1>          jc      sysmnt_err2 ; 'file not found !' error
                             <1>  ;jz   error
                             <1>  ;jc   error
                             <1>         ; jsr r0,namei / get the i-number of the special file
                             <1>                   ; br errora / no such file
00005E7D 6683E803            <1>          sub  ax, 3 ; Retro UNIX 8086 v1 modification !
                             <1>                ;     i-number-3, 0 = fd0, 5 = hd3
                             <1>         ; sub $4,r1 / i-number-4 rk=1,tap=2+n
00005E81 7207                <1>          jc   short sysmnt_err0 ; 'permission denied !' error
                             <1>  ;jc   error
                             <1>         ; ble errora / less than 0?  yes, error
00005E83 6683F805            <1>          cmp  ax, 5 ;
                             <1>         ; cmp  r1,$9. / greater than 9  tap 7
00005E87 7701                <1>  ja    short sysmnt_err0 ; 'permission denied !' error
                             <1>  ;ja   error
                             <1>                ; bgt errora / yes, error
                             <1>          ; AX = Retro UNIX 8086 v1 Device Number (0 to 5)
                             <1> iopen_retn:
00005E89 C3                  <1>  retn
                             <1>         ; rts    r0 / return with device number in r1
                             <1> sysmnt_err0:
00005E8A C705[CF740000]0B00- <1>  mov   dword [u.error], ERR_FILE_ACCESS ; permission denied !
00005E92 0000                <1>
00005E94 E9E3E1FFFF          <1>  jmp   error
                             <1> iopen:
                             <1>  ; 19/05/2015
                             <1>  ; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 21/05/2013 - 27/08/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; open file whose i-number is in r1
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - inode number
                             <1>  ; OUTPUTS ->
                             <1>  ;    file's inode in core
                             <1>  ;    r1 - inode number (positive)
                             <1>  ;
                             <1>  ; ((AX = R1))
                             <1>          ; ((Modified registers: edx, ebx, ecx, esi, edi, ebp))
                             <1>  ;
                             <1> ; / open file whose i-number is in r1
00005E99 F6C480              <1>  test  ah, 80h ; Bit 15 of AX
                             <1>         ;tst r1 / write or read access?
00005E9C 756A                <1>          jnz  short iopen_2
                             <1>         ;blt 2f / write, go to 2f
00005E9E B202                <1>  mov   dl, 2 ; read access
00005EA0 E850F9FFFF          <1>  call  access
```

```
                             <1>              ; jsr r0,access; 2
                             <1>  ; / get inode into core with read access
                             <1>  ; DL=2
                             <1> iopen_0:
00005EA5 6683F828            <1>         cmp  ax, 40
                             <1>        ; cmp r1,$40. / is it a special file
00005EA9 77DE                <1>         ja   short iopen_retn
                             <1>        ;bgt  3f / no. 3f
00005EAB 6650                <1>  push  ax
                             <1>        ; mov r1,-(sp) / yes, figure out
00005EAD 0FB6D8              <1>  movzx ebx, al
00005EB0 66C1E302            <1>  shl   bx, 2
                             <1>        ; asl r1
00005EB4 81C3[B85E0000]      <1>         add     ebx, iopen_1 - 4
00005EBA FF23                <1>  jmp   dword [ebx]
                             <1>              ; jmp *1f-2(r1) / which one and transfer to it
                             <1> iopen_1: ; 1:
00005EBC [225F0000]          <1>  dd    otty ; tty, AX = 1 (runix)
                             <1>         ;otty / tty ; r1=2
                             <1>               ;oppt / ppt ; r1=4
00005EC0 [BF5F0000]          <1>  dd    sret ; mem, AX = 2 (runix)
                             <1>         ;sret / mem ; r1=6
                             <1>         ;sret / rf0
                             <1>               ;sret / rk0
                             <1>               ;sret / tap0
                             <1>               ;sret / tap1
                             <1>               ;sret / tap2
                             <1>               ;sret / tap3
                             <1>               ;sret / tap4
                             <1>               ;sret / tap5
                             <1>               ;sret / tap6
                             <1>               ;sret / tap7
00005EC4 [BF5F0000]          <1>         dd   sret ; fd0, AX = 3 (runix only)
00005EC8 [BF5F0000]          <1>         dd   sret ; fd1, AX = 4 (runix only)
00005ECC [BF5F0000]          <1>         dd   sret ; hd0, AX = 5 (runix only)
00005ED0 [BF5F0000]          <1>         dd   sret ; hd1, AX = 6 (runix only)
00005ED4 [BF5F0000]          <1>         dd   sret ; hd2, AX = 7 (runix only)
00005ED8 [BF5F0000]          <1>         dd   sret ; hd3, AX = 8 (runix only)
                             <1>  ;dd   error ; lpr, AX = 9 (error !)
00005EDC [BF5F0000]          <1>         dd      sret ; lpr, AX = 9 (runix)
00005EE0 [315F0000]          <1>  dd    ocvt ; tty0, AX = 10 (runix)
                             <1>         ;ocvt / tty0
00005EE4 [315F0000]          <1>  dd    ocvt ; tty1, AX = 11 (runix)
                             <1>         ;ocvt / tty1
00005EE8 [315F0000]          <1>  dd    ocvt ; tty2, AX = 12 (runix)
                             <1>         ;ocvt / tty2
00005EEC [315F0000]          <1>  dd    ocvt ; tty3, AX = 13 (runix)
                             <1>         ;ocvt / tty3
00005EF0 [315F0000]          <1>  dd    ocvt ; tty4, AX = 14 (runix)
                             <1>         ;ocvt / tty4
00005EF4 [315F0000]          <1>  dd    ocvt ; tty5, AX = 15 (runix)
                             <1>         ;ocvt / tty5
00005EF8 [315F0000]          <1>  dd    ocvt ; tty6, AX = 16 (runix)
                             <1>         ;ocvt / tty6
00005EFC [315F0000]          <1>  dd    ocvt ; tty7, AX = 17 (runix)
                             <1>         ;ocvt / tty7
00005F00 [315F0000]          <1>  dd    ocvt ; COM1, AX = 18 (runix only)
                             <1>         ;error / crd
00005F04 [315F0000]          <1>  dd    ocvt ; COM2, AX = 19 (runix only)
                             <1>
                             <1> iopen_2: ; 2: / check open write access
00005F08 66F7D8              <1>  neg   ax
                             <1>        ;neg r1 / make inode number positive
00005F0B B201                <1>  mov   dl, 1 ; write access
00005F0D E8E3F8FFFF          <1>  call  access
                             <1>        ;jsr r0,access; 1 / get inode in core
                             <1>  ; DL=1
00005F12 66F705[56710000]00- <1>  test  word [i.flgs], 4000h ; Bit 14 : Directory flag
00005F1A 40                  <1>
                             <1>        ;bit $40000,i.flgs / is it a directory?
00005F1B 7488                <1>  jz    short iopen_0
                             <1>  ;mov  [u.error], ERR_DIR_ACCESS
                             <1>  ;jmp  error ; permission denied !
00005F1D E968FFFFFF          <1>  jmp   sysmnt_err0
                             <1>  ;;jnz error
                             <1>              ; bne 2f / yes, transfer (error)
                             <1>         ;;jmp     short iopen_0
                             <1>  ;cmp  ax, 40
                             <1>        ; cmp r1,$40. / no, is it a special file?
                             <1>         ;ja  short iopen_2
                             <1>         ;bgt 3f / no, return
                             <1>  ;push ax
                             <1>        ;mov r1,-(sp) / yes
                             <1>  ;movzx      ebx, al
                             <1>  ;shl  bx, 1
                             <1>        ; asl r1
                             <1>  ;add  ebx, ipen_3 - 2
                             <1>  ;jmp  dword [ebx]
                             <1>        ; jmp *1f-2(r1) / figure out
                             <1>              ; / which special file it is and transfer
                             <1> ;iopen_3: ; 1:
                             <1> ;dd    otty ; tty, AX = 1 (runix)
                             <1>         ;otty / tty ; r1=2
                             <1>               ;leadr / ppt ; r1=4
                             <1> ;dd    sret ; mem, AX = 2 (runix)
                             <1>         ;sret / mem ; r1=6
                             <1>         ;sret / rf0
                             <1>               ;sret / rk0
                             <1>               ;sret / tap0
                             <1>               ;sret / tap1
                             <1>               ;sret / tap2
                             <1>               ;sret / tap3
                             <1>               ;sret / tap4
                             <1>               ;sret / tap5
                             <1>               ;sret / tap6
                             <1>               ;sret / tap7
                             <1> ;dd    sret ; fd0, AX = 3 (runix only)
                             <1> ;dd    sret ; fd1, AX = 4 (runix only)
                             <1> ;dd    sret ; hd0, AX = 5 (runix only)
                             <1> ;dd    sret ; hd1, AX = 6 (runix only)
                             <1> ;dd    sret ; hd2, AX = 7 (runix only)
                             <1> ;dd    sret ; hd3, AX = 8 (runix only)
                             <1> ;dd    sret ; lpr, AX = 9  (runix)
                             <1>  ;dd   ejec ; lpr, AX = 9  (runix)
                             <1> ;dd    sret ; tty0, AX = 10 (runix)
                             <1>         ;ocvt / tty0
                             <1> ;dd    sret ; tty1, AX = 11 (runix)
                             <1>         ;ocvt / tty1
                             <1> ;dd    sret ; tty2, AX = 12 (runix)
                             <1>         ;ocvt / tty2
                             <1> ;dd    sret ; tty3, AX = 13 (runix)
                             <1>         ;ocvt / tty3
                             <1> ;dd    sret ; tty4, AX = 14 (runix)
                             <1>         ;ocvt / tty4
                             <1> ;dd    sret ; tty5, AX = 15 (runix)
                             <1>         ;ocvt / tty5
                             <1> ;dd    sret ; tty6, AX = 16 (runix)
                             <1>         ;ocvt / tty6
                             <1> ;dd    sret ; tty7, AX = 17 (runix)
                             <1>         ;ocvt / tty7
                             <1> ;dd    ocvt ; COM1, AX = 18 (runix only)
                             <1>         ;/ ejec / lpr
                             <1> ;dd    ocvt ; COM2, AX = 19 (runix only)
                             <1>
                             <1>
                             <1> otty: ;/ open console tty for reading or writing
                             <1>  ; 16/11/2015
                             <1>  ; 12/11/2015
                             <1>  ; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 21/05/2013 - 13/07/2014 (Retro UNIX 8086 v1)
                             <1>  ; 16/07/2013
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;  If a tty is open for read or write by
                             <1>  ;     a process (u.uno), only same process can open
                             <1>  ;     same tty to write or read (R->R&W or W->W&R).
                             <1>  ;
                             <1>  ; (INPUT: DL=2 for Read, DL=1 for Write, DL=0 for sysstty)
                             <1>  ;
00005F22 0FB61D[C9740000]    <1>  movzx ebx, byte [u.uno] ; process number
00005F29 8A83[D5710000]      <1>  mov   al, [ebx+p.ttyc-1] ; current/console tty
                             <1>  ; 13/01/2014
00005F2F EB02                <1>  jmp   short ottyp
                             <1> ocvt:
00005F31 2C0A                <1>  sub   al, 10
                             <1> ottyp:
                             <1>  ; 16/11/2015
                             <1>  ; 12/11/2015
                             <1>  ; 18/05/2015 (32 bit modifications)
                             <1>  ; 06/12/2013 - 13/07/2014
00005F33 88C6                <1>  mov   dh, al ; tty number
00005F35 0FB6D8              <1>  movzx       ebx, al ; AL = tty number (0 to 9), AH = 0
00005F38 D0E3                <1>  shl   bl, 1  ; aligned to word
                             <1>  ;26/01/2014
00005F3A 81C3[F4700000]      <1>  add   ebx, ttyl
00005F40 668B0B              <1>  mov   cx, [ebx]
                             <1>           ; CL = lock value (0 or process number)
                             <1>           ; CH = open count
00005F43 20C9                <1>  and   cl, cl
                             <1>  ; 13/01/2014
00005F45 7439                <1>  jz    short otty_ret
                             <1>  ;
                             <1>  ; 16/11/2015
00005F47 3A0D[C9740000]      <1>  cmp   cl, [u.uno]
00005F4D 745C                <1>  je    short ottys_3
                             <1>  ;
00005F4F 0FB6D9              <1>  movzx       ebx, cl ; the process which has locked the tty
00005F52 D0E3                <1>  shl   bl, 1
00005F54 668B83[74710000]    <1>  mov   ax, [ebx+p.pid-2]
                             <1>  ;movzx      ebx, byte [u.uno]
00005F5B 8A1D[C9740000]      <1>  mov   bl, [u.uno]
00005F61 D0E3                <1>  shl   bl, 1
00005F63 663B83[94710000]    <1>  cmp   ax, [ebx+p.ppid-2]
00005F6A 743F                <1>  je    short ottys_3  ; 16/11/2015
                             <1>  ;
                             <1>  ; the tty is locked by another process
                             <1>  ; except the parent process (p.ppid)
                             <1>        ;
00005F6C C705[CF740000]0B00- <1>  mov     dword [u.error], ERR_DEV_ACCESS
00005F74 0000                <1>
                             <1>             ; permission denied ! error
                             <1> otty_err: ; 13/01/2014
00005F76 08D2                <1>  or    dl, dl ; DL = 0 -> called by sysstty
00005F78 0F85FEE0FFFF        <1>  jnz   error
00005F7E F9                  <1>  stc
00005F7F C3                  <1>  retn
                             <1> otty_ret:
                             <1>  ; 13/01/2014
00005F80 80FE07              <1>  cmp   dh, 7
00005F83 761C                <1>  jna   short ottys_2
                             <1>  ; 16/11/2015
                             <1> com_port_check:
00005F85 BE[12710000]        <1>  mov   esi, com1p
00005F8A 80FE08              <1>  cmp   dh, 8 ; COM1 (tty8) ?
00005F8D 7601                <1>  jna   short ottys_1 ; yes, it is COM1
00005F8F 46                  <1>  inc   esi   ; no, it is COM2 (tty9)
                             <1> ottys_1:
                             <1>  ; 12/11/2015
00005F90 803E00              <1>  cmp   byte [esi], 0 ; E3h (or 23h)
00005F93 770C                <1>  ja    short com_port_ready
                             <1>  ;
00005F95 C705[CF740000]0F00- <1>        mov     dword [u.error], ERR_DEV_NOT_RDY
00005F9D 0000                <1>
                             <1>                 ; device not ready ! error
00005F9F EBD5                <1>  jmp   short otty_err
                             <1> com_port_ready:
                             <1> ottys_2:
00005FA1 08C9                <1>  or    cl, cl  ; cl = lock/owner, ch = open count
00005FA3 7506                <1>  jnz   short ottys_3
00005FA5 8A0D[C9740000]      <1>  mov   cl, [u.uno]
                             <1> ottys_3:
00005FAB FEC5                <1>  inc   ch
00005FAD 66890B              <1>  mov   [ebx], cx ; set tty lock again
                             <1>  ; 06/12/2013
00005FB0 FEC6                <1>  inc   dh ; tty number + 1
00005FB2 BB[B0740000]        <1>  mov   ebx, u.ttyp
                             <1>  ; 13/01/2014
00005FB7 F6C202              <1>  test  dl, 2 ; open for read sign
00005FBA 7501                <1>  jnz   short ottys_4
00005FBC 43                  <1>  inc   ebx
                             <1> ottys_4:
                             <1>  ; Set 'u.ttyp' ('the recent TTY') value
00005FBD 8833                <1>  mov   [ebx], dh ; tty number + 1
                             <1> sret:
00005FBF 08D2                <1>  or    dl, dl ; sysstty system call check (DL=0)
00005FC1 7402                <1>  jz    short iclose_retn
00005FC3 6658                <1>  pop   ax
                             <1> iclose_retn:
00005FC5 C3                  <1>  retn
                             <1>
                             <1>  ;
                             <1>  ; Original UNIX v1 'otty' routine:
                             <1>  ;
                             <1>  ;mov    $100,*$tks / set interrupt enable bit (zero others) in
                             <1>        ;                  / reader status reg
                             <1>        ;mov    $100,*$tps / set interrupt enable bit (zero others) in
                             <1>        ;                  / punch status reg
                             <1>        ;mov    tty+[ntty*8]-8+6,r5 / r5 points to the header of the
                             <1>        ;                           / console tty buffer
                             <1>        ;incb   (r5) / increment the count of processes that opened the
                             <1>        ;            / console tty
                             <1>        ;tst    u.ttyp / is there a process control tty (i.e., has a tty
                             <1>        ;              / buffer header
                             <1>        ;bne    sret / address been loaded into u.ttyp yet)?  yes, branch
                             <1>        ;mov    r5,u.ttyp / no, make the console tty the process control
                             <1>        ;                 / tty
                             <1>        ;br     sret / ?
                             <1> ;sret:
                             <1>       ;clr *$ps / set processor priority to zero
                             <1> ;pop   ax
                             <1>            ;mov (sp)+,r1 / pop stack to r1
                             <1> ;3:
                             <1> ;retn
                             <1>            ;rts r0
                             <1>
                             <1> ;ocvt: ; < open tty >
                             <1>  ; 13/01/2014
                             <1>  ; 06/12/2013 (major modification: p.ttyc, u.ttyp)
                             <1>  ; 24/09/2013 consistency check -> ok
                             <1>  ; 16/09/2013
                             <1>  ; 03/09/2013
                             <1>  ; 27/08/2013
                             <1>  ; 16/08/2013
                             <1>  ; 16/07/2013
                             <1>  ; 27/05/2013
                             <1>  ; 21/05/2013
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification !
                             <1>  ;
                             <1>  ; In original UNIX v1, 'ocvt' routine
                             <1>  ;          (exactly different than this one)
                             <1>  ;     was in 'u9.s' file.
                             <1>  ;
                             <1>  ; 16/07/2013
                             <1>  ; Retro UNIX 8086 v1 modification:
                             <1>  ;  If a tty is open for read or write by
                             <1>  ;     a process (u.uno), only same process can open
                             <1>  ;     same tty to write or read (R->R&W or W->W&R).
                             <1>  ;
                             <1>  ; INPUT: DL=2 for Read DL=1 for Write
                             <1>
                             <1>  ; 16/09/2013
                             <1>  ; sub      al, 10
                             <1>
                             <1>  ; 06/12/2013
                             <1>  ;cmp  al, 7
                             <1>         ;jna     short ottyp
                             <1>  ; 13/01/2014
                             <1>  ;jmp  short ottyp
                             <1>
                             <1>
                             <1> ;oppt: / open paper tape for reading or writing
                             <1> ;        mov    $100,*$prs / set reader interrupt enable bit
                             <1> ;        tstb   pptiflg / is file already open
                             <1> ;        bne    2f / yes, branch
                             <1> ;1:
                             <1> ;        mov    $240,*$ps / no, set processor priority to 5
                             <1> ;        jsr    r0,getc; 2 / remove all entries in clist
                             <1> ;               br .+4 / for paper tape input and place in free list
                             <1> ;        br     1b
                             <1> ;        movb   $2,pptiflg / set pptiflg to indicate file just open
                             <1> ;        movb   $10.,toutt+1 / place 10 in paper tape input tout entry
                             <1> ;        br     sret
                             <1> ;2:
                             <1> ;        jmp    error / file already open
                             <1>
                             <1> iclose:
                             <1>  ; 19/05/2015
                             <1>  ; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 21/05/2013 - 13/01/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; close file whose i-number is in r1
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - inode number
                             <1>  ; OUTPUTS ->
                             <1>  ;    file's inode in core
                             <1>  ;    r1 - inode number (positive)
                             <1>  ;
                             <1>  ; ((AX = R1))
                             <1>        ;    ((Modified registers: -ebx-, edx))
                             <1>  ;
                             <1> ;/ close file whose i-number is in r1
00005FC6 B202                <1>  mov   dl, 2 ; 12/01/2014
00005FC8 F6C480              <1>  test  ah, 80h ; Bit 15 of AX
                             <1>        ;tst r1 / test i-number
                             <1>         ;jnz short iclose_2
                             <1>        ;blt 2f / if neg., branch
00005FCB 7405                <1>  jz    short iclose_0 ; 30/07/2013
                             <1>  ; 16/07/2013
00005FCD 66F7D8              <1>  neg   ax ; make it positive
                             <1>  ; 12/01/2014
00005FD0 FECA                <1>  dec   dl ; dl = 1 (open for write)
                             <1> iclose_0:
00005FD2 6683F828            <1>  cmp   ax, 40
                             <1>        ;cmp r1,$40. / is it a special file
00005FD6 77ED                <1>         ja   short iclose_retn  ; 13/01/2014
                             <1>        ;bgt 3b / no, return
                             <1>  ; 12/01/2014
                             <1>  ; DL=2 -> special file was opened for reading
                             <1>  ; DL=1 -> special file was opened for writing
00005FD8 6650                <1>  push  ax
                             <1>        ;mov r1,-(sp) / yes, save r1 on stack
00005FDA 0FB6D8              <1>  movzx ebx, al
00005FDD 66C1E302            <1>  shl   bx, 2
                             <1>        ; asl r1
00005FE1 81C3[E55F0000]      <1>  add   ebx, iclose_1 - 4
00005FE7 FF23                <1>  jmp   dword [ebx]
                             <1>        ; jmp *1f-2(r1) / compute jump address and transfer
                             <1> iclose_1 :
00005FE9 [35600000]          <1>  dd    ctty ; tty, AX = 1 (runix)
00005FED [86600000]          <1>  dd    cret ; mem, AX = 2 (runix)
00005FF1 [86600000]          <1>  dd    cret ; fd0, AX = 3 (runix only)
00005FF5 [86600000]          <1>  dd    cret ; fd1, AX = 4 (runix only)
00005FF9 [86600000]          <1>  dd    cret ; hd0, AX = 5 (runix only)
00005FFD [86600000]          <1>  dd    cret ; hd1, AX = 6 (runix only)
00006001 [86600000]          <1>  dd    cret ; hd2, AX = 7 (runix only)
00006005 [86600000]          <1>  dd    cret ; hd3, AX = 8 (runix only)
00006009 [86600000]          <1>  dd    cret ; lpr, AX = 9 (runix)
```

```
                             <1>  ;dd   error; lpr, AX = 9 (error !)
                             <1>  ;;dd  offset ejec ;;lpr, AX = 9
0000600D [44600000]          <1>  dd    ccvt ; tty0, AX = 10 (runix)
00006011 [44600000]          <1>  dd    ccvt ; tty1, AX = 11 (runix)
00006015 [44600000]          <1>  dd    ccvt ; tty2, AX = 12 (runix)
00006019 [44600000]          <1>  dd    ccvt ; tty3, AX = 13 (runix)
0000601D [44600000]          <1>  dd    ccvt ; tty4, AX = 14 (runix)
00006021 [44600000]          <1>  dd    ccvt ; tty5, AX = 15 (runix)
00006025 [44600000]          <1>  dd    ccvt ; tty6, AX = 16 (runix)
00006029 [44600000]          <1>  dd    ccvt ; tty7, AX = 17 (runix)
0000602D [44600000]          <1>  dd    ccvt ; COM1, AX = 18 (runix only)
00006031 [44600000]          <1>  dd    ccvt ; COM2, AX = 19 (runix only)
                             <1>
                             <1>  ; 1:
                             <1>  ;        ctty   / tty
                             <1>  ;        cppt   / ppt
                             <1>  ;        sret   / mem
                             <1>  ;        sret   / rf0
                             <1>  ;        sret   / rk0
                             <1>  ;        sret   / tap0
                             <1>  ;        sret   / tap1
                             <1>  ;        sret   / tap2
                             <1>  ;        sret   / tap3
                             <1>  ;        sret   / tap4
                             <1>  ;        sret   / tap5
                             <1>  ;        sret   / tap6
                             <1>  ;        sret   / tap7
                             <1>  ;        ccvt   / tty0
                             <1>  ;        ccvt   / tty1
                             <1>  ;        ccvt   / tty2
                             <1>  ;        ccvt   / tty3
                             <1>  ;        ccvt   / tty4
                             <1>  ;        ccvt   / tty5
                             <1>  ;        ccvt   / tty6
                             <1>  ;        ccvt   / tty7
                             <1>  ;        error / crd
                             <1>
                             <1> ;iclose_2: ; 2: / negative i-number
                             <1>  ;neg  ax
                             <1>        ;neg r1 / make it positive
                             <1>  ;cmp  ax, 40
                             <1>        ;cmp r1,$40. / is it a special file?
                             <1>         ;ja  short @b
                             <1>        ;bgt    3b / no. return
                             <1>  ;push ax
                             <1>        ;mov r1,-(sp)
                             <1>  ;movzx      ebx, al
                             <1>  ;shl  bx, 1
                             <1>        ;asl r1 / yes. compute jump address and transfer
                             <1>  ;add  ebx, iclose_3 - 2
                             <1>  ;jmp  dword [ebx]
                             <1>        ;jmp *1f-2(r1) / figure out
                             <1> ;iclose_3:
                             <1>  ;dd   ctty ; tty, AX = 1 (runix)
                             <1>  ;dd   sret ; mem, AX = 2 (runix)
                             <1>  ;dd   sret ; fd0, AX = 3 (runix only)
                             <1>  ;dd   sret ; fd1, AX = 4 (runix only)
                             <1>  ;dd   sret ; hd0, AX = 5 (runix only)
                             <1>  ;dd   sret ; hd1, AX = 6 (runix only)
                             <1>  ;dd   sret ; hd2, AX = 7 (runix only)
                             <1>  ;dd   sret ; hd3, AX = 8 (runix only)
                             <1>   ;dd  sret ; lpr, AX = 9
                             <1>  ;dd   ejec ; lpr, AX = 9  (runix)
                             <1>  ;dd   ccvt ; tty0, AX = 10 (runix)
                             <1>  ;dd   ccvt ; tty1, AX = 11 (runix)
                             <1>  ;dd   ccvt ; tty2, AX = 12 (runix)
                             <1>  ;dd   ccvt ; tty3, AX = 13 (runix)
                             <1>  ;dd   ccvt ; tty4, AX = 14 (runix)
                             <1>  ;dd   ccvt ; tty5, AX = 15 (runix)
                             <1>  ;dd   ccvt ; tty6, AX = 16 (runix)
                             <1>  ;dd   ccvt ; tty7, AX = 17 (runix)
                             <1>  ;dd   ccvt ; COM1, AX = 18 (runix only)
                             <1>  ;dd   ccvt ; COM2, AX = 19 (runix only)
                             <1>
                             <1>  ;1:
                             <1>  ;           ctty   / tty
                             <1>  ;       leadr  / ppt
                             <1>  ;       sret   / mem
                             <1>  ;       sret   / rf0
                             <1>  ;       sret   / rk0
                             <1>  ;       sret   / tap0
                             <1>  ;       sret   / tap1
                             <1>  ;       sret   / tap2
                             <1>  ;       sret   / tap3
                             <1>  ;       sret   / tap4
                             <1>  ;       sret   / tap5
                             <1>  ;       sret   / tap6
                             <1>  ;       sret   / tap7
                             <1>  ;       ccvt   / tty0
                             <1>  ;       ccvt   / tty1
                             <1>  ;       ccvt   / tty2
                             <1>  ;       ccvt   / tty3
                             <1>  ;       ccvt   / tty4
                             <1>  ;       ccvt   / tty5
                             <1>  ;       ccvt   / tty6
                             <1>  ;       ccvt   / tty7
                             <1>  ;/       ejec / lpr
                             <1>
                             <1> ctty: ; / close console tty
                             <1>  ; 18/05/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 21/05/2013 - 26/01/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification !
                             <1>  ; (DL = 2 -> it is open for reading)
                             <1>  ; (DL = 1 -> it is open for writing)
                             <1>  ; (DL = 0 -> it is open for sysstty system call)
                             <1>  ;
                             <1>  ; 06/12/2013
00006035 0FB61D[C9740000]    <1>         movzx   ebx, byte [u.uno] ; process number
0000603C 8A83[D5710000]      <1>         mov     al, [ebx+p.ttyc-1]
                             <1>  ; 13/01/2014
00006042 EB02                <1>  jmp   short cttyp
                             <1> ccvt:
00006044 2C0A                <1>  sub   al, 10
                             <1> cttyp:
                             <1>  ; 18/05/2015 (32 bit modifications)
                             <1>  ; 16/08/2013 - 26/01/2014
00006046 0FB6D8              <1>  movzx       ebx, al ; tty number (0 to 9)
00006049 D0E3                <1>  shl   bl, 1  ; aligned to word
                             <1>  ; 26/01/2014
0000604B 81C3[F4700000]      <1>  add   ebx, ttyl
00006051 88C6                <1>  mov   dh, al ; tty number
00006053 668B03              <1>  mov   ax, [ebx]
                             <1>            ; AL = lock value (0 or process number)
                             <1>            ; AH = open count
00006056 20E4                <1>  and   ah, ah
00006058 750F                <1>  jnz   short ctty_ret
0000605A C705[CF740000]0A00- <1>         mov     dword [u.error], ERR_DEV_NOT_OPEN
00006062 0000                <1>
                             <1>               ; device not open ! error
                             <1>  ;jmp  short ctty_err ; open count = 0, it is not open !
00006064 E913E0FFFF          <1>  jmp   error
                             <1>  ; 26/01/2014
                             <1> ctty_ret:
00006069 FECC                <1>  dec   ah ; decrease open count
0000606B 7502                <1>  jnz   short ctty_1
0000606D 30C0                <1>  xor   al, al ; unlock/free tty
                             <1> ctty_1:
0000606F 668903              <1>  mov   [ebx], ax ; close tty instance
                             <1>  ;
00006072 BB[B0740000]        <1>  mov   ebx, u.ttyp
00006077 F6C201              <1>  test  dl, 1 ; open for write sign
0000607A 7401                <1>  jz    short ctty_2
0000607C 43                  <1>  inc   ebx
                             <1> ctty_2:
0000607D FEC6                <1>  inc   dh ; tty number + 1
0000607F 3A33                <1>  cmp   dh, [ebx]
00006081 7503                <1>  jne   short cret
                             <1>  ; Reset/Clear 'u.ttyp' ('the recent TTY') value
00006083 C60300              <1>  mov   byte [ebx], 0
                             <1> cret:
00006086 08D2                <1>  or    dl, dl ; sysstty system call check (DL=0)
00006088 7402                <1>  jz    short ctty_3
0000608A 6658                <1>  pop   ax
                             <1> ctty_3:
0000608C C3                  <1>  retn
                             <1>
                             <1> ;ctty_err: ; 13/01/2014
                             <1> ;or    dl, dl ; DL = 0 -> called by sysstty
                             <1> ;jnz   error
                             <1> ;stc
                             <1> ;retn
                             <1>
                             <1>
                             <1>  ; Original UNIX v1 'ctty' routine:
                             <1>  ;
                             <1>         ;mov    tty+[ntty*8]-8+6,r5
                             <1>  ;           ;/ point r5 to the console tty buffer
                             <1>         ;decb   (r5) / dec number of processes using console tty
                             <1>         ;br     sret / return via sret
                             <1>
                             <1> ;ccvt: ; < close tty >
                             <1>  ; 21/05/2013 - 13/01/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification !
                             <1>  ;
                             <1>  ; In original UNIX v1, 'ccvt' routine
                             <1>  ;           (exactly different than this one)
                             <1>  ;     was in 'u9.s' file.
                             <1>  ;
                             <1>  ; DL = 2 -> it is open for reading
                             <1>  ; DL = 1 -> it is open for writing
                             <1>  ;
                             <1>  ; 17/09/2013
```

```
<1>  ;sub  al, 10
<1>  ;cmp  al, 7
<1>  ;jna  short cttyp
<1>  ; 13/01/2014
<1>  ;jmp  short cttyp
<1>
<1> ;cppt: / close paper tape
<1> ;        clrb   pptiflg / set pptiflg to indicate file not open
<1> ;1:
<1> ;        mov    $240,*$ps /set process or priority to 5
<1> ;        jsr    r0,getc; 2 / remove all ppt input entries from clist
<1> ;                          / and assign to free list
<1> ;               br sret
<1> ;        br     1b
<1>
<1> ;ejec:
<1> ; jmp   error
<1> ;/ejec:
<1> ;/       mov    $100,*$lps / set line printer interrupt enable bit
<1> ;/       mov    $14,r1 / 'form feed' character in r1 (new page).
<1> ;/       jsr    r0,lptoc / space the printer to a new page
<1> ;/       br     sret / return to caller via 'sret'
    %include 'u8.s'        ; 11/06/2015
<1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS8.INC
<1> ; Last Modification: 24/10/2015
<1> ; ----------------------------------------------------------------------------
<1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
<1> ; (v0.1 - Beginning: 11/07/2012)
<1> ;
<1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
<1> ; (Original) Source Code by Ken Thompson (1971-1972)
<1> ; <Bell Laboratories (17/3/1972)>
<1> ; <Preliminary Release of UNIX Implementation Document>
<1> ;
<1> ; Retro UNIX 8086 v1 - U8.ASM (18/01/2014) //// UNIX v1 -> u8.s
<1> ;
<1> ; ****************************************************************************
<1>
<1> ;; I/O Buffer - Retro UNIX 386 v1 modification
<1> ;;     (8+512 bytes, 8 bytes header, 512 bytes data)
<1> ;; Word 1, byte 0 = device id
<1> ;; Word 1, byte 1 = status bits (bits 8 to 15)
<1> ;;         bit 9 = write bit
<1> ;;        bit 10 = read bit
<1> ;;        bit 12 = waiting to write bit
<1> ;;        bit 13 = waiting to read bit
<1> ;;        bit 15 = inhibit bit
<1> ;; Word 2 (byte 2 & byte 3) = reserved (for now - 07/06/2015)
<1> ;; Word 3 + Word 4 (byte 4,5,6,7) = physical block number
<1> ;;             (In fact, it is 32 bit LBA for Retro UNIX 386 v1)
<1> ;;
<1> ;; I/O Buffer ((8+512 bytes in original Unix v1))
<1> ;;          ((4+512 bytes in Retro UNIX 8086 v1))
<1> ;;
<1> ;; I/O Queue Entry (of original UNIX operating system v1)
<1> ;; Word 1, Byte 0 = device id
<1> ;; Word 1, Byte 1 = (bits 8 to 15)
<1> ;;         bit 9 = write bit
<1> ;;        bit 10 = read bit
<1> ;;        bit 12 = waiting to write bit
<1> ;;        bit 13 = waiting to read bit
<1> ;;        bit 15 = inhibit bit
<1> ;; Word 2 = physical block number (In fact, it is LBA for Retro UNIX 8086 v1)
<1> ;;
<1> ;; Original UNIX v1 ->
<1> ;;          Word 3 = number of words in buffer (=256)
<1> ;; Original UNIX v1 ->
<1> ;;          Word 4 = bus address (addr of first word of data buffer)
<1> ;;
<1> ;; Retro UNIX 8086 v1 -> Buffer Header (I/O Queue Entry) size is 4 bytes !
<1> ;;
<1> ;; Device IDs (of Retro Unix 8086 v1)
<1> ;;         0 = fd0
<1> ;;        1 = fd1
<1> ;;        2 = hd0
<1> ;;        3 = hd1
<1> ;;        4 = hd2
<1> ;;        5 = hd3
<1>
<1> ; Retro UNIX 386 v1 - 32 bit modifications (rfd, wfd, rhd, whd) - 09/06/2015
<1>
<1> rfd:   ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
<1>  ; 26/04/2013
<1>        ; 13/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)

<1>        ;sub  ax, 3 ; zero based device number (Floppy disk)
<1>        ;jmp  short bread ; **** returns to routine that called readi

<1>
<1> rhd:   ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
<1>  ; 26/04/2013
```

```
                             <1>        ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)
                             <1>        ;sub  ax, 3 ; zero based device number (Hard disk)
                             <1>        ;jmp  short bread ; **** returns to routine that called readi
                             <1>
                             <1> bread:
                             <1>  ; 14/07/2015
                             <1>  ; 10/07/2015
                             <1>  ; 09/06/2015
                             <1>  ; 07/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 13/03/2013 - 29/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; / read a block from a block structured device
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    [u.fopf] points to the block number
                             <1>  ;    CX = maximum block number allowed on device
                             <1>  ;      ; that was an arg to bread, in original Unix v1, but
                             <1>  ;      ; CX register is used instead of arg in Retro Unix 8086 v1
                             <1>  ;    [u.count]    number of bytes to read in
                             <1>  ; OUTPUTS ->
                             <1>  ;    [u.base] starting address of data block or blocks in user area
                             <1>  ;    [u.fopf] points to next consecutive block to be read
                             <1>  ;
                             <1>  ; ((Modified registers: eAX, eDX, eCX, eBX, eSI, eDI, eBP))
                             <1>  ;
                             <1>  ; NOTE: Original UNIX v1 has/had a defect/bug here, even if read
                             <1>  ;       byte count is less than 512, block number in *u.fofp (u.off)
                             <1>  ;     is increased by 1. For example: If user/program request
                             <1>  ;       to read 16 bytes in current block, 'sys read' increases
                             <1>  ;     the next block number just as 512 byte reading is done.
                             <1>  ;       This wrong is done in 'bread'. So, in Retro UNIX 8086 v1,
                             <1>  ;       for user (u) structure compatibility (because 16 bit is not
                             <1>  ;       enough to keep byte position/offset of the disk), this
                             <1>  ;     defect will not be corrected, user/program must request
                             <1>  ;     512 byte read per every 'sys read' call to block devices
                             <1>  ;       for achieving correct result. In future version(s),
                             <1>  ;     this defect will be corrected by using different
                             <1>  ;       user (u) structure.  26/07/2013 - Erdogan Tan
                             <1>
                             <1>        ; jsr r0,tstdeve / error on special file I/O
                             <1>                     ; / (only works on tape)
                             <1>        ; mov *u.fofp,r1 / move block number to r1
                             <1>        ; mov $2.-cold,-(sp) / "2-cold" to stack
                             <1> ;1:
                             <1>        ; cmp r1,(r0) / is this block # greater than or equal to
                             <1>                  ; / maximum block # allowed on device
                             <1>        ; jnb short @f
                             <1>        ; bhis      1f / yes, 1f (error)
                             <1>        ; mov r1,-(sp) / no, put block # on stack
                             <1>        ; jsr r0,preread / read in the block into an I/O buffer
                             <1>        ; mov (sp)+,r1 / return block # to r1
                             <1>        ; inc r1 / bump block # to next consecutive block
                             <1>        ; dec (sp) / "2-1-cold" on stack
                             <1>        ; bgt 1b / 2-1-cold = 0?  No, go back and read in next block
                             <1> ;1:
                             <1>        ; tst (sp)+ / yes, pop stack to clear off cold calculation
                             <1>  ;push ecx ; **
                             <1>  ;26/04/2013
                             <1>  ;sub  ax, 3 ; 3 to 8 -> 0 to 5
0000608D 2C03                <1>  sub   al, 3
                             <1>        ; AL = Retro Unix 8086 v1 disk (block device) number
0000608F A2[E2740000]        <1>  mov   [u.brwdev], al
                             <1>  ; 09/06/2015
00006094 0FB6D8              <1>  movzx ebx, al
00006097 8B8B[086E0000]      <1>  mov   ecx, [ebx+drv.size] ; disk size (in sectors)
                             <1> bread_0:
0000609D 51                  <1>  push  ecx ; ** ; 09/06/2015
                             <1>  ; 10/07/2015 (Retro UNIX 386 v1 modification!)
                             <1>  ; [u.fopf] points to byte position in disk, not sector/block !
0000609E 8B1D[90740000]      <1>  mov   ebx, [u.fofp]
000060A4 8B03                <1>  mov   eax, [ebx]
000060A6 C1E809              <1>  shr   eax, 9 ; convert byte position to block/sector number
                             <1>        ; mov *u.fofp,r1 / restore r1 to initial value of the
                             <1>                     ; / block #
000060A9 39C8                <1>  cmp   eax, ecx
                             <1>        ; cmp r1,(r0)+ / block # greater than or equal to maximum
                             <1>                             ; / block number allowed
                             <1>  ;jnb  error           ; 18/04/2013
                             <1>        ; bhis error10 / yes, error
000060AB 720F                <1>  jb    short bread_1
000060AD C705[CF740000]1000- <1>  mov   dword [u.error], ERR_DEV_VOL_SIZE  ; 'out of volume' error
000060B5 0000                <1>
000060B7 E9C0DFFFFF          <1>  jmp   error
                             <1> bread_1:
                             <1>  ; inc       dword [ebx] ; 10/07/2015 (Retro UNIX 386 v1 - modification!)
                             <1>        ; inc *u.fofp / no, *u.fofp has next block number
                             <1>  ; eAX = Block number (zero based)
                             <1>        ;;jsr r0,preread / read in the block whose number is in r1
                             <1> preread: ;; call preread
000060BC BF[E2740000]        <1>  mov   edi, u.brwdev ; block device number for direct I/O
000060C1 E864020000          <1>  call  bufaloc_0 ; 26/04/2013
```

```
                             <1>  ;; jc       error
                             <1>  ; eBX = Buffer (Header) Address -Physical-
                             <1>        ; eAX = Block/Sector number (r1)
                             <1>         ; jsr r0,bufaloc / get a free I/O buffer (r1 has block number)
                             <1>  ; 14/03/2013
000060C6 740A                <1>         jz   short bread_2 ; Retro UNIX 8086 v1 modification
                             <1>               ; br 1f / branch if block already in a I/O buffer
000060C8 66810B0004          <1>  or    word [ebx], 400h ; set read bit (10) in I/O Buffer
                             <1>               ; bis $2000,(r5) / set read bit (bit 10 in I/O buffer)
000060CD E8B3010000          <1>  call  poke
                             <1>               ; jsr r0,poke / perform the read
                             <1>  ;;jc  error ;2 0/07/2013
                             <1> ; 1:
                             <1>        ; clr *$ps / ps = 0
                             <1>               ; rts r0
                             <1> ;; return from preread
                             <1> bread_2:
000060D2 66810B0040          <1>  or    word [ebx], 4000h
                             <1>        ; bis $40000,(r5)
                             <1>               ; / set bit 14 of the 1st word of the I/O buffer
                             <1> bread_3: ; 1:
000060D7 66F7030024          <1>  test  word [ebx], 2400h
                             <1>        ; bit $22000,(r5) / are 10th and 13th bits set (read bits)
000060DC 7407                <1>  jz    short bread_4
                             <1>        ; beq 1f / no
                             <1>        ; cmp cdev,$1 / disk or drum?
                             <1>        ; ble 2f / yes
                             <1>        ; tstb uquant / is the time quantum = 0?
                             <1>        ; bne 2f / no, 2f
                             <1>        ; mov r5,-(sp) / yes, save r5 (buffer address)
                             <1>        ; jsr r0,sleep; 31.
                             <1>               ; / put process to sleep in channel 31 (tape)
                             <1>        ; mov (sp)+,r5 / restore r5
                             <1>        ; br 1b / go back
                             <1> ; 2: / drum or disk
                             <1>         ;; mov     cx, [s.wait_]+2 ;; 29/07/2013
000060DE E8AAF3FFFF          <1>  call  idle
                             <1>        ; jsr r0,idle; s.wait+2 / wait
000060E3 EBF2                <1>  jmp   short bread_3
                             <1>               ; br 1b
                             <1> bread_4: ; 1: / 10th and 13th bits not set
000060E5 668123FFBF          <1>  and   word [ebx], 0BFFFh ; 1011111111111111b
                             <1>        ; bic $40000,(r5) / clear bit 14
                             <1>               ; jsr r0,tstdeve / test device for error (tape)
000060EA 83C308              <1>  add   ebx, 8
                             <1>        ; add $8,r5 / r5 points to data in I/O buffer
                             <1>  ; 09/06/2015
000060ED 66833D[DF740000]00  <1>  cmp   word [u.pcount], 0
000060F5 7705                <1>  ja    short bread_5
000060F7 E896F9FFFF          <1>  call  trans_addr_w ; translate virtual address to physical (w)
                             <1> bread_5:
                             <1>  ; eBX = system (I/O) buffer address
000060FC E870000000          <1>  call  dioreg
                             <1>               ; jsr r0,dioreg / do bookkeeping on u.count etc.
                             <1>  ; esi =  start address of the transfer (in the buffer)
                             <1>  ; edi =  [u.pbase], destination address in user's memory space
                             <1>  ; ecx =  transfer count (in bytes)
                             <1>  ;
                             <1> ;1: / r5 points to beginning of data in I/O buffer, r2 points to beginning
                             <1> ;   / of users data
00006101 F3A4                <1>  rep   movsb
                             <1>        ; movb (r5)+,(r2)+ / move data from the I/O buffer
                             <1>               ; dec r3 / to the user's area in core starting at u.base
                             <1>               ; bne 1b
00006103 59                  <1>  pop   ecx ; **
00006104 833D[A4740000]00    <1>  cmp   dword [u.count], 0
                             <1>        ; tst u.count / done
0000610B 7790                <1>  ja    short bread_0 ; 09/06/2015
                             <1>               ; beq 1f / yes, return
                             <1>        ; tst -(r0) / no, point r0 to the argument again
                             <1>               ; br bread / read some more
                             <1> ; 1:
0000610D 58                  <1>  pop   eax ; ****
                             <1>               ; mov (sp)+,r0
0000610E C3                  <1>         retn       ; 09/06/2015
                             <1>  ;jmp     ret_
                             <1>        ;jmp ret  / jump to routine that called readi
                             <1>
                             <1> wfd:    ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 26/04/2013
                             <1>        ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)

                             <1>        ;sub  ax, 3 ; zero based device number (Hard disk)
                             <1>        ;jmp  short bwrite ; **** returns to routine that called writei
                             <1>
                             <1> whd:   ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>        ; 14/03/2013 Retro UNIX 8086 v1 device (not an original unix v1 device)

                             <1>        ;sub  ax, 3 ; zero based device number (Hard disk)
                             <1>        ;jmp  short bwrite ; **** returns to routine that called writei ('jmp ret')
                             <1>
                             <1> bwrite:
```

```
                             <1>  ; 14/07/2015
                             <1>  ; 10/07/2015
                             <1>  ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 14/03/2013 - 20/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ;; / write on block structured device
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    [u.fopf] points to the block number
                             <1>  ;    CX = maximum block number allowed on device
                             <1>  ;      ; that was an arg to bwrite, in original Unix v1, but
                             <1>  ;      ; CX register is used instead of arg in Retro Unix 8086 v1
                             <1>  ;    [u.count]    number of bytes to user desires to write
                             <1>  ; OUTPUTS ->
                             <1>  ;    [u.fopf] points to next consecutive block to be written into
                             <1>  ;
                             <1>  ; ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
                             <1>  ;
                             <1>  ; NOTE: Original UNIX v1 has/had a defect/bug here, even if write
                             <1>  ;       byte count is less than 512, block number in *u.fofp (u.off)
                             <1>  ;     is increased by 1. For example: If user/program request
                             <1>  ;       to write 16 bytes in current block, 'sys write' increases
                             <1>  ;     the next block number just as 512 byte writing is done.
                             <1>  ;       This wrong is done in 'bwrite'. So, in Retro UNIX 8086 v1,
                             <1>  ;       for user (u) structure compatibility (because 16 bit is not
                             <1>  ;       enough to keep byte position/offset of the disk), this
                             <1>  ;     defect will not be corrected, user/program must request
                             <1>  ;     512 byte write per every 'sys write' call to block devices
                             <1>  ;       for achieving correct result. In future version(s),
                             <1>  ;     this defect will be corrected by using different
                             <1>  ;       user (u) structure.  26/07/2013 - Erdogan Tan
                             <1>
                             <1>              ; jsr r0,tstdeve / test the device for an error
                             <1>  ;push ecx ; **
                             <1>  ;26/04/2013
                             <1>  ;sub  ax, 3 ; 3 to 8 -> 0 to 5
0000610F 2C03                <1>  sub   al, 3
                             <1>        ; AL = Retro Unix 8086 v1 disk (block device) number
00006111 A2[E2740000]        <1>  mov   [u.brwdev], al
                             <1>  ; 09/06/2015
00006116 0FB6D8              <1>  movzx ebx, al
00006119 8B8B[086E0000]      <1>  mov   ecx, [ebx+drv.size] ; disk size (in sectors)
                             <1> bwrite_0:
0000611F 51                  <1>  push  ecx ; ** ; 09/06/2015
                             <1>  ; 10/07/2015 (Retro UNIX 386 v1 modification!)
                             <1>  ; [u.fopf] points to byte position in disk, not sector/block !
00006120 8B1D[90740000]      <1>  mov   ebx, [u.fofp]
00006126 8B03                <1>  mov   eax, [ebx]
00006128 C1E809              <1>  shr   eax, 9 ; convert byte position to block/sector number
                             <1>        ; mov *u.fofp,r1 / put the block number in r1
0000612B 39C8                <1>  cmp   eax, ecx
                             <1>        ; cmp r1,(r0)+ / does block number exceed maximum allowable #
                             <1>                               ; / block number allowed
                             <1>  ;jnb  error      ; 18/04/2013
                             <1>        ; bhis error10 / yes, error
0000612D 720F                <1>        jb    short bwrite_1
0000612F C705[CF740000]1000- <1>  mov   dword [u.error], ERR_DEV_VOL_SIZE  ; 'out of volume' error
00006137 0000                <1>
00006139 E93EDFFFFF          <1>  jmp   error
                             <1> bwrite_1:
                             <1>  ; inc       dword [ebx] ; 10/07/2015 (Retro UNIX 386 v1 - modification!)
                             <1>        ; inc *u.fofp / no, increment block number
                             <1>  ; 09/06/2015 - 10/07/2015
0000613E 66833D[DF740000]00  <1>  cmp   word [u.pcount], 0
00006146 7705                <1>  ja    short bwrite_2
00006148 E841F9FFFF          <1>  call  trans_addr_r ; translate virtual address to physical (r)
                             <1> bwrite_2:
0000614D BF[E2740000]        <1>  mov   edi, u.brwdev  ; block device number for direct I/O
00006152 E8C4000000          <1>        call  bwslot ; 26/04/2013 (wslot -> bwslot)
                             <1>        ; jsr r0,wslot / get an I/O buffer to write into
                             <1>        ; add $8,r5 / r5 points to data in I/O buffer
00006157 E815000000          <1>         call dioreg
                             <1>        ; jsr r0,dioreg / do the necessary bookkeeping
                             <1>  ; esi =  destination address (in the buffer)
                             <1>  ; edi =  [u.pbase], start address of transfer in user's memory space
                             <1>  ; ecx =  transfer count (in bytes)
                             <1> ; 1: / r2 points to the users data; r5 points to the I/O buffers data area
0000615C 87F7                <1>  xchg  esi, edi ; 14/07/2015
0000615E F3A4                <1>  rep   movsb
                             <1>        ; movb (r2)+,(r5)+ / ; r3, has the byte count
                             <1>              ; dec r3 / area to the I/O buffer
                             <1>              ; bne 1b
00006160 E8FE000000          <1>  call  dskwr
                             <1>        ; jsr r0,dskwr / write it out on the device
00006165 59                  <1>  pop   ecx ; **
00006166 833D[A4740000]00    <1>         cmp     dword [u.count], 0
                             <1>        ; tst u.count / done
0000616D 77B0                <1>  ja    short bwrite_0 ; 09/06/2015
                             <1>        ; beq 1f / yes, 1f
                             <1>        ; tst -(r0) / no, point r0 to the argument of the call
                             <1>              ; br bwrite / go back and write next block
                             <1> ; 1:
```

```
0000616F 58                 <1>  pop   eax ; ****
                            <1>               ; mov (sp)+,r0
00006170 C3                 <1>  retn        ; 09/06/2015
                            <1>         ;jmp     ret_
                            <1>        ; jmp ret / return to routine that called writei
                            <1> ;error10:
                            <1> ;       jmp     error  ; / see 'error' routine
                            <1>
                            <1> dioreg:
                            <1>  ; 14/07/2015
                            <1>  ; 10/07/2015 (UNIX v1 bugfix - [u.fofp]: byte pos., not block)
                            <1>  ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
                            <1>  ; 14/03/2013 (Retro UNIX 8086 v1)
                            <1>  ;
                            <1>  ; bookkeeping on block transfers of data
                            <1>  ;
                            <1>  ; * returns value of u.pbase before it gets updated, in EDI
                            <1>  ; * returns byte count (to transfer) in ECX (<=512)
                            <1>  ; 10/07/2015
                            <1>  ; * returns byte offset from beginning of current sector buffer
                            <1>  ; (beginning of data) in ESI
                            <1>  ;
00006171 8B0D[A4740000]     <1>  mov   ecx, [u.count]
                            <1>        ; mov u.count,r3 / move char count to r3
00006177 81F900020000       <1>        cmp   ecx, 512
                            <1>        ; cmp r3,$512. / more than 512. char?
0000617D 7605               <1>  jna   short dioreg_0
                            <1>        ; blos 1f / no, branch
0000617F B900020000         <1>  mov   ecx, 512
                            <1>        ; mov $512.,r3 / yes, just take 512.
                            <1> dioreg_0:
                            <1>  ; 09/06/2015
00006184 663B0D[DF740000]   <1>  cmp   cx, [u.pcount]
0000618B 7607               <1>  jna   short dioreg_1
0000618D 668B0D[DF740000]   <1>  mov   cx, [u.pcount]
                            <1> dioreg_1:
                            <1> ; 1:
00006194 8B15[A0740000]     <1>  mov   edx, [u.base] ; 09/06/2015 (eax -> edx)
                            <1>          ; mov u.base,r2 / put users base in r2
0000619A 010D[A8740000]     <1>  add   [u.nread], ecx
                            <1>        ; add r3,u.nread / add the number to be read to u.nread
000061A0 290D[A4740000]     <1>  sub   [u.count], ecx
                            <1>        ; sub r3,u.count / update count
000061A6 010D[A0740000]     <1>  add   [u.base], ecx
                            <1>        ; add r3,u.base / update base
                            <1>  ; 10/07/2015
                            <1>  ; Retro UNIX 386 v1 - modification !
                            <1>  ; (File pointer points to byte position, not block/sector no.)
                            <1>  ; (It will point to next byte position instead of next block no.)
000061AC 8B35[90740000]     <1>  mov   esi, [u.fofp] ; u.fopf points to byte position pointer
000061B2 8B06               <1>  mov   eax, [esi] ; esi points to current byte pos. on the disk
000061B4 010E               <1>  add   [esi], ecx ; ecx is added to set the next byte position
000061B6 25FF010000         <1>  and   eax, 1FFh  ; get offset from beginning of current block
000061BB 89DE               <1>  mov   esi, ebx   ; beginning of data in sector/block buffer
000061BD 01C6               <1>  add   esi, eax   ; esi contains start address of the transfer
                            <1>  ; 09/06/2015 - 10/07/2015
000061BF 66290D[DF740000]   <1>  sub   [u.pcount], cx
000061C6 81E2FF0F0000       <1>  and   edx, PAGE_OFF ; 0FFFh
000061CC 8B3D[DB740000]     <1>  mov   edi, [u.pbase]
000061D2 81E700F0FFFF       <1>  and   edi, ~PAGE_OFF
000061D8 01D7               <1>  add   edi, edx
000061DA 893D[DB740000]     <1>  mov   [u.pbase], edi
000061E0 010D[DB740000]     <1>  add   [u.pbase], ecx ; 14/07/2015
000061E6 C3                 <1>  retn
                            <1>        ; rts r0 / return
                            <1>
                            <1> dskrd:
                            <1>  ; 18/08/2015
                            <1>  ; 02/07/2015
                            <1>  ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
                            <1>  ; 14/03/2013 - 29/07/2013 (Retro UNIX 8086 v1)
                            <1>  ;
                            <1>  ; 'dskrd' acquires an I/O buffer, puts in the proper
                            <1>  ; I/O queue entries (via bufaloc) then reads a block
                            <1>  ; (number specified in r1) in the acquired buffer.)
                            <1>  ; If the device is busy at the time dskrd is called,
                            <1>  ; dskrd calls idle.
                            <1>  ;
                            <1>  ; INPUTS ->
                            <1>  ;    r1 - block number
                            <1>  ;    cdev - current device number
                            <1>  ; OUTPUTS ->
                            <1>  ;    r5 - points to first data word in I/O buffer
                            <1>  ;
                            <1>  ; ((AX = R1)) input/output
                            <1>  ; ((BX = R5)) output
                            <1>  ;
                            <1>         ; ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
                            <1>  ;
000061E7 E830010000         <1>  call  bufaloc
                            <1>        ; jsr r0,bufaloc / shuffle off to bufaloc;
                            <1>                      ; / get a free I/O buffer
```

```
                             <1>  ;;jc  error ; 20/07/2013
000061EC 741B                <1>  jz    short dskrd_1 ; Retro UNIX 8086 v1 modification
                             <1>              ; br 1f / branch if block already in a I/O buffer
                             <1> dskrd_0: ; 10/07/2015 (wslot)
000061EE 66810B0004          <1>  or    word [ebx], 400h ; set read bit (10) in I/O Buffer
                             <1>              ; bis $2000,(r5) / set bit 10 of word 1 of
                             <1>                       ; / I/O queue entry for buffer
000061F3 E88D000000          <1>  call  poke
                             <1>        ; jsr r0,poke / just assigned in bufaloc,
                             <1>                  ; /     bit 10=1 says read
                             <1>  ; 09/06/2015
000061F8 730F                <1>  jnc   short dskrd_1
                             <1>  ;
000061FA C705[CF740000]1100- <1>  mov   dword [u.error], ERR_DRV_READ ; disk read error !
00006202 0000                <1>
00006204 E973DEFFFF          <1>  jmp   error
                             <1> dskrd_1: ; 1:
                             <1>              ;clr *$ps
00006209 66F7030024          <1>        test  word [ebx], 2400h
                             <1>        ; bit $22000,(r5) / if either bits 10, or 13 are 1;
                             <1>                    ; / jump to idle
0000620E 7407                <1>        jz    short dskrd_2
                             <1>        ; beq 1f
                             <1>         ;;mov   ecx, [s.wait_]
00006210 E878F2FFFF          <1>        call  idle
                             <1>        ; jsr r0,idle; s.wait+2
00006215 EBF2                <1>  jmp   short dskrd_1
                             <1>              ; br 1b
                             <1> dskrd_2: ; 1:
00006217 83C308              <1>         add  ebx, 8
                             <1>        ; add $8,r5 / r5 points to first word of data in block
                             <1>                ; / just read in
0000621A C3                  <1>        retn
                             <1>        ; rts r0
                             <1>
                             <1> bwslot:
                             <1>  ; 10/07/2015
                             <1>  ;     If the block/sector is not placed in a buffer
                             <1>  ;     before 'wslot', it must be read before
                             <1>  ;     it is written! (Otherwise transfer counts less
                             <1>  ;     than 512 bytes will be able to destroy existing
                             <1>  ;     data on disk.)
                             <1>  ;
                             <1>  ; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 26/04/2013(Retro UNIX 8086 v1)
                             <1>  ; Retro UNIX 8086 v1 modification !
                             <1>  ; ('bwslot' will be called from 'bwrite' only!)
                             <1>  ; INPUT -> eDI - points to device id (in u.brwdev)
                             <1>  ;     -> eAX = block number
                             <1>  ;
0000621B E80A010000          <1>  call  bufaloc_0
00006220 742A                <1>  jz    short wslot_0 ; block/sector already is in the buffer
                             <1> bwslot_0:
                             <1>  ; 10/07/2015
00006222 8B35[90740000]      <1>  mov   esi, [u.fofp]
00006228 8B06                <1>  mov   eax, [esi]
0000622A 25FF010000          <1>  and   eax, 1FFh ; offset from beginning of the sector/block
0000622F 750C                <1>  jnz   short bwslot_1 ; it is not a full sector write
                             <1>               ; recent disk data must be placed in the buffer
00006231 813D[A4740000]0002- <1>  cmp   dword [u.count], 512
00006239 0000                <1>
0000623B 730F                <1>  jnb   short wslot_0
                             <1> bwslot_1:
0000623D E8ACFFFFFF          <1>  call  dskrd_0
00006242 83EB08              <1>  sub   ebx, 8 ; set ebx to the buffer header address again
00006245 EB05                <1>  jmp   short wslot_0
                             <1>
                             <1> wslot:
                             <1>  ; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ;           (32 bit modifications)
                             <1>  ; 14/03/2013 - 29/07/2013(Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; 'wslot' calls 'bufaloc' and obtains as a result, a pointer
                             <1>  ; to the I/O queue of an I/O buffer for a block structured
                             <1>  ; device. It then checks the first word of I/O queue entry.
                             <1>  ; If bits 10 and/or 13 (read bit, waiting to read bit) are set,
                             <1>  ; wslot calls 'idle'. When 'idle' returns, or if bits 10
                             <1>  ; and/or 13 are not set, 'wslot' sets bits 9 and 15 of the first
                             <1>  ; word of the I/O queue entry (write bit, inhibit bit).
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - block number
                             <1>  ;    cdev - current (block/disk) device number
                             <1>  ;
                             <1>  ; OUTPUTS ->
                             <1>  ;    bufp - bits 9 and 15 are set,
                             <1>  ;           the remainder of the word left unchanged
                             <1>  ;    r5 - points to first data word in I/O buffer
                             <1>  ;
                             <1>  ; ((AX = R1)) input/output
                             <1>  ; ((BX = R5)) output
                             <1>  ;
```

```
                              <1>         ; ((Modified registers: eDX, eCX, eBX, eSI, eDI, eBP))
                              <1>
00006247 E8D0000000           <1>  call  bufaloc
                              <1>  ; 10/07/2015
                              <1>        ; jsr r0,bufaloc / get a free I/O buffer; pointer to first
                              <1>              ; br 1f / word in buffer in r5
                              <1>  ; eBX = Buffer (Header) Address (r5) (ES=CS=DS, system/kernel segment)
                              <1>         ; eAX = Block/Sector number (r1)
                              <1> wslot_0: ;1:
0000624C 66F7030024           <1>        test  word [ebx], 2400h
                              <1>        ; bit $22000,(r5) / check bits 10, 13 (read, waiting to read)
                              <1>                    ; / of I/O queue entry
00006251 7407                 <1>  jz    short wslot_1
                              <1>                  ; beq 1f  / branch if 10, 13 zero (i.e., not reading,
                              <1>             ; / or not waiting to read)
                              <1>
                              <1>         ;; mov     ecx, [s.wait_] ; 29/07/2013
00006253 E835F2FFFF           <1>  call  idle
                              <1>        ; jsr r0,idle; / if buffer is reading or writing to read,
                              <1>                          ; / idle
00006258 EBF2                 <1>  jmp   short wslot_0
                              <1>        ; br 1b / till finished
                              <1> wslot_1: ;1:
0000625A 66810B0082           <1>         or      word [ebx], 8200h
                              <1>             ; bis $101000,(r5) / set bits 9, 15 in 1st word of I/O queue
                              <1>                                ; / (write, inhibit bits)
                              <1>             ; clr     *$ps / clear processor status
0000625F 83C308               <1>         add  ebx, 8 ; 11/06/2015
                              <1>        ; add $8,r5 / r5 points to first word in data area
                              <1>                 ; / for this block
00006262 C3                   <1>        retn
                              <1>        ; rts r0
                              <1> dskwr:
                              <1>  ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 14/03/2013 - 03/08/2013 (Retro UNIX 8086 v1)
                              <1>  ;
                              <1>  ; 'dskwr' writes a block out on disk, via ppoke. The only
                              <1>  ; thing dskwr does is clear bit 15 in the first word of I/O queue
                              <1>  ; entry pointed by 'bufp'. 'wslot' which must have been called
                              <1>  ; previously has supplied all the information required in the
                              <1>  ; I/O queue entry.
                              <1>  ;
                              <1>  ; (Modified registers: eCX, eDX, eBX, eSI, eDI)
                              <1>  ;
                              <1>  ;
00006263 8B1D[4A740000]       <1>  mov   ebx, [bufp]
00006269 668123FF7F           <1>  and   word [ebx], 7FFFh ; 0111111111111111b
                              <1>        ; bic $100000,*bufp / clear bit 15 of I/O queue entry at
                              <1>                                   ; / bottom of queue
0000626E E812000000           <1>  call  poke
                              <1>  ; 09/06/2015
00006273 730F                 <1>  jnc   short dskwr_1
00006275 C705[CF740000]1200-  <1>  mov   dword [u.error], ERR_DRV_WRITE ; disk write error !
0000627D 0000                 <1>
0000627F E9F8DDFFFF           <1>  jmp   error
                              <1> dskwr_1:
00006284 C3                   <1>  retn
                              <1>
                              <1>
                              <1> ;ppoke:
                              <1>             ; mov $340,*$ps
                              <1>             ; jsr r0,poke
                              <1>             ; clr *$ps
                              <1>        ; rts r0
                              <1> poke:
                              <1>  ; 24/10/2015
                              <1>  ; 20/08/2015
                              <1>  ; 18/08/2015
                              <1>  ; 02/07/2015
                              <1>  ; 09/06/2015 (Retro UNIX 386 v1 - Beginning)
                              <1>  ; 15/03/2013 - 18/01/2014 (Retro UNIX 8086 v1)
                              <1>  ;
                              <1>  ; (NOTE: There are some disk I/O code modifications & extensions
                              <1>  ; & exclusions on original 'poke' & other device I/O procedures of
                              <1>  ; UNIX v1 OS for performing disk I/O functions by using IBM PC
                              <1>  ; compatible rombios calls in Retro UNIX 8086 v1 kernel.)
                              <1>  ;
                              <1>  ; Basic I/O functions for all block structured devices
                              <1>  ;
                              <1>         ; (Modified registers: eCX, eDX, eSI, eDI)
                              <1>  ;
                              <1>  ; 20/07/2013 modifications
                              <1>  ;             (Retro UNIX 8086 v1 features only !)
                              <1>  ; INPUTS ->
                              <1>  ;        (EBX = buffer header address)
                              <1>  ; OUTPUTS ->
                              <1>  ;       cf=0 -> successed r/w (at least, for the caller's buffer)
                              <1>  ;       cf=1 -> error, word [eBX] = 0FFFFh
                              <1>  ;            (drive not ready or r/w error!)
                              <1>  ;       (dword [EBX+4] <> 0FFFFFFFFh indicates r/w success)
                              <1>  ;       (dword [EBx+4] = 0FFFFFFFFh means RW/IO error)
                              <1>  ;         (also it indicates invalid buffer data)
```

```
                             <1>  ;
00006285 53                  <1>  push  ebx
                             <1>              ; mov r1,-(sp)
                             <1>              ; mov r2,-(sp)
                             <1>              ; mov r3,-(sp)
00006286 50                  <1>  push  eax ; Physical Block Number (r1) (mget)
                             <1>  ;
                             <1>  ; 09/06/2015
                             <1>  ; (permit read/write after a disk  R/W error)
00006287 8A0B                <1>  mov   cl, [ebx] ; device id (0 to 5)
00006289 B001                <1>  mov   al, 1
0000628B D2E0                <1>  shl   al, cl
0000628D 8405[6A740000]      <1>  test  al, [active] ; busy ? (error)
00006293 7408                <1>  jz    short poke_0
00006295 F6D0                <1>  not   al
00006297 2005[6A740000]      <1>  and   [active], al ; reset busy bit for this device only
                             <1> poke_0:
0000629D BE[62740000]        <1>         mov     esi, bufp + (4*(nbuf+2))
                             <1>        ; mov $bufp+nbuf+nbuf+6,r2 / r2 points to highest priority
                             <1>                           ; / I/O queue pointer
                             <1> poke_1: ; 1:
000062A2 83EE04              <1>         sub  esi, 4
000062A5 8B1E                <1>  mov   ebx, [esi]
                             <1>        ; mov -(r2),r1 / r1 points to an I/O queue entry
000062A7 668B03              <1>  mov   ax, [ebx] ; 17/07/2013
000062AA F6C406              <1>        test  ah, 06h
                             <1>  ;test word [ebx], 600h ; 0000011000000000b
                             <1>        ; bit $3000,(r1) / test bits 9 and 10 of word 1 of I/O
                             <1>                     ; / queue entry
000062AD 745E                <1>         jz      short poke_5
                             <1>        ; beq 2f / branch to 2f if both are clear
                             <1>  ; 31/07/2013
                             <1>  ;test ah, 0B0h ; (*)
                             <1>  ;;test      word [ebx], 0B000h ; 1011000000000000b
                             <1>        ; bit $130000,(r1) / test bits 12, 13, and 15
                             <1>         ;jnz     short poke_5 ; 31/07/2013 (*)
                             <1>        ; bne 2f / branch if any are set
                             <1>  ;movzx      ecx, byte [ebx] ; 09/06/2015 ; Device Id
                             <1>              ; movb (r1),r3 / get device id
000062AF 0FB6C8              <1>  movzx ecx, al ; 18/08/2015
                             <1>  ;mov  edi, ecx ; 26/04/2013
000062B2 31C0                <1>  xor   eax, eax ; 0
                             <1>  ;cmp  [edi+drv.error], al ; 0
                             <1>        ; tstb deverr(r3) / test for errors on this device
                             <1>        ;jna  short poke_2
                             <1>        ; beq 3f / branch if no errors
                             <1>  ; 02/07/2015
                             <1>  ;dec  eax
                             <1>  ;mov  [ebx+4], ax ; 0FFFFFFFFh ; -1
                             <1>              ; mov $-1,2(r1) / destroy associativity
                             <1>  ;shr  eax, 24
                             <1>  ;mov  [ebx], eax ; 000000FFh, reset
                             <1>        ; clrb 1(r1) / do not do I/O
                             <1>  ;jmp     short poke_5
                             <1>         ;       ; br 2f
                             <1>                 ; rts r0
                             <1> poke_2: ; 3:
                             <1>  ; 02/07/2015
000062B4 FEC1                <1>  inc   cl ; 0FFh -> 0
000062B6 7455                <1>  jz    short poke_5
000062B8 FEC0                <1>  inc   al ; mov ax, 1
000062BA FEC9                <1>  dec   cl
000062BC 7402                <1>  jz    short poke_3
                             <1>  ; 26/04/2013 Modification
                             <1>  ;inc  al ; mov ax, 1
                             <1>  ;or   cl, cl ; Retro UNIX 8086 v1 device id.
                             <1>  ;jz   short poke_3 ; cl = 0
000062BE D2E0                <1>  shl   al, cl ; shl ax, cl
                             <1> poke_3:
                             <1>  ;test [active], ax
000062C0 8405[6A740000]      <1>  test  [active], al
                             <1>        ; bit $2,active / test disk busy bit
000062C6 7545                <1>  jnz     short poke_5
                             <1>        ; bne 2f / branch if bit is set
                             <1>  ;or   [active], ax
000062C8 0805[6A740000]      <1>  or    [active], al
                             <1>        ; bis $2,active / set disk busy bit
000062CE 6650                <1>  push  ax
000062D0 E8CB000000          <1>  call  diskio ; Retro UNIX 8086 v1 Only !
                             <1>  ;mov    [edi+drv.error], ah
000062D5 6658                <1>  pop   ax
000062D7 730E                <1>  jnc   short poke_4 ; 20/07/2013
                             <1>  ;cmp  [edi+drv.error], al ; 0
                             <1>  ;jna  short poke_4
                             <1>        ; tstb deverr(r3) / test for errors on this device
                             <1>              ; beq 3f / branch if no errors
                             <1>  ; 02/07/2015 (32 bit modification)
                             <1>  ; 20/07/2013
000062D9 C74304FFFFFFFF      <1>  mov   dword [ebx+4], 0FFFFFFFFh ; -1
                             <1>              ; mov $-1,2(r1) / destroy associativity
000062E0 66C703FF00          <1>  mov   word [ebx], 0FFh ; 20/08/2015
                             <1>        ; clrb 1(r1) / do not do I/O
```

```
000062E5 EB26                <1>  jmp     short poke_5
                             <1> poke_4:; 20/07/2013
                             <1>  ; 17/07/2013
000062E7 F6D0                <1>  not   al
000062E9 2005[6A740000]      <1>  and   [active], al ; reset, not busy
                             <1>  ; eBX = system I/O buffer header (queue entry) address
                             <1> seta: ; / I/O queue bookkeeping; set read/write waiting bits.
000062EF 668B03              <1>  mov   ax, [ebx]
                             <1>              ; mov (r1),r3 / move word 1 of I/O queue entry into r3
000062F2 66250006            <1>         and  ax, 600h
                             <1>        ; bic $!3000,r3 / clear all bits except 9 and 10
000062F6 668123FFF9          <1>  and   word [ebx], 0F9FFh
                             <1>              ; bic $3000,(r1) / clear only bits 9 and 10
000062FB C0E403              <1>  shl   ah, 3
                             <1>              ; rol r3
                             <1>                 ; rol r3
                             <1>                 ; rol r3
000062FE 660903              <1>  or    [ebx], ax
                             <1>        ; bis r3,(r1) / or old value of bits 9 and 10 with
                             <1>                 ; bits 12 and 13
00006301 E887F1FFFF          <1>  call  idle ; 18/01/2014
                             <1>  ;; sti
                             <1>  ;hlt  ; wait for a hardware interrupt
                             <1>  ;; cli
                             <1>  ; NOTE: In fact, disk controller's 'disk I/O completed'
                             <1>         ; interrupt would be used to reset busy bits, but INT 13h
                             <1>  ; returns when disk I/O is completed. So, here, as temporary
                             <1>  ; method, this procedure will wait for a time according to
                             <1>  ; multi tasking and time sharing concept.
                             <1>  ;
                             <1>  ; 24/10/2015
                             <1>  ;not  ax
00006306 66B8FF00            <1>  mov   ax, 0FFh ; 24/10/2015 (temporary)
0000630A 662103              <1>  and   [ebx], ax ; clear bits 12 and 13
                             <1> poke_5: ;2:
0000630D 81FE[4A740000]      <1>         cmp     esi, bufp
                             <1>                ; cmp r2,$bufp / test to see if entire I/O queue
                             <1>                             ; / has been scanned
00006313 778D                <1>  ja      short poke_1
                             <1>                ; bhi 1b
                             <1>  ; 24/03/2013
                             <1>              ; mov (sp)+,r3
                             <1>              ; mov (sp)+,r2
                             <1>              ; mov (sp)+,r1
00006315 58                  <1>         pop  eax  ; Physical Block Number (r1) (mget)
00006316 5B                  <1>  pop   ebx
                             <1>  ; 02/07/2015 (32 bit modification)
                             <1>  ; 20/07/2013
                             <1>  ;cmp  dword [ebx+4], 0FFFFFFFFh
00006317 803BFF              <1>  cmp   byte [ebx], 0FFh ; 20/08/2015
                             <1>  ;
                             <1>  ; 'poke' returns with cf=0 if the requested buffer is read
                             <1>  ; or written succesfully; even if an error occurs while
                             <1>  ; reading to or writing from other buffers. 20/07/2013
                             <1>  ;
                             <1>  ; 09/06/2015
0000631A F5                  <1>  cmc
0000631B C3                  <1>  retn
                             <1>                 ; rts r0
                             <1>
                             <1> bufaloc:
                             <1>  ; 20/08/2015
                             <1>  ; 19/08/2015
                             <1>  ; 02/07/2015
                             <1>  ; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ;           (32 bit modifications)
                             <1>  ; 13/03/2013 - 29/07/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; bufaloc - Block device I/O buffer allocation
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;    r1 - block number
                             <1>  ;    cdev - current (block/disk) device number
                             <1>  ;    bufp+(2*n)-2 --- n = 1 ... nbuff
                             <1>  ; OUTPUTS ->
                             <1>  ;    r5 - pointer to buffer allocated
                             <1>  ;    bufp ... bufp+12 --- (bufp), (bufp)+2
                             <1>  ;
                             <1>  ; ((AX = R1)) input/output
                             <1>  ; ((BX = R5)) output
                             <1>         ;    ((Modified registers: DX, CX, BX, SI, DI, BP))
                             <1>  ;    zf=1 -> block already in a I/O buffer
                             <1>  ;    zf=0 -> a new I/O buffer has been allocated
                             <1>  ;    ((DL = Device ID))
                             <1>  ;    (((DH = 0 or 1)))
                             <1>  ;    (((CX = previous value of word ptr [bufp])))
                             <1>  ;    ((CX and DH will not be used after return)))
                             <1>
                             <1>  ;;push      esi ; ***
                             <1>        ; mov r2,-(sp) / save r2 on stack
                             <1>              ; mov $340,*$ps / set processor priority to 7
                             <1>  ; 20/07/2013
                             <1>  ; 26/04/2013
0000631C 0FB61D[66740000]    <1>  movzx ebx, byte [cdev] ; 0 or 1
00006323 BF[68740000]        <1>  mov   edi, rdev  ; offset mdev = offset rdev + 1
00006328 01DF                <1>  add   edi, ebx
                             <1> bufaloc_0: ; 26/04/2013 !! here is called from bread or bwrite !!
                             <1>              ;; eDI points to device id.
0000632A 0FB61F              <1>  movzx ebx, byte [edi] ; [EDI] -> rdev/mdev or brwdev
                             <1>  ; 11/06/20215
0000632D 80BB[246E0000]F0    <1>  cmp   byte [ebx+drv.status], 0F0h ; Drive not ready !
00006334 720F                <1>  jb    short bufaloc_9
00006336 C705[CF740000]0F00- <1>  mov   dword [u.error], ERR_DRV_NOT_RDY
0000633E 0000                <1>
00006340 E937DDFFFF          <1>  jmp   error
                             <1> bufaloc_9:
00006345 89DA                <1>  mov   edx, ebx ; dh = 0, dl = device number (0 to 5)
                             <1> bufaloc_10: ; 02/07/2015
00006347 31ED                <1>  xor   ebp, ebp ; 0
00006349 55                  <1>  push  ebp ; 0
0000634A 89E5                <1>         mov     ebp, esp
                             <1>  ;
                             <1> bufaloc_1: ;1:
                             <1>        ; clr -(sp) / vacant buffer
0000634C BE[4A740000]        <1>         mov  esi, bufp
                             <1>        ; mov $bufp,r2 / bufp contains pointers to I/O queue
                             <1>                   ; / entrys in buffer area
                             <1> bufaloc_2: ;2:
00006351 8B1E                <1>  mov   ebx, [esi]
                             <1>              ; mov (r2)+,r5 / move pointer to word 1 of an I/O
                             <1>                  ; queue entry into r5
00006353 66F70300F6          <1>  test  word [ebx], 0F600h
                             <1>        ; bit $173000,(r5) / lock+keep+active+outstanding
00006358 7503                <1>         jnz  short bufaloc_3
                             <1>        ; bne 3f / branch when
                             <1>               ; / any of bits 9,10,12,13,14,15 are set
                             <1>                        ; / (i.e., buffer busy)
0000635A 897500              <1>         mov     [ebp], esi ; pointer to I/0 queue entry
                             <1>                 ; mov  r2,(sp) ;/ save pointer to last non-busy buffer
                             <1>              ; / found points to word 2 of I/O queue entry)
                             <1> bufaloc_3: ;3:
                             <1>  ;mov  dl, [edi] ; 26/04/2013
                             <1>  ;
0000635D 3813                <1>  cmp   [ebx], dl
                             <1>        ; cmpb (r5),cdev / is device in I/O queue entry same
                             <1>                     ; / as current device
0000635F 7508                <1>  jne   short bufaloc_4
                             <1>              ; bne 3f
00006361 394304              <1>  cmp   [ebx+4], eax
                             <1>              ; cmp 2(r5),r1 / is block number in I/O queue entry,
                             <1>                   ; / same as current block number
00006364 7503                <1>        jne   short bufaloc_4
                             <1>        ; bne 3f
                             <1>  ;add  esp, 4
00006366 59                  <1>  pop   ecx
                             <1>              ; tst (sp)+ / bump stack pointer
00006367 EB20                <1>  jmp   short bufaloc_7 ; Retro Unix 8086 v1 modification
                             <1>                      ; jump to bufaloc_6 in original Unix v1
                             <1>              ; br 1f / use this buffer
                             <1> bufaloc_4: ;3:
00006369 83C604              <1>  add   esi, 4 ; 20/08/2015
                             <1>  ;
0000636C 81FE[5A740000]      <1>  cmp   esi, bufp + (nbuf*4)
                             <1>        ; cmp r2,$bufp+nbuf+nbuf
00006372 72DD                <1>  jb    short bufaloc_2
                             <1>        ; blo 2b / go to 2b if r2 less than bufp+nbuf+nbuf (all
                             <1>                        ; / buffers not checked)
00006374 5E                  <1>         pop  esi
                             <1>        ; mov (sp)+,r2 / once all bufs are examined move pointer
                             <1>                   ; / to last free block
00006375 09F6                <1>        or    esi, esi
00006377 7507                <1>  jnz   short bufaloc_5
                             <1>        ; bne 2f / if (sp) is non zero, i.e.,
                             <1>          ; / if a free buffer is found branch to 2f
                             <1>         ;; mov  ecx, [s.wait_]
00006379 E80FF1FFFF          <1>  call  idle
                             <1>        ; jsr r0,idle; s.wait+2 / idle if no free buffers
0000637E EBC7                <1>  jmp   short bufaloc_10 ; 02/07/2015
                             <1>              ; br 1b
                             <1> bufaloc_5: ;2:
                             <1>        ; tst (r0)+ / skip if warmed over buffer
00006380 FEC6                <1>  inc   dh ; Retro UNIX 8086 v1 modification
                             <1> bufaloc_6: ;1:
00006382 8B1E                <1>         mov        ebx, [esi]
                             <1>        ; mov -(r2),r5 / put pointer to word 1 of I/O queue
                             <1>                   ; / entry in r5
                             <1>  ;; 26/04/2013
                             <1>         ;mov dl, [edi] ; byte [rdev] or byte [mdev]
00006384 8813                <1>  mov   [ebx], dl
                             <1>        ; movb cdev,(r5) / put current device number
                             <1>                     ; / in I/O queue entry
00006386 894304              <1>  mov   [ebx+4], eax
                             <1>        ; mov r1,2(r5) / move block number into word 2
                             <1>                   ; / of I/O queue entry
                             <1> bufaloc_7: ;1:
00006389 81FE[4A740000]      <1>         cmp  esi, bufp
                             <1>        ; cmp r2,$bufp / bump all entrys in bufp
                             <1>                   ; / and put latest assigned
0000638F 760A                <1>  jna   short bufaloc_8
                             <1>             ; blos 1f / buffer on the top
                             <1>             ; / (this makes if the lowest priority)
00006391 83EE04              <1>  sub   esi, 4
00006394 8B0E                <1>  mov   ecx, [esi]
00006396 894E04              <1>  mov   [esi+4], ecx
                             <1>        ; mov -(r2),2(r2) / job for a particular device
00006399 EBEE                <1>  jmp   short bufaloc_7
                             <1>        ; br 1b
                             <1> bufaloc_8: ;1:
0000639B 891E                <1>         mov  [esi], ebx
                             <1>        ; mov r5,(r2)
                             <1>  ;;pop esi ; ***
                             <1>             ; mov (sp)+,r2 / restore r2
0000639D 08F6                <1>        or    dh, dh ; 0 or 1 ?
                             <1>        ; Retro UNIX 8086 v1 modification
                             <1>        ; zf=1 --> block already is in an I/O buffer
                             <1>        ; zf=0 --> a new I/O buffer has been allocated
0000639F C3                  <1>  retn
                             <1>        ; rts r0
                             <1>
                             <1> diskio:
                             <1>  ; 10/07/2015
                             <1>  ; 02/07/2015
                             <1>  ; 16/06/2015
                             <1>  ; 11/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ;          (80386 protected mode modifications)
                             <1>  ; 15/03/2013 - 29/04/2013 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 feature only !
                             <1>  ;
                             <1>  ; Derived from proc_chs_read procedure of TRDOS DISKIO.ASM (2011)
                             <1>  ; 04/07/2009 - 20/07/2011
                             <1>  ;
                             <1>  ; NOTE: Reads only 1 block/sector (sector/block size is 512 bytes)
                             <1>  ;
                             <1>         ; INPUTS ->
                             <1>  ;        eBX = System I/O Buffer header address
                             <1>  ;
                             <1>         ; OUTPUTS -> cf=0 --> done
                             <1>  ;          cf=1 ---> error code in AH
                             <1>  ;
                             <1>  ; (Modified registers: eAX, eCX, eDX)
                             <1>
                             <1> ;rw_disk_sector:
                             <1>  ; 10/07/2015
                             <1>  ; 02/07/2015
                             <1>  ; 11/06/2015 - Retro UNIX 386 v1 - 'u8.s'
                             <1>  ; 21/02/2015 ('dsectpm.s', 'read_disk_sector')
                             <1>  ; 16/02/2015 (Retro UNIX 386 v1 test - 'unix386.s')
                             <1>  ; 01/12/2014 - 18/01/2015 ('dsectrm2.s')
                             <1>  ;
                             <1>  ;mov  dx, 0201h ; Read 1 sector/block
000063A0 B602                <1>  mov   dh, 2
000063A2 668B03              <1>  mov   ax, [ebx]
                             <1>  ;
000063A5 56                  <1>  push  esi ; ****
000063A6 53                  <1>  push  ebx ; ***
                             <1>  ;
000063A7 0FB6C8              <1>  movzx ecx, al
000063AA 89CE                <1>  mov   esi, ecx
                             <1>  ;
000063AC 38F1                <1>  cmp   cl, dh ; 2
000063AE 7202                <1>  jb    short rwdsk0
000063B0 047E                <1>  add   al, 7Eh  ; 80h, 81h, 82h, 83h
                             <1> rwdsk0:
000063B2 A2[D56D0000]        <1>  mov   [drv], al
000063B7 81C6[246E0000]      <1>  add   esi, drv.status
                             <1>  ; 11/06/2015
000063BD 803EF0              <1>  cmp   byte [esi], 0F0h
000063C0 720F                <1>  jb      short rwdsk1
                             <1>  ; 'drive not ready' error
000063C2 C705[CF740000]0F00- <1>  mov   dword [u.error], ERR_DRV_NOT_RDY
000063CA 0000                <1>
000063CC E9ABDCFFFF          <1>  jmp   error
                             <1> rwdsk1:
000063D1 F6C402              <1>  test  ah, 2
                             <1>  ;test ax, 200h ; Bit 9 of word 0 (status word)
                             <1>                 ; write bit
000063D4 7402                <1>  jz    short rwdsk2
                             <1>  ;test ah, 4
                             <1>  ;;test      ax, 400h ; Bit 10 of word 0 (status word)
                             <1>  ;             ; read bit
                             <1>  ;jz   short diskio_ret
000063D6 FEC6                <1>  inc   dh ; 03h = write
                             <1> rwdsk2:
000063D8 88C2                <1>  mov   dl, al
000063DA 83C304              <1>  add   ebx, 4 ; sector/block address/number pointer
```

```
000063DD 8B03                <1>  mov   eax, [ebx] ; sector/block number (LBA)
000063DF C0E102              <1>  shl   cl, 2
000063E2 81C1[086E0000]      <1>  add   ecx, drv.size ; disk size
000063E8 3B01                <1>  cmp   eax, [ecx] ; Last sector + 1 (number of secs.)
000063EA 720F                <1>  jb      short rwdsk3
                             <1>  ; 'out of volume' error
000063EC C705[CF740000]1000- <1>  mov   dword [u.error], ERR_DEV_VOL_SIZE
000063F4 0000                <1>
000063F6 E981DCFFFF          <1>  jmp   error
                             <1> rwdsk3:
                             <1>  ; 11/06/2015
000063FB 83C304              <1>  add   ebx, 4 ; buffer address
000063FE C605[FE740000]04    <1>  mov   byte [retry_count], 4
00006405 F60601              <1>  test  byte [esi], 1 ; LBA ready ?
00006408 7432                <1>         jz      short rwdsk_chs
                             <1> rwdsk_lba:
                             <1>  ; LBA read/write (with private LBA function)
                             <1>  ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
0000640A 83C607              <1>        add     esi, drv.error - drv.status ; 10/07/2015
0000640D 89C1                <1>  mov   ecx, eax ; sector number
                             <1>  ; ebx = buffer (data) address
                             <1>  ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
                             <1> rwdsk_lba_retry:
                             <1>  ;mov  dl, [drv]
                             <1>         ; Function 1Bh = LBA read, 1Ch = LBA write
0000640F B419                <1>  mov   ah, 1Ch - 3h ; LBA write function number - 3
00006411 00F4                <1>  add   ah, dh
00006413 B001                <1>  mov   al, 1
                             <1>  ;int  13h
00006415 E8E9C3FFFF          <1>  call  int13h
0000641A 8826                <1>  mov   [esi], ah ; error code ; 10/07/2015
0000641C 730E                <1>  jnc   short rwdsk_lba_ok
0000641E 80FC80              <1>  cmp   ah, 80h ; time out ?
00006421 7408                <1>        je      short rwdsk_lba_fails
00006423 FE0D[FE740000]      <1>  dec   byte [retry_count]
00006429 7504                <1>        jnz     short rwdsk_lba_reset ; 10/07/2015
                             <1> rwdsk_lba_fails:
0000642B F9                  <1>  stc
                             <1> rwdsk_lba_ok:
0000642C 5B                  <1>  pop   ebx ; ***
0000642D 5E                  <1>  pop   esi ; ****
0000642E C3                  <1>  retn
                             <1> rwdsk_lba_reset:
0000642F B40D                <1>  mov   ah, 0Dh ; Alternate reset
                             <1>  ;int  13h
00006431 E8CDC3FFFF          <1>        call int13h
00006436 73D7                <1>  jnc     short rwdsk_lba_retry
00006438 8826                <1>  mov   [esi], ah ; error code ; 10/07/2015
0000643A EBF0                <1>  jmp   short rwdsk_lba_ok
                             <1>  ;
                             <1>  ; CHS read (convert LBA address to CHS values)
                             <1> rwdsk_chs:
                             <1>  ; 10/07/2015
0000643C 81EE[246E0000]      <1>  sub   esi, drv.status
00006442 89F1                <1>  mov   ecx, esi
00006444 81C6[2B6E0000]      <1>  add   esi, drv.error
                             <1>  ; 02/07/2015
                             <1>  ; 16/06/2015
                             <1>  ; 11/06/2015
0000644A 53                  <1>  push  ebx ; ** ; buffer
0000644B D1E1                <1>  shl   ecx, 1
0000644D 51                  <1>  push  ecx ; *
                             <1>  ;
0000644E 89CB                <1>  mov   ebx, ecx
00006450 8835[FD740000]      <1>  mov   [rwdsk], dh ; 02/07/2015
00006456 31D2                <1>  xor   edx, edx ; 0
00006458 29C9                <1>  sub   ecx, ecx
0000645A 81C3[FA6D0000]      <1>        add     ebx, drv.spt
00006460 668B0B              <1>  mov   cx, [ebx] ; sector per track
                             <1>         ; EDX:EAX = LBA
00006463 F7F1                <1>  div   ecx
00006465 88D1                <1>  mov   cl, dl      ; sector number - 1
00006467 FEC1                <1>  inc   cl    ; sector number (1 based)
00006469 5B                  <1>  pop   ebx ; * ; 11/06/2015
0000646A 6651                <1>  push  cx
0000646C 81C3[EC6D0000]      <1>        add     ebx, drv.heads
00006472 668B0B              <1>  mov   cx, [ebx] ; heads
00006475 31D2                <1>  xor   edx, edx
                             <1>         ; EAX = cylinders * heads + head
00006477 F7F1                <1>  div   ecx
00006479 6659                <1>  pop   cx     ; sector number
0000647B 88D6                <1>  mov   dh, dl ; head number
0000647D 8A15[D56D0000]      <1>  mov   dl, [drv]
00006483 88C5                <1>  mov   ch, al ; cylinder (bits 0-7)
00006485 C0E406              <1>  shl   ah, 6
00006488 08E1                <1>  or    cl, ah ; cylinder (bits 8-9)
                             <1>                 ; sector (bits 0-7)
0000648A 5B                  <1>  pop   ebx ; ** ; buffer ; 11/06/2015
                             <1>         ; CL = sector (bits 0-5)
                             <1>         ;      cylinder (bits 8-9 -> bits 6-7)
                             <1>         ; CH = cylinder (bits 0-7)
                             <1>         ; DH = head
                             <1>        ; DL = drive
                             <1>  ;
0000648B C605[FE740000]04    <1>  mov   byte [retry_count], 4
                             <1> rwdsk_retry:
00006492 8A25[FD740000]      <1>  mov   ah, [rwdsk] ; 02h = read, 03h = write
00006498 B001                <1>  mov   al, 1 ; sector count
                             <1>  ;int  13h
0000649A E864C3FFFF          <1>  call  int13h
0000649F 8826                <1>  mov   [esi], ah ; error code ; 10/07/2015
000064A1 730E                <1>  jnc   short rwdsk_ok ; ah = 0
000064A3 80FC80              <1>  cmp   ah, 80h ; time out ?
000064A6 7408                <1>  je    short rwdsk_fails
000064A8 FE0D[FE740000]      <1>  dec   byte [retry_count]
000064AE 7504                <1>  jnz   short rwdsk_reset
                             <1> rwdsk_fails:
000064B0 F9                  <1>  stc
                             <1> rwdsk_ok:
000064B1 5B                  <1>  pop   ebx ; ***
000064B2 5E                  <1>  pop   esi ; ****
000064B3 C3                  <1>  retn
                             <1> rwdsk_reset:
                             <1>  ; 02/02/2015
000064B4 28E4                <1>  sub   ah, ah
000064B6 80FA80              <1>  cmp   dl, 80h
000064B9 7202                <1>  jb    short rwdsk_fd_reset
000064BB B40D                <1>  mov   ah, 0Dh ; Alternate reset
                             <1> rwdsk_fd_reset:
                             <1>  ;int  13h
000064BD E841C3FFFF          <1>         call int13h
000064C2 73CE                <1>  jnc   short rwdsk_retry
000064C4 8826                <1>  mov   [esi], ah ; error code ; 10/07/2015
000064C6 EBE9                <1>  jmp   short rwdsk_ok
                             <1>
                             <1>
                             <1> ; Original UNIX v1 - drum (& disk) interrupt routine
                             <1> ;(Equivalent to IRQ 14 & IRQ 15 disk/hardware interrupts)
                             <1> ;
                             <1> ; This feature is not used in Retro UNIX 386 (& 8086) for now.
                             <1> ; Because, current Retro UNIX 386 disk I/O -INT13H- routine is
                             <1> ; derived from IBM PC AT -infact: XT286- BIOS source code, int 13h
                             <1> ; that uses hardware -transfer has been completed-  interrupt inside it.
                             <1> ; In a next Retro UNIX 386 version, these interrupts
                             <1> ; (fdc_int, hdc1_int, hdc2_int) will be handled by a separate routine
                             <1> ; as in original unix v1.
                             <1> ; I am not removing IBM BIOS source code derivatives -compatible code-
                             <1> ; for now, regarding the new/next 32 bit TRDOS project by me
                             <1> ; (to keep source code files easy adaptable to 32 bit TRDOS.)
                             <1> ;
                             <1> ; Erdogan tan (10/07/2015)
                             <1>
                             <1> ;drum: / interrupt handler
                             <1> ;       jsr     r0,setisp / save r1,r2,r3, and clockp on the stack
                             <1> ;       jsr     r0,trapt; dcs; rfap; 1 / check for stray interrupt or
                             <1> ;                                      / error
                             <1> ;               br 3f / no, error
                             <1> ;       br      2f / error
                             <1> ;
                             <1> ;disk:
                             <1> ;       jsr     r0,setisp / save r1,r2,r3, and clockp on the stack
                             <1> ;       jmp     *$0f
                             <1> ;0:
                             <1> ;       jsr     r0,trapt; rkcs; rkap; 2
                             <1> ;              br 3f / no, errors
                             <1> ;       mov     $115,(r2) / drive reset, errbit was set
                             <1> ;       mov     $1f,0b-2 / next time jmp *$0f is executed jmp will be
                             <1> ;                         / to 1f
                             <1> ;       br      4f
                             <1> ;1:
                             <1> ;       bit     $20000,rkcs
                             <1> ;       beq     4f / wait for seek complete
                             <1> ;       mov     $0b,0b-2
                             <1> ;       mov     rkap,r1
                             <1> ;2:
                             <1> ;       bit     $3000,(r1) / are bits 9 or 10 set in the 1st word of
                             <1> ;                          / the disk buffer
                             <1> ;       bne     3f / no, branch ignore error if outstanding
                             <1> ;       inc     r1
                             <1> ;       asr     (r1)
                             <1> ;       asr     (r1)
                             <1> ;       asr     (r1) / reissue request
                             <1> ;       dec     r1
                             <1> ;3:
                             <1> ;       bic     $30000,(r1) / clear bits 12 and 13 in 1st word of buffer
                             <1> ;       mov     ac,-(sp)
                             <1> ;       mov     mq,-(sp) / put these on the stack
                             <1> ;       mov     sc,-(sp)
                             <1> ;       jsr     r0,poke
                             <1> ;       mov     (sp)+,sc
                             <1> ;       mov     (sp)+,mq / pop them off stack
                             <1> ;       mov     (sp)+,ac
                             <1> ;4:
                             <1> ;       jmp     retisp / u4-3
                             <1> ;
                             <1> ;trapt:                 / r2 points to the
                             <1> ;       mov     (r0)+,r2 / device control register
                             <1> ;       mov     *(r0)+,r1 / transaction pointer points to buffer
                             <1> ;       tst     (sp)+
                             <1> ;       tstb    (r2) / is ready bit of dcs set?
                             <1> ;       bge     4b / device still active so branch
                             <1> ;       bit     (r0),active / was device busy?
                             <1> ;       beq     4b / no, stray interrupt
                             <1> ;       bic     (r0)+,active / yes, set active to zero
                             <1> ;       tst     (r2) / test the err(bit is) of dcs
                             <1> ;       bge     2f / if no error jump to 2f
                             <1> ;       tst     (r0)+ / skip on error
                             <1> ; 2:
                             <1> ;       jmp     (r0)
                                 %include 'u9.s'        ; 29/06/2015
                             <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS9.INC
                             <1> ; Last Modification: 09/12/2015
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from 'Retro UNIX 8086 v1' source code by Erdogan Tan
                             <1> ; (v0.1 - Beginning: 11/07/2012)
                             <1> ;
                             <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                             <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                             <1> ; <Bell Laboratories (17/3/1972)>
                             <1> ; <Preliminary Release of UNIX Implementation Document>
                             <1> ;
                             <1> ; Retro UNIX 8086 v1 - U9.ASM (01/09/2014) //// UNIX v1 -> u9.s
                             <1> ;
                             <1> ; ****************************************************************************
                             <1>
                             <1> getch:
                             <1>  ; 30/06/2015
                             <1>  ; 18/02/2015 - Retro UNIX 386 v1 - feature only!
000064C8 28C0                <1>  sub   al, al ; 0
                             <1> getch_q: ; 06/08/2015
000064CA 8A25[D6700000]      <1>  mov   ah, [ptty] ; active (current) video page
000064D0 EB06                <1>        jmp     short getc_n
                             <1>
                             <1> getc:
                             <1>  ; 12/11/2015
                             <1>  ; 15/09/2015
                             <1>  ; 01/07/2015
                             <1>  ; 30/06/2015
                             <1>  ; 18/02/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 13/05/2013 - 04/07/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification !
                             <1>  ;
                             <1>  ; 'getc' gets (next) character
                             <1>  ;      from requested TTY (keyboard) buffer
                             <1>  ; INPUTS ->
                             <1>  ;     [u.ttyn] = tty number (0 to 7) (8 is COM1, 9 is COM2)
                             <1>  ;     AL=0 -> Get (next) character from requested TTY buffer
                             <1>  ;     (Keyboard buffer will point to
                             <1>  ;                 next character at next call)
                             <1>  ;     AL=1 -> Test a key is available in requested TTY buffer
                             <1>  ;     (Keyboard buffer will point to
                             <1>  ;                 current character at next call)
                             <1>  ; OUTPUTS ->
                             <1>  ;     (If AL input is 1) ZF=1 -> 'empty buffer' (no chars)
                             <1>  ;                        ZF=0 -> AX has (current) character
                             <1>  ;      AL = ascii code
                             <1>  ;      AH = scan code   (AH = line status for COM1 or COM2)
                             <1>  ;                   (cf=1 -> error code/flags in AH)
                             <1>  ; Original UNIX V1 'getc':
                             <1>  ;          get a character off character list
                             <1>  ;
                             <1>  ; ((Modified registers: eAX, eBX, eCX, eDX, eSI, eDI))
                             <1>  ;
                             <1>  ; 30/06/20045 (32 bit modifications)
                             <1>  ; 16/07/2013
                             <1>  ; mov       [getctty], ah
                             <1>  ;
                             <1>
000064D2 8A25[CE740000]      <1>  mov   ah, [u.ttyn]      ; 28/07/2013
                             <1> getc_n:
                             <1>  ; 30/06/2015
000064D8 08E4                <1>  or    ah, ah
000064DA 740D                <1>  jz    short getc0
000064DC D0E4                <1>  shl   ah, 1
000064DE 0FB6DC              <1>  movzx ebx, ah
000064E1 81C3[D8700000]      <1>  add   ebx, ttychr
000064E7 EB05                <1>  jmp   short getc1
                             <1> getc0:
000064E9 BB[D8700000]        <1>  mov   ebx, ttychr
                             <1> getc1:
000064EE 668B0B              <1>  mov   cx, [ebx]   ; ascii & scan code
                             <1>                    ; (by kb_int)
000064F1 6609C9              <1>  or    cx, cx
000064F4 7508                <1>  jnz   short getc2
000064F6 20C0                <1>  and   al, al
```

```
000064F8 7416               <1>  jz    short getc_s
000064FA 6631C0             <1>  xor   ax, ax
000064FD C3                 <1>  retn
                            <1> getc2:
000064FE 20C0               <1>  and   al, al
00006500 6689C8             <1>  mov   ax, cx
00006503 66B90000           <1>  mov   cx, 0
00006507 7506               <1>  jnz   short getc3
                            <1> getc_sn:
00006509 66890B             <1>  mov   [ebx], cx ; 0, reset
0000650C 6639C8             <1>  cmp   ax, cx  ; zf = 0
                            <1> getc3:
0000650F C3                 <1>  retn
                            <1> getc_s:
                            <1>  ; 12/11/2015
                            <1>  ; 15/09/2015
                            <1>  ; 01/07/2015
                            <1>  ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
                            <1>  ; 16/07/2013 - 14/02/2014 (Retro UNIX 8086 v1)
                            <1>  ;
                            <1>  ; tty  of the current process is not
                            <1>  ; current tty (ptty); so, current process only
                            <1>  ; can use keyboard input when its tty becomes
                            <1>  ; current tty (ptty).
                            <1>  ; 'sleep' is for preventing an endless lock
                            <1>  ; during this tty input request.
                            <1>  ; (Because, the user is not looking at the video page
                            <1>  ; of the process to undersand there is a keyboard
                            <1>  ; input request.)
                            <1>  ;
                            <1>  ;((Modified registers: eAX, eBX, eCX, eDX, eSI, eDI))
                            <1>  ;
                            <1>  ; 05/10/2013
                            <1>  ; ah = byte ptr [u.ttyn] ; (tty number)
                            <1>  ;
                            <1>  ; 10/10/2013
                            <1> gcw0:
00006510 B10A               <1>  mov   cl, 10 ; ch = 0
                            <1> gcw1:
                            <1>  ; 12/11/2015
00006512 E867DCFFFF         <1>  call intract ; jumps to 'sysexit' if [u.quit] = FFFFh
                            <1>  ; 10/10/2013
00006517 E871EFFFFF         <1>  call  idle
0000651C 668B03             <1>  mov   ax, [ebx]   ; ascii & scan code
                            <1>                    ; (by kb_int)
0000651F 6609C0             <1>  or    ax, ax
                            <1> ;jnz   short gcw3
00006522 7519               <1>  jnz   short gcw2 ; 15/09/2015
                            <1>  ; 30/06/2015
00006524 FEC9               <1>  dec   cl
00006526 75EA               <1>  jnz   short gcw1
                            <1>  ;
00006528 8A25[CE740000]     <1>  mov   ah, [u.ttyn]      ; 20/10/2013
                            <1> ;; 10/12/2013
                            <1> ;cmp   ah, [ptty]
                            <1> ;jne   short gcw2
                            <1> ;; 14/02/2014
                            <1> ;cmp   byte [u.uno], 1
                            <1> ;jna   short gcw0
                            <1> ;gcw2:
0000652E E8EEEFFFFF         <1>  call  sleep
                            <1>  ;
                            <1>  ; 20/09/2013
00006533 8A25[CE740000]     <1>  mov   ah, [u.ttyn]
00006539 30C0               <1>  xor   al, al
0000653B EB9B               <1>  jmp   short getc_n
                            <1> ;gcw3:
                            <1> gcw2:  ; 15/09/2015
                            <1>  ; 10/10/2013
0000653D 30C9               <1>  xor   cl, cl
0000653F EBC8               <1>  jmp   short getc_sn
                            <1>
                            <1> sndc:   ; <Send character>
                            <1>  ;
                            <1>  ; 21/11/2015
                            <1>  ; 18/11/2015
                            <1>  ; 17/11/2015
                            <1>  ; 16/11/2015
                            <1>  ; 11/11/2015
                            <1>  ; 10/11/2015
                            <1>  ; 09/11/2015
                            <1>  ; 08/11/2015
                            <1>  ; 07/11/2015
                            <1>  ; 06/11/2015 (serial4.asm, 'sendchr')
                            <1>  ; 29/10/2015
                            <1>  ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
                            <1>  ; 14/05/2013 - 28/07/2014 (Retro UNIX 8086 v1)
                            <1>  ;
                            <1>  ; Retro UNIX 8086 v1 feature only !
                            <1>  ;
                            <1>  ; ah = [u.ttyn]
                            <1>  ;
```

```
                                 <1>  ; 30/06/2015
00006541 80EC08                  <1>  sub   ah, 8 ; ; 0 = tty8 or 1 = tty9
                                 <1>  ; 07/11/2015
00006544 0FB6DC                  <1>  movzx ebx, ah ; serial port index (0 or 1)
                                 <1> sndc0:
                                 <1>  ; 07/11/2015
00006547 E82EF0FFFF              <1>  call  isintr ; quit (ctrl+break) check
0000654C 7405                    <1>  jz    short sndc1
0000654E E82BDCFFFF              <1>  call  intract ; quit (ctrl+break) check
                                 <1>  ; CPU will jump to 'sysexit' if 'u.quit' = 0FFFFh (yes)
                                 <1> sndc1:
                                 <1>  ; 16/11/2015
00006553 6689C1                  <1>  mov   cx, ax      ; *** al = character (to be sent)
                                 <1> sndcx:
00006556 8A83[1A710000]          <1>  mov   al, [ebx+schar] ; last sent character
0000655C 8AA3[18710000]          <1>  mov   ah, [ebx+rchar] ; last received character
                                 <1>  ;
                                 <1>  ; 17/11/2015
                                 <1>  ; check 'request for response' status
00006562 80BB[14710000]00        <1>  cmp   byte [ebx+req_resp], 0
00006569 7411                    <1>  jz    short query
                                 <1>  ; 18/11/2015
0000656B C683[14710000]00        <1>  mov   byte [ebx+req_resp], 0
                                 <1> response:
00006572 FE05[17710000]          <1>  inc   byte [comqr] ; query or response status
00006578 B0FF                    <1>  mov   al, 0FFh
0000657A EB14                    <1>  jmp   short sndc3
                                 <1> query:
0000657C 08C0                    <1>  or    al, al  ; 0 = query (also end of text)
0000657E 750E                    <1>  jnz   short sndc2 ; normal character
                                 <1>  ;cmp  ah, 0FFh     ; is it responded by terminal ?
                                 <1>  ;je   short sndc2  ; yes, already responded
                                 <1>  ; 16/11/2015
                                 <1>  ; query: request for response (again)
00006580 8883[18710000]          <1>  mov   [ebx+rchar], al ; 0 ; reset
00006586 FE05[17710000]          <1>  inc   byte [comqr] ; query or response status
0000658C EB02                    <1>  jmp   short sndc3
                                 <1> sndc2:
0000658E 88C8                    <1>  mov   al, cl      ; *** character (to be sent)
                                 <1> sndc3:
00006590 8883[1A710000]          <1>  mov   [ebx+schar], al ; current character (to be sent)
00006596 88D8                    <1>  mov   al, bl ; 0 or 1 (serial port index)
                                 <1>  ; 30/06/2015
00006598 E884D5FFFF              <1>  call  sp_status ; get serial port status
                                 <1>  ; AL = Line status, AH = Modem status
                                 <1>  ; 07/11/2015
0000659D A880                    <1>  test  al, 80h
0000659F 7504                    <1>  jnz   short sndc4
000065A1 A820                    <1>  test  al, 20h     ; Transmitter holding register empty ?
000065A3 751D                    <1>  jnz   short sndc5
                                 <1> sndc4: ; Check line status again
                                 <1>  ; 16/11/2015
000065A5 6651                    <1>  push  cx
000065A7 B906000000              <1>  mov   ecx, 6 ; 6*30 micro seconds (~5556 chars/second)
000065AC E87BB0FFFF              <1>  call  WAITF
000065B1 6659                    <1>  pop   cx
                                 <1>  ;
000065B3 88D8                    <1>  mov   al, bl ; 0 or 1 (serial port index)
000065B5 E867D5FFFF              <1>  call  sp_status ; get serial port status
                                 <1>  ; 16/11/2015
                                 <1>  ; 09/11/2015
                                 <1>  ; 08/11/2015
000065BA A880                    <1>  test  al, 80h     ; time out error
000065BC 756C                    <1>         jnz     short sndc7
000065BE A820                    <1>  test  al, 20h     ; Transmitter holding register empty ?
000065C0 7468                    <1>         jz   short sndc7
                                 <1> sndc5:
000065C2 8A83[1A710000]          <1>  mov   al, [ebx+schar] ; character (to be sent)
000065C8 66BAF803                <1>  mov   dx, 3F8h   ; data port (COM2)
000065CC 28DE                    <1>  sub   dh, bl
000065CE EE                      <1>  out   dx, al         ; send on serial port
                                 <1>  ; 10/11/2015
                                 <1>  ; delay for 3*30 (3*(15..80)) micro seconds
                                 <1>  ; (to improve text flow to the terminal)
                                 <1>  ; ('diskette.inc': 'WAITF')
                                 <1>  ; Uses port 61h, bit 4 to have CPU speed independent waiting.
                                 <1>  ; (refresh periods = 1 per 30 microseconds on most machines)
000065CF 6651                    <1>  push  cx
000065D1 B906000000              <1>  mov   ecx, 6 ; 6*30 micro seconds (~5556 chars/second)
000065D6 E851B0FFFF              <1>  call  WAITF
000065DB 6659                    <1>  pop   cx
                                 <1>        ;
000065DD E898EFFFFF              <1>  call  isintr ; quit (ctrl+break) check
000065E2 7405                    <1>  jz    short sndc6
000065E4 E895DBFFFF              <1>  call  intract ; quit (ctrl+break) check
                                 <1>  ; CPU will jump to 'sysexit' if 'u.quit' = 0FFFFh (yes)
                                 <1>  ;21/11/2015
                                 <1> sndc6:
                                 <1>  ; 18/11/2015
                                 <1>  ; 07/11/2015
000065E9 88D8                    <1>  mov   al, bl ; al = 0 (tty8) or 1 (tty9)
000065EB E831D5FFFF              <1>  call  sp_status ; get serial port status
```

```
                             <1>  ; AL = Line status, AH = Modem status
000065F0 3C80                <1>  cmp   al, 80h
000065F2 7336                <1>  jnb   short sndc7
                             <1>  ;
000065F4 803D[17710000]01    <1>  cmp   byte [comqr], 1  ; 'query or response' ?
000065FB 7248                <1>  jb    short sndc8        ; no, normal character
000065FD 883D[17710000]      <1>  mov   byte [comqr], bh ; 0 ; reset
                             <1>  ; 17/11/2015
00006603 E885EEFFFF          <1>  call  idle
                             <1>  ;
00006608 38BB[1A710000]      <1>  cmp   [ebx+schar], bh ; 0 ; query ?
0000660E 0F877AFFFFFF        <1>         ja      sndc2       ; response (will be followed by
                             <1>                  ; a normal character)
                             <1>  ; Query request must be responded by the terminal
                             <1>  ; before sending a normal character !
00006614 53                  <1>  push  ebx
00006615 6651                <1>  push  cx ; *** cl = character (to be sent)
00006617 8A25[CE740000]      <1>  mov   ah, [u.ttyn]
0000661D E8FFEEFFFF          <1>  call  sleep ; this process will be awakened by
                             <1>              ; received data available interrupt
00006622 6659                <1>  pop   cx ; *** cl = character (to be sent)
00006624 5B                  <1>  pop   ebx
00006625 E92CFFFFFF          <1>         jmp  sndcx
                             <1> sndc7:
                             <1>   ; 16/11/2015
0000662A 803D[17710000]01    <1>  cmp   byte [comqr], 1 ; 'query or response' ?
00006631 7213                <1>  jb    short sndc9       ; no
                             <1>  ;
00006633 88BB[18710000]      <1>  mov   [ebx+rchar], bh ; 0 ; reset
00006639 88BB[1A710000]      <1>  mov   [ebx+schar], bh ; 0 ; reset
                             <1>  ;
0000663F 883D[17710000]      <1>  mov   byte [comqr], bh ; 0 ; reset
                             <1> sndc8:
00006645 F5                  <1>  cmc  ; jnc -> jc, jb -> jnb
                             <1> sndc9:
                             <1>  ; AL = Line status, AH = Modem status
00006646 C3                  <1>  retn
                             <1>
                             <1> putc:
                             <1>  ; 13/08/2015
                             <1>  ; 30/06/2015 (Retro UNIX 386 v1 - Beginning)
                             <1>  ; 15/05/2013 - 27/07/2014 (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 modification !
                             <1>  ;
                             <1>  ; 'putc' puts a character
                             <1>  ;      onto requested (tty) video page or
                             <1>  ;      serial port
                             <1>  ; INPUTS ->
                             <1>  ;     AL = ascii code of the character
                             <1>  ;     AH = video page (tty) number (0 to 7)
                             <1>  ;                   (8 is COM1, 9 is COM2)
                             <1>  ; OUTPUTS ->
                             <1>  ;    (If AL input is 1) ZF=1 -> 'empty buffer' (no chars)
                             <1>  ;                       ZF=0 -> AX has (current) character
                             <1>  ;     cf=0 and AH = 0 -> no error
                             <1>  ;     cf=1 and AH > 0 -> error (only for COM1 and COM2)

                             <1>  ;
                             <1>  ; Original UNIX V1 'putc':
                             <1>  ;     put a character at the end of character list
                             <1>  ;
                             <1>  ; ((Modified registers: eAX, eBX, eCX, eDX, eSI, eDI))
                             <1>  ;
00006647 80FC07              <1>  cmp   ah, 7
0000664A 0F87F1FEFFFF        <1>         ja      sndc
                             <1>  ; 30/06/2015
00006650 0FB6DC              <1>  movzx ebx, ah
                             <1>  ; 13/08/2015
00006653 B407                <1>  mov   ah, 07h ; black background, light gray character color
00006655 E9C3AEFFFF          <1>  jmp   write_tty ; 'video.inc'
                             <1>
                             <1> get_cpos:
                             <1>  ; 29/06/2015 (Retro UNIX 386 v1)
                             <1>  ; 04/12/2013 (Retro UNIX 8086 v1 - 'sysgtty')
                             <1>  ;
                             <1>  ; INPUT -> bl = video page number
                             <1>  ; RETURN -> dx = cursor position
                             <1>
0000665A 53                  <1>  push  ebx
0000665B 83E30F              <1>  and   ebx, 0Fh ; 07h ; tty0 to tty7
0000665E D0E3                <1>  shl   bl, 1
00006660 81C3[C6700000]      <1>  add   ebx, cursor_posn
00006666 668B13              <1>  mov   dx, [ebx]
00006669 5B                  <1>  pop   ebx
0000666A C3                  <1>  retn
                             <1>
                             <1> read_ac_current:
                             <1>  ; 29/06/2015 (Retro UNIX 386 v1)
                             <1>  ; 04/12/2013 (Retro UNIX 8086 v1 - 'sysgtty')
                             <1>  ;
                             <1>  ; INPUT -> bl = video page number
                             <1>  ; RETURN -> ax = character (al) and attribute (ah)
                             <1>
0000666B E82EB0FFFF          <1>  call  find_position ; 'video.inc'
                             <1>  ; dx = status port
                             <1>  ; esi = cursor location/address
00006670 81C600800B00        <1>  add   esi, 0B8000h      ; 30/08/2014 (Retro UNIX 386 v1)
00006676 668B06              <1>  mov   ax, [esi]   ; get the character and attribute
00006679 C3                  <1>  retn
                             <1>
                             <1> syssleep:
                             <1>  ; 29/06/2015 - (Retro UNIX 386 v1)
                             <1>  ; 11/06/2014 - (Retro UNIX 8086 v1)
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 feature only
                             <1>  ; (INPUT -> none)
                             <1>  ;
0000667A 0FB61D[C9740000]    <1>  movzx ebx, byte [u.uno] ; process number
00006681 8AA3[D5710000]      <1>  mov   ah, [ebx+p.ttyc-1] ; current/console tty
00006687 E895EEFFFF          <1>  call  sleep
0000668C E90BDAFFFF          <1>  jmp   sysret
                             <1>
                             <1> vp_clr:
                             <1>  ; Reset/Clear Video Page
                             <1>  ;
                             <1>  ; 30/06/2015 - (Retro UNIX 386 v1)
                             <1>  ; 21/05/2013 - 30/10/2013(Retro UNIX 8086 v1) (U0.ASM)
                             <1>  ;
                             <1>  ; Retro UNIX 8086 v1 feature only !
                             <1>  ;
                             <1>  ; INPUTS ->
                             <1>  ;   BL = video page number
                             <1>  ;
                             <1>  ; OUTPUT ->
                             <1>  ;   none
                             <1>  ; ((Modified registers: eAX, BH, eCX, eDX, eSI, eDI))
                             <1>  ;
                             <1>  ; 04/12/2013
00006691 28C0                <1>  sub   al, al
                             <1>  ; al = 0 (clear video page)
                             <1>  ; bl = video page
00006693 B407                <1>  mov   ah, 07h
                             <1>  ; ah = 7 (attribute/color)
00006695 6631C9              <1>  xor   cx, cx ; 0, left upper column (cl) & row (cl)
00006698 66BA4F18            <1>  mov   dx, 184Fh ; right lower column & row (dl=24, dh=79)
0000669C E829B0FFFF          <1>  call  scroll_up
                             <1>  ; bl = video page
000066A1 6631D2              <1>  xor   dx, dx ; 0 (cursor position)
000066A4 E998AFFFFF          <1>  jmp   set_cpos
                             <1>
                             <1> sysmsg:
                             <1>  ; 11/11/2015
                             <1>  ; 01/07/2015 - (Retro UNIX 386 v1 feature only!)
                             <1>  ; Print user-application message on user's console tty
                             <1>  ;
                             <1>  ; Input -> EBX = Message address
                             <1>  ;          ECX = Message length (max. 255)
                             <1>  ;          DL = Color (IBM PC Rombios color attributes)
                             <1>  ;
000066A9 81F9FF000000        <1>  cmp   ecx, MAX_MSG_LEN ; 255
000066AF 0F87E7D9FFFF        <1>  ja    sysret ; nothing to do with big message size
000066B5 08C9                <1>  or    cl, cl
000066B7 0F84DFD9FFFF        <1>  jz    sysret
000066BD 20D2                <1>  and   dl, dl
000066BF 7502                <1>  jnz   short sysmsg0
000066C1 B207                <1>  mov   dl, 07h ; default color
                             <1>        ; (black background, light gray character)
                             <1> sysmsg0:
000066C3 891D[A0740000]      <1>  mov   [u.base], ebx
000066C9 8815[D7700000]      <1>  mov   [ccolor], dl ; color attributes
000066CF 89E5                <1>  mov   ebp, esp
000066D1 31DB                <1>  xor   ebx, ebx ; 0
000066D3 891D[A8740000]      <1>  mov   [u.nread], ebx ; 0
                             <1>  ;
000066D9 381D[E1740000]      <1>  cmp   [u.kcall], bl ; 0
000066DF 7769                <1>  ja    short sysmsgk ; Temporary (01/07/2015)
                             <1>  ;
000066E1 890D[A4740000]      <1>  mov   [u.count], ecx
000066E7 41                  <1>  inc   ecx ; + 00h ; ASCIZZ
000066E8 29CC                <1>  sub   esp, ecx
000066EA 89E7                <1>  mov   edi, esp
000066EC 89E6                <1>  mov   esi, esp
000066EE 66891D[DF740000]    <1>  mov   [u.pcount], bx ; reset page (phy. addr.) counter
                             <1>  ; 11/11/2015
000066F5 8A25[B0740000]      <1>  mov   ah, [u.ttyp] ; recent open tty
                             <1>  ; 0 = none
000066FB FECC                <1>  dec   ah
000066FD 790C                <1>  jns   short sysmsg1
000066FF 8A1D[C9740000]      <1>  mov   bl, [u.uno] ; process number
00006705 8AA3[D5710000]      <1>  mov   ah, [ebx+p.ttyc-1] ; user's (process's) console tty
                             <1> sysmsg1:
0000670B 8825[CE740000]      <1>  mov   [u.ttyn], ah
                             <1> sysmsg2:
```

```
00006711 E895F5FFFF          <1>  call  cpass
00006716 7416                <1>  jz    short sysmsg5
00006718 AA                  <1>  stosb
00006719 20C0                <1>  and   al, al
0000671B 75F4                <1>  jnz   short sysmsg2
                             <1> sysmsg3:
0000671D 80FC07              <1>  cmp   ah, 7 ; tty number
00006720 7711                <1>  ja    short sysmsg6 ; serial port
00006722 E83E000000          <1>  call  print_cmsg
                             <1> sysmsg4:
00006727 89EC                <1>  mov   esp, ebp
00006729 E96ED9FFFF          <1>  jmp   sysret
                             <1> sysmsg5:
0000672E C60700              <1>  mov   byte [edi], 0
00006731 EBEA                <1>  jmp   short sysmsg3
                             <1> sysmsg6:
00006733 8A06                <1>  mov   al, [esi]
00006735 E807FEFFFF          <1>  call  sndc
0000673A 72EB                <1>  jc    short sysmsg4
0000673C 803E00              <1>  cmp   byte [esi], 0  ; 0 is stop character
0000673F 76E6                <1>  jna   short sysmsg4
00006741 46                  <1>  inc   esi
00006742 8A25[CE740000]      <1>  mov   ah, [u.ttyn]
00006748 EBE9                <1>  jmp   short sysmsg6
                             <1>
                             <1> sysmsgk: ; Temporary (01/07/2015)
                             <1>  ; The message has been sent by Kernel (ASCIIZ string)
                             <1>  ; (ECX -character count- will not be considered)
0000674A 8B35[A0740000]      <1>  mov   esi, [u.base]
00006750 8A25[D6700000]      <1>  mov   ah, [ptty] ; present/current screen (video page)
00006756 8825[CE740000]      <1>  mov   [u.ttyn], ah
0000675C C605[E1740000]00    <1>  mov   byte [u.kcall], 0
00006763 EBB8                <1>  jmp   short sysmsg3
                             <1>
                             <1>
                             <1> print_cmsg:
                             <1>  ; 01/07/2015 (retro UNIX 386 v1 feature only !)
                             <1>  ;
                             <1>  ; print message (on user's console tty)
                             <1>  ;     with requested color
                             <1>  ;
                             <1>  ; INPUTS:
                             <1>  ;     esi = message address
                             <1>  ;     [u.ttyn] = tty number (0 to 7)
                             <1>  ;     [ccolor] = color attributes (IBM PC BIOS colors)
                             <1>  ;
00006765 AC                  <1>  lodsb
                             <1> pcmsg1:
00006766 56                  <1>  push  esi
00006767 0FB61D[CE740000]    <1>         movzx   ebx, byte [u.ttyn]
0000676E 8A25[D7700000]      <1>  mov   ah, [ccolor]
00006774 E8A4ADFFFF          <1>  call  write_tty
00006779 5E                  <1>  pop   esi
0000677A AC                  <1>  lodsb
0000677B 20C0                <1>  and   al, al  ; 0
0000677D 75E7                <1>  jnz   short pcmsg1
0000677F C3                  <1>  retn
                             <1>
                             <1> sysgeterr:
                             <1>  ; 09/12/2015
                             <1>  ; 21/09/2015 - (Retro UNIX 386 v1 feature only!)
                             <1>  ; Get last error number or page fault count
                             <1>  ; (for debugging)
                             <1>  ;
                             <1>  ; Input -> EBX = return type
                             <1>  ;        0 = last error code (which is in 'u.error')
                             <1>  ;        FFFFFFFFh = page fault count for running process
                             <1>  ;        FFFFFFFEh = total page fault count
                             <1>  ;        1 .. FFFFFFFDh = undefined
                             <1>  ;
                             <1>  ; Output -> EAX = last error number or page fault count
                             <1>  ;        (depending on EBX input)
                             <1>  ;
00006780 21DB                <1>  and   ebx, ebx
00006782 750F                <1>  jnz   short glerr_2
                             <1> glerr_0:
00006784 A1[CF740000]        <1>  mov   eax, [u.error]
                             <1> glerr_1:
00006789 A3[80740000]        <1>  mov   [u.r0], eax
0000678E E909D9FFFF          <1>  jmp   sysret
                             <1> glerr_2:
00006793 43                  <1>  inc   ebx ; FFFFFFFFh -> 0, FFFFFFFEh -> FFFFFFFFh
00006794 74FD                <1>  jz    short glerr_2 ; page fault count for process
00006796 43                  <1>  inc   ebx ; FFFFFFFFh -> 0
00006797 75EB                <1>  jnz   short glerr_0
00006799 A1[50810000]        <1>  mov   eax, [PF_Count] ; total page fault count
0000679E EBE9                <1>         jmp     short glerr_1
                             <1> glerr_3:
000067A0 A1[E3740000]        <1>  mov   eax, [u.pfcount]
000067A5 EBE2                <1>  jmp   short glerr_1

                                 ; 07/03/2015
                                 ; Temporary Code
                                 display_disks:
000067A7 803D[D86D0000]00         cmp   byte [fd0_type], 0
000067AE 7605                     jna   short ddsks1
000067B0 E87D000000               call  pdskm
                                 ddsks1:
000067B5 803D[D96D0000]00         cmp   byte [fd1_type], 0
000067BC 760C                     jna   short ddsks2
000067BE C605[2B6D0000]31         mov   byte [dskx], '1'
000067C5 E868000000               call  pdskm
                                 ddsks2:
000067CA 803D[DA6D0000]00         cmp   byte [hd0_type], 0
000067D1 7654                     jna   short ddsk6
000067D3 66C705[296D0000]68-      mov   word [dsktype], 'hd'
000067DB 64
000067DC C605[2B6D0000]30         mov   byte [dskx], '0'
000067E3 E84A000000               call  pdskm
                                 ddsks3:
000067E8 803D[DB6D0000]00         cmp   byte [hd1_type], 0
000067EF 7636                     jna   short ddsk6
000067F1 C605[2B6D0000]31         mov   byte [dskx], '1'
000067F8 E835000000               call  pdskm
                                 ddsks4:
000067FD 803D[DC6D0000]00         cmp   byte [hd2_type], 0
00006804 7621                     jna   short ddsk6
00006806 C605[2B6D0000]32         mov   byte [dskx], '2'
0000680D E820000000               call  pdskm
                                 ddsks5:
00006812 803D[DD6D0000]00         cmp   byte [hd3_type], 0
00006819 760C                     jna   short ddsk6
0000681B C605[2B6D0000]33         mov   byte [dskx], '3'
00006822 E80B000000               call  pdskm
                                 ddsk6:
00006827 BE[3A6D0000]             mov   esi, nextline
0000682C E806000000               call  pdskml
                                 pdskm_ok:
00006831 C3                       retn
                                 pdskm:
00006832 BE[276D0000]             mov   esi, dsk_ready_msg
                                 pdskml:
00006837 AC                       lodsb
00006838 08C0                     or    al, al
0000683A 74F5                     jz    short pdskm_ok
0000683C 56                       push  esi
0000683D 31DB                     xor   ebx, ebx ; 0
                                              ; Video page 0 (bl=0)
0000683F B407                     mov   ah, 07h ; Black background,
                                              ; light gray forecolor
00006841 E8D7ACFFFF               call  write_tty
00006846 5E                       pop   esi
00006847 EBEE                     jmp   short pdskml

00006849 90<rept>                align 16

                                 gdt:   ; Global Descriptor Table
                                  ; (30/07/2015, conforming cs)
                                  ; (26/03/2015)
                                  ; (24/03/2015, tss)
                                  ; (19/03/2015)
                                  ; (29/12/2013)
                                  ;
00006850 0000000000000000         dw 0, 0, 0, 0           ; NULL descriptor
                                  ; 18/08/2014
                                              ; 8h kernel code segment, base = 00000000h
00006858 FFFF0000009ACF00         dw 0FFFFh, 0, 9A00h, 00CFh    ; KCODE
                                              ; 10h kernel data segment, base = 00000000h
00006860 FFFF00000092CF00         dw 0FFFFh, 0, 9200h, 00CFh    ; KDATA
                                              ; 1Bh user code segment, base address = 400000h ; CORE
00006868 FFFB000040FACF00         dw 0FBFFh, 0, 0FA40h, 00CFh   ; UCODE
                                              ; 23h user data segment, base address = 400000h ; CORE
00006870 FFFB000040F2CF00         dw 0FBFFh, 0, 0F240h, 00CFh   ; UDATA
                                              ; Task State Segment
00006878 6700                     dw 0067h ; Limit = 103 ; (104-1, tss size = 104 byte,
                                                     ;  no IO permission in ring 3)
                                 gdt_tss0:
0000687A 0000                     dw 0  ; TSS base address, bits 0-15
                                 gdt_tss1:
0000687C 00                       db 0  ; TSS base address, bits 16-23
                                                    ; 49h
0000687D E9                       db 11101001b ; E9h => P=1/DPL=11/0/1/0/B/1 --> B = Task is busy (1)
0000687E 00                       db 0 ; G/0/0/AVL/LIMIT=0000 ; (Limit bits 16-19 = 0000) (G=0, 1 byte)
                                 gdt_tss2:
0000687F 00                       db 0  ; TSS base address, bits 24-31

                                 gdt_end:
                                  ;; 9Ah = 1001 1010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
                                                          ;; Type= 1 (code)/C=0/R=1/A=0
                                        ; P= Present, DPL=0=ring 0,  1= user (0= system)
                                        ; 1= Code C= non-Conforming, R= Readable, A = Accessed

                                  ;; 92h = 1001 0010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
                                                          ;; Type= 0 (data)/E=0/W=1/A=0
                                        ; P= Present, DPL=0=ring 0,  1= user (0= system)
                                        ; 0= Data E= Expansion direction (1= down, 0= up)
                                        ; W= Writeable, A= Accessed

                                  ;; FAh = 1111 1010b (GDT byte 5) P=1/DPL=11/1/TYPE=1010,
                                                          ;; Type= 1 (code)/C=0/R=1/A=0
                                        ; P= Present, DPL=3=ring 3,  1= user (0= system)
                                        ; 1= Code C= non-Conforming, R= Readable, A = Accessed

                                  ;; F2h = 1111 0010b (GDT byte 5) P=1/DPL=11/1/TYPE=0010,
                                                          ;; Type= 0 (data)/E=0/W=1/A=0
                                        ; P= Present, DPL=3=ring 3,  1= user (0= system)
                                        ; 0= Data E= Expansion direction (1= down, 0= up)

                                  ;; CFh = 1100 1111b (GDT byte 6) G=1/B=1/0/AVL=0, Limit=1111b (3)

                                        ;; Limit = FFFFFh (=> FFFFFh+1= 100000h) // bits 0-15, 48-51 //
                                        ;      = 100000h * 1000h (G=1) = 4GB
                                        ;; Limit = FFBFFh (=> FFBFFh+1= FFC00h) // bits 0-15, 48-51 //
                                        ;      = FFC00h * 1000h (G=1) = 4GB - 4MB
                                        ; G= Granularity (1= 4KB), B= Big (32 bit),
                                        ; AVL= Available to programmers

                                 gdtd:
00006880 2F00                            dw gdt_end - gdt - 1    ; Limit (size)
00006882 [50680000]                      dd gdt                 ; Address of the GDT

                                  ; 20/08/2014
                                 idtd:
00006886 FF01                            dw idt_end - idt - 1    ; Limit (size)
00006888 [406E0000]                      dd idt                 ; Address of the IDT

                                 Align 4

                                  ; 21/08/2014
                                 ilist:
                                  ;times      32 dd cpu_except ; INT 0 to INT 1Fh
                                  ;
                                  ; Exception list
                                  ; 25/08/2014
0000688C [34090000]               dd    exc0  ; 0h,  Divide-by-zero Error
00006890 [3B090000]               dd    exc1
00006894 [42090000]               dd    exc2
00006898 [49090000]               dd    exc3
0000689C [4D090000]               dd    exc4
000068A0 [51090000]               dd    exc5
000068A4 [55090000]               dd    exc6  ; 06h,  Invalid Opcode
000068A8 [59090000]               dd    exc7
000068AC [5D090000]               dd    exc8
000068B0 [61090000]               dd    exc9
000068B4 [65090000]               dd    exc10
000068B8 [69090000]               dd    exc11
000068BC [6D090000]               dd    exc12
000068C0 [71090000]               dd    exc13 ; 0Dh, General Protection Fault
000068C4 [75090000]               dd    exc14 ; 0Eh, Page Fault
000068C8 [79090000]               dd    exc15
000068CC [7D090000]               dd    exc16
000068D0 [81090000]               dd    exc17
000068D4 [85090000]               dd    exc18
000068D8 [89090000]               dd    exc19
000068DC [8D090000]               dd    exc20
000068E0 [91090000]               dd    exc21
000068E4 [95090000]               dd    exc22
000068E8 [99090000]               dd    exc23
000068EC [9D090000]               dd    exc24
000068F0 [A1090000]               dd    exc25
000068F4 [A5090000]               dd    exc26
000068F8 [A9090000]               dd    exc27
000068FC [AD090000]               dd    exc28
00006900 [B1090000]               dd    exc29
00006904 [B5090000]               dd    exc30
00006908 [B9090000]               dd    exc31
                                  ; Interrupt list
0000690C [6A070000]               dd    timer_int   ; INT 20h
                                        ;dd   irq0
00006910 [730C0000]               dd    keyb_int    ; 27/08/2014
                                        ;dd   irq1
00006914 [8A080000]               dd    irq2
                                        ; COM2 int
00006918 [8E080000]               dd    irq3
                                        ; COM1 int
0000691C [99080000]               dd    irq4
00006920 [A4080000]               dd    irq5
                                 ;DISKETTE_INT: ;06/02/2015
00006924 [AE270000]               dd    fdc_int           ; 16/02/2015, IRQ 6 handler
                                        ;dd   irq6
                                 ; Default IRQ 7 handler against spurious IRQs (from master PIC)
                                 ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
00006928 [200C0000]               dd    default_irq7      ; 25/02/2015
                                        ;dd   irq7
                                 ; Real Time Clock Interrupt
0000692C [C30A0000]               dd    rtc_int           ; 23/02/2015, IRQ 8 handler
```

```
                                        ;dd   irq8  ; INT 28h
00006930 [B4080000]               dd    irq9
00006934 [B8080000]               dd    irq10
00006938 [BC080000]               dd    irq11
0000693C [C0080000]               dd    irq12
00006940 [C4080000]               dd    irq13
                                 ;HDISK_INT1:  ;06/02/2015
00006944 [EA2F0000]               dd    hdc1_int    ; 21/02/2015, IRQ 14 handler
                                        ;dd   irq14
                                 ;HDISK_INT2:  ;06/02/2015
00006948 [11300000]               dd    hdc2_int    ; 21/02/2015, IRQ 15 handler
                                        ;dd   irq15 ; INT 2Fh
                                        ; 14/08/2015
0000694C [803F0000]               dd    sysent            ; INT 30h (system calls)

                                  ;dd   ignore_int
00006950 00000000                 dd    0

                                 ;;;
                                 ;;; 11/03/2015
                                 %include 'kybdata.inc'   ; KEYBOARD (BIOS) DATA
                             <1> ; Retro UNIX 386 v1 Kernel - KYBDATA.INC
                             <1> ; Last Modification: 11/03/2015
                             <1> ;       (Data Section for 'KEYBOARD.INC')
                             <1> ;
                             <1> ; ///////// KEYBOARD DATA //////////////
                             <1>
                             <1> ; 05/12/2014
                             <1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)
                             <1> ; 03/06/86  KEYBOARD BIOS
                             <1>
                             <1> ;---------------------------------------------------------------------------------
                             <1> ;KEY IDENTIFICATION SCAN TABLES
                             <1> ;---------------------------------------------------------------------------------
                             <1>
                             <1> ;----- TABLES FOR ALT CASE ------------
                             <1> ;----- ALT-INPUT-TABLE
00006954 524F50514B          <1> K30:   db    82,79,80,81,75
00006959 4C4D474849          <1>  db    76,77,71,72,73          ; 10 NUMBER ON KEYPAD
                             <1> ;----- SUPER-SHIFT-TABLE
0000695E 101112131415        <1>  db    16,17,18,19,20,21 ; A-Z TYPEWRITER CHARS
00006964 161718191E1F        <1>  db    22,23,24,25,30,31
0000696A 202122232425        <1>  db    32,33,34,35,36,37
00006970 262C2D2E2F30        <1>  db    38,44,45,46,47,48
00006976 3132                <1>  db    49,50
                             <1>
                             <1> ;----- TABLE OF SHIFT KEYS AND MASK VALUES
                             <1> ;----- KEY_TABLE
00006978 52                  <1> _K6:    db      INS_KEY                 ; INSERT KEY
00006979 3A4546381D          <1>  db    CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
0000697E 2A36                <1>          db      LEFT_KEY,RIGHT_KEY
                             <1> _K6L    equ     $-_K6
                             <1>
                             <1> ;----- MASK_TABLE
00006980 80                  <1> _K7:    db      INS_SHIFT               ; INSERT MODE SHIFT
00006981 4020100804          <1>  db    CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
00006986 0201                <1>  db    LEFT_SHIFT,RIGHT_SHIFT
                             <1>
                             <1> ;----- TABLES FOR CTRL CASE          ;---- CHARACTERS ------
00006988 1BFF00FFFFFF        <1> _K8:   db    27,-1,0,-1,-1,-1  ; Esc, 1, 2, 3, 4, 5
0000698E 1EFFFFFFFF1F        <1>  db    30,-1,-1,-1,-1,31 ; 6, 7, 8, 9, 0, -
00006994 FF7FFF111705        <1>  db    -1,127,-1,17,23,5 ; =, Bksp, Tab, Q, W, E
0000699A 12141915090F        <1>  db    18,20,25,21,9,15  ; R, T, Y, U, I, O
000069A0 101B1D0AFF01        <1>  db    16,27,29,10,-1,1  ; P, [, ], Enter, Ctrl, A
000069A6 13040607080A        <1>  db    19,4,6,7,8,10           ; S, D, F, G, H, J
000069AC 0B0CFFFFFFFF        <1>  db    11,12,-1,-1,-1,-1 ; K, L, :, ', `, LShift
000069B2 1C1A18031602        <1>  db    28,26,24,3,22,2         ; Bkslash, Z, X, C, V, B
000069B8 0E0DFFFFFFFF        <1>  db    14,13,-1,-1,-1,-1 ; N, M, ,, ., /, RShift
000069BE 96FF20FF            <1>  db    150,-1,' ',-1          ; *, ALT, Spc, CL
                             <1>  ;                       ;----- FUNCTIONS ------
000069C2 5E5F60616263        <1>  db    94,95,96,97,98,99 ; F1 - F6
000069C8 64656667FFFF        <1>  db    100,101,102,103,-1,-1   ; F7 - F10, NL, SL
000069CE 778D848E738F        <1>  db    119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5
000069D4 749075917692        <1>  db    116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
000069DA 93FFFFFF898A        <1>  db    147,-1,-1,-1,137,138    ; Del, SysReq, Undef, WT, F11, F12
                             <1>
                             <1> ;----- TABLES FOR LOWER CASE ----------
000069E0 1B3132333435363738- <1> K10:   db    27,'1234567890-=',8,9
000069E9 39302D3D0809        <1>
000069EF 71776572747975696F- <1>  db    'qwertyuiop[]',13,-1,'asdfghjkl;',39
000069F8 705B5D0DFF61736466- <1>
00006A01 67686A6B6C3B27      <1>
00006A08 60FF5C7A786376626E- <1>  db    96,-1,92,'zxcvbnm,./',-1,'*',-1,' ',-1
00006A11 6D2C2E2FFF2AFF20FF  <1>
                             <1> ;----- LC TABLE SCAN
00006A1A 3B3C3D3E3F          <1>  db    59,60,61,62,63          ; BASE STATE OF F1 - F10
00006A1F 4041424344          <1>  db    64,65,66,67,68
00006A24 FFFF                <1>  db    -1,-1             ; NL, SL
                             <1>
                             <1> ;----- KEYPAD TABLE
00006A26 474849FF4BFF        <1> K15:   db    71,72,73,-1,75,-1 ; BASE STATE OF KEYPAD KEYS
00006A2C 4DFF4F50515253      <1>  db    77,-1,79,80,81,82,83
```

```
00006A33 FFFF5C8586          <1>  db    -1,-1,92,133,134  ; SysRq, Undef, WT, F11, F12
                             <1>
                             <1> ;----- TABLES FOR UPPER CASE ----------
00006A38 1B21402324255E262A- <1> K11:   db    27,'!@#$%',94,'&*()_+',8,0
00006A41 28295F2B0800        <1>
00006A47 51574552545955494F- <1>  db    'QWERTYUIOP{}',13,-1,'ASDFGHJKL:"'
00006A50 507B7D0DFF41534446- <1>
00006A59 47484A4B4C3A22      <1>
00006A60 7EFF7C5A584356424E- <1>  db    126,-1,'|ZXCVBNM<>?',-1,'*',-1,' ',-1
00006A69 4D3C3E3FFF2AFF20FF  <1>
                             <1> ;----- UC TABLE SCAN
00006A72 5455565758          <1> K12:   db    84,85,86,87,88          ; SHIFTED STATE OF F1 - F10
00006A77 595A5B5C5D          <1>  db    89,90,91,92,93
00006A7C FFFF                <1>  db    -1,-1             ; NL, SL
                             <1>
                             <1> ;----- NUM STATE TABLE
00006A7E 3738392D3435362B31- <1> K14:   db    '789-456+1230.'         ; NUMLOCK STATE OF KEYPAD KEYS
00006A87 3233302E            <1>
                             <1>  ;
00006A8B FFFF7C8788          <1>  db    -1,-1,124,135,136 ; SysRq, Undef, WT, F11, F12
                             <1>
                             <1> Align  4
                             <1> ;---------------------------------------
                             <1> ;VIDEO DISPLAY DATA AREA      ;
                             <1> ;---------------------------------------
00006A90 03                  <1> CRT_MODE     db    3     ; CURRENT DISPLAY MODE (TYPE)
00006A91 29                  <1> CRT_MODE_SET db    29h   ; CURRENT SETTING OF THE 3X8 REGISTER
                             <1>                    ; (29h default setting for video mode 3)
                             <1>                    ; Mode Select register Bits
                             <1>                    ;   BIT 0 - 80x25 (1), 40x25 (0)
                             <1>                    ;   BIT 1 - ALPHA (0), 320x200 GRAPHICS (1)
                             <1>                    ;   BIT 2 - COLOR (0), BW (1)
                             <1>                    ;   BIT 3 - Video Sig. ENABLE (1), DISABLE (0)
                             <1>                    ;   BIT 4 - 640x200 B&W Graphics Mode (1)
                             <1>                    ;   BIT 5 - ALPHA mode BLINKING (1)
                             <1>                    ;   BIT 6, 7 - Not Used
                             <1>
                             <1> ; Mode 0 - 2Ch = 101100b ; 40x25 text, 16 gray colors
                             <1> ; Mode 1 - 28h = 101000b ; 40x25 text, 16 fore colors, 8 back colors
                             <1> ; Mode 2 - 2Dh = 101101b ; 80x25 text, 16 gray colors
                             <1> ; MODE 3 - 29h = 101001b ; 80x25 text, 16 fore color, 8 back color
                             <1> ; Mode 4 - 2Ah = 101010b ; 320x200 graphics, 4 colors
                             <1> ; Mode 5 - 2Eh = 101110b ; 320x200 graphics, 4 gray colors
                             <1> ; Mode 6 - 1Eh = 011110b ; 640x200 graphics, 2 colors
                             <1> ; Mode 7 - 29h = 101001b ; 80x25 text, black & white colors
                             <1> ; Mode & 37h = Video signal OFF
                             <1>
                             <1>
                             <1> ; 26/08/2014
                             <1> ; Retro UNIX 8086 v1 - UNIX.ASM (03/03/2014)
                             <1> ; Derived from IBM "pc-at"
                             <1> ; rombios source code (06/10/1985)
                             <1> ; 'dseg.inc'
                             <1>
                             <1> ;--------------------------------------;
                             <1> ;SYSTEM DATA AREA       ;
                             <1> ;---------------------------------------
00006A92 00                  <1> BIOS_BREAK   db    0           ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
                             <1>
                             <1> ;---------------------------------------
                             <1> ;KEYBOARD DATA AREAS          ;
                             <1> ;---------------------------------------
                             <1>
00006A93 00                  <1> KB_FLAG      db    0           ; KEYBOARD SHIFT STATE AND STATUS FLAGS
00006A94 00                  <1> KB_FLAG_1    db    0           ; SECOND BYTE OF KEYBOARD STATUS
00006A95 00                  <1> KB_FLAG_2    db    0           ; KEYBOARD LED FLAGS
00006A96 00                  <1> KB_FLAG_3    db    0           ; KEYBOARD MODE STATE AND TYPE FLAGS
00006A97 00                  <1> ALT_INPUT    db    0           ; STORAGE FOR ALTERNATE KEY PAD ENTRY
00006A98 [A86A0000]          <1> BUFFER_START dd    KB_BUFFER   ; OFFSET OF KEYBOARD BUFFER START
00006A9C [C86A0000]          <1> BUFFER_END   dd    KB_BUFFER + 32    ; OFFSET OF END OF BUFFER
00006AA0 [A86A0000]          <1> BUFFER_HEAD  dd    KB_BUFFER   ; POINTER TO HEAD OF KEYBOARD BUFFER
00006AA4 [A86A0000]          <1> BUFFER_TAIL  dd    KB_BUFFER   ; POINTER TO TAIL OF KEYBOARD BUFFER
                             <1> ; ------     HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
00006AA8 0000<rept>          <1> KB_BUFFER    times 16 dw 0           ; ROOM FOR 16 SCAN CODE ENTRIES
                             <1>
                             <1> ; /// End Of KEYBOARD DATA ///
                                 %include 'vidata.inc'    ; VIDEO (BIOS) DATA
                             <1> ; Retro UNIX 386 v1 Kernel - VIDATA.INC
                             <1> ; Last Modification: 11/03/2015
                             <1> ;          (Data section for 'VIDEO.INC')
                             <1> ;
                             <1> ; ///////// VIDEO DATA ///////////////
                             <1>
                             <1> video_params:
                             <1>  ; 02/09/2014 (Retro UNIX 386 v1)
                             <1>  ;ORGS.ASM ----- 06/10/85   COMPATIBILITY MODULE
                             <1>  ; VIDEO MODE 3
00006AC8 71505A0A1F0619      <1>  db    71h,50h,5Ah,0Ah,1Fh,6,19h     ; SET UP FOR 80X25
00006ACF 1C02070607          <1>  db    1Ch,2,7,6,7 ; cursor start = 6, cursor stop = 7
00006AD4 00000000            <1>  db    0,0,0,0
                             <1>
                             <1> ; /// End Of VIDEO DATA ///
```

```
                                  %include 'diskdata.inc'  ; DISK (BIOS) DATA (initialized)
                              <1> ; Retro UNIX 386 v1 Kernel - DISKDATA.INC
                              <1> ; Last Modification: 11/03/2015
                              <1> ;(Initialized Disk Parameters Data section for 'DISKIO.INC')
                              <1> ;
                              <1> ; ******************************************************************************
                              <1> 
                              <1> ;----------------------------------------
                              <1> ;80286 INTERRUPT LOCATIONS     :
                              <1> ;REFERENCED BY POST & BIOS     :
                              <1> ;----------------------------------------
                              <1> 
00006AD8 [3B6B0000]           <1> DISK_POINTER:dd    MD_TBL6           ; Pointer to Diskette Parameter Table
                              <1> 
                              <1> ; IBM PC-XT Model 286 source code ORGS.ASM (06/10/85) - 14/12/2014
                              <1> ;-----------------------------------------------------------------
                              <1> ; DISK_BASE                                       :
                              <1> ;THIS IS THE SET OF PARAMETERS REQUIRED FOR             :
                              <1> ;DISKETTE OPERATION. THEY ARE POINTED AT BY THE         :
                              <1> ;DATA VARIABLE @DISK_POINTER. TO MODIFY THE PARAMETERS,       :
                              <1> ;BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT          :
                              <1> ;-----------------------------------------------------------------
                              <1> 
                              <1> ;DISK_BASE:
                              <1> ;DB    11011111B   ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
                              <1> ;DB    2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
                              <1> ;DB    MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
                              <1> ;DB    2           ; 512 BYTES/SECTOR
                              <1> ;;DB   15          ; EOT (LAST SECTOR ON TRACK)
                              <1> ;db    18          ; (EOT for 1.44MB diskette)
                              <1> ;DB    01BH        ; GAP LENGTH
                              <1> ;DB    0FFH        ; DTL
                              <1> ;;DB   054H        ; GAP LENGTH FOR FORMAT
                              <1> ;db    06ch        ; (for 1.44MB dsikette)
                              <1> ;DB    0F6H        ; FILL BYTE FOR FORMAT
                              <1> ;DB    15          ; HEAD SETTLE TIME (MILLISECONDS)
                              <1> ;DB    8           ; MOTOR START TIME (1/8 SECONDS)
                              <1> 
                              <1> ;----------------------------------------
                              <1> ;ROM BIOS DATA AREAS           :
                              <1> ;----------------------------------------
                              <1> 
                              <1> ;DATA        SEGMENT AT 40H          ; ADDRESS= 0040:0000
                              <1> 
                              <1> ;@EQUIP_FLAG DW    ?           ; INSTALLED HARDWARE FLAGS
                              <1> 
                              <1> ;----------------------------------------
                              <1> ;DISKETTE DATA AREAS           :
                              <1> ;----------------------------------------
                              <1> 
                              <1> ;@SEEK_STATUSDB    ?           ; DRIVE RECALIBRATION STATUS
                              <1> ;                        ; BIT 3-0 = DRIVE 3-0 RECALIBRATION
                              <1> ;                        ; BEFORE NEXT SEEK IF BIT IS = 0
                              <1> ;@MOTOR_STATUS     DB    ?           ; MOTOR STATUS
                              <1> ;                        ; BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
                              <1> ;                        ; BIT 7 = CURRENT OPERATION IS A WRITE
                              <1> ;@MOTOR_COUNTDB    ?           ; TIME OUT COUNTER FOR MOTOR(S) TURN OFF
                              <1> ;@DSKETTE_STATUS DB      ?           ; RETURN CODE STATUS BYTE
                              <1> ;                        ; CMD_BLOCK  IN STACK FOR DISK OPERATION
                              <1> ;@NEC_STATUS DB    7 DUP(?)    ; STATUS BYTES FROM DISKETTE OPERATION
                              <1> 
                              <1> ;----------------------------------------
                              <1> ;POST AND BIOS WORK DATA AREA :
                              <1> ;----------------------------------------
                              <1> 
                              <1> ;@INTR_FLAG  DB    ?           ; FLAG INDICATING AN INTERRUPT HAPPENED
                              <1> 
                              <1> ;----------------------------------------
                              <1> ;TIMER DATA AREA         :
                              <1> ;----------------------------------------
                              <1> 
                              <1> ; 17/12/2014  (IRQ 0 - INT 08H)
                              <1> ;TIMER_LOW   equ   46Ch        ; Timer ticks (counter)  @ 40h:006Ch
                              <1> ;TIMER_HIGH  equ   46Eh        ; (18.2 timer ticks per second)
                              <1> ;TIMER_OFL   equ   470h        ; Timer - 24 hours flag  @ 40h:0070h
                              <1> 
                              <1> ;----------------------------------------
                              <1> ;ADDITIONAL MEDIA DATA         :
                              <1> ;----------------------------------------
                              <1> 
                              <1> ;@LASTRATE   DB    ?           ; LAST DISKETTE DATA RATE SELECTED
                              <1> ;@DSK_STATE  DB    ?           ; DRIVE 0 MEDIA STATE
                              <1> ;      DB    ?           ; DRIVE 1 MEDIA STATE
                              <1> ;      DB    ?           ; DRIVE 0 OPERATION START STATE
                              <1> ;      DB    ?           ; DRIVE 1 OPERATION START STATE
                              <1> ;@DSK_TRK    DB    ?           ; DRIVE 0 PRESENT CYLINDER
                              <1> ;      DB    ?           ; DRIVE 1 PRESENT CYLINDER
                              <1> 
                              <1> ;DATA        ENDS              ; END OF BIOS DATA SEGMENT
                              <1> 
                              <1> ;--------------------------------------------------------
                              <1> ;DRIVE TYPE TABLE                    :
```

```
                             <1> ;--------------------------------------------------------
                             <1>         ; 16/02/2015 (unix386.s, 32 bit modifications)
                             <1> DR_TYPE:
00006ADC 01                  <1>         DB    01           ;DRIVE TYPE, MEDIA TABLE
                             <1>                 ;DW      MD_TBL1
00006ADD [FA6A0000]          <1>         dd    MD_TBL1
00006AE1 82                  <1>         DB    02+BIT7ON
                             <1>         ;DW      MD_TBL2
00006AE2 [076B0000]          <1>                 dd      MD_TBL2
00006AE6 02                  <1> DR_DEFAULT:  DB    02
                             <1>                 ;DW      MD_TBL3
00006AE7 [146B0000]          <1>         dd      MD_TBL3
00006AEB 03                  <1>         DB    03
                             <1>                 ;DW      MD_TBL4
00006AEC [216B0000]          <1>         dd      MD_TBL4
00006AF0 84                  <1>         DB    04+BIT7ON
                             <1>                 ;DW      MD_TBL5
00006AF1 [2E6B0000]          <1>         dd      MD_TBL5
00006AF5 04                  <1>         DB    04
                             <1>                 ;DW      MD_TBL6
00006AF6 [3B6B0000]          <1>         dd      MD_TBL6
                             <1> DR_TYPE_E       equ $                   ; END OF TABLE
                             <1> ;DR_CNT      EQU   (DR_TYPE_E-DR_TYPE)/3
                             <1> DR_CNT       equ   (DR_TYPE_E-DR_TYPE)/5
                             <1> ;--------------------------------------------------------
                             <1> ;MEDIA/DRIVE PARAMETER TABLES             :
                             <1> ;--------------------------------------------------------
                             <1> ;--------------------------------------------------------
                             <1> ;360 KB MEDIA IN 360 KB DRIVE             :
                             <1> ;--------------------------------------------------------
                             <1> MD_TBL1:
00006AFA DF                  <1>  DB    11011111B   ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006AFB 02                  <1>  DB    2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006AFC 25                  <1>  DB    MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006AFD 02                  <1>  DB    2           ; 512 BYTES/SECTOR
00006AFE 09                  <1>  DB    09          ; EOT (LAST SECTOR ON TRACK)
00006AFF 2A                  <1>  DB    02AH        ; GAP LENGTH
00006B00 FF                  <1>  DB    0FFH        ; DTL
00006B01 50                  <1>  DB    050H        ; GAP LENGTH FOR FORMAT
00006B02 F6                  <1>  DB    0F6H        ; FILL BYTE FOR FORMAT
00006B03 0F                  <1>  DB    15          ; HEAD SETTLE TIME (MILLISECONDS)
00006B04 08                  <1>  DB    8           ; MOTOR START TIME (1/8 SECONDS)
00006B05 27                  <1>  DB    39          ; MAX. TRACK NUMBER
00006B06 80                  <1>  DB    RATE_250    ; DATA TRANSFER RATE
                             <1> ;--------------------------------------------------------
                             <1> ;360 KB MEDIA IN 1.2 MB DRIVE             :
                             <1> ;--------------------------------------------------------
                             <1> MD_TBL2:
00006B07 DF                  <1>  DB    11011111B   ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006B08 02                  <1>  DB    2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006B09 25                  <1>  DB    MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006B0A 02                  <1>  DB    2           ; 512 BYTES/SECTOR
00006B0B 09                  <1>  DB    09          ; EOT (LAST SECTOR ON TRACK)
00006B0C 2A                  <1>  DB    02AH        ; GAP LENGTH
00006B0D FF                  <1>  DB    0FFH        ; DTL
00006B0E 50                  <1>  DB    050H        ; GAP LENGTH FOR FORMAT
00006B0F F6                  <1>  DB    0F6H        ; FILL BYTE FOR FORMAT
00006B10 0F                  <1>  DB    15          ; HEAD SETTLE TIME (MILLISECONDS)
00006B11 08                  <1>  DB    8           ; MOTOR START TIME (1/8 SECONDS)
00006B12 27                  <1>  DB    39          ; MAX. TRACK NUMBER
00006B13 40                  <1>  DB    RATE_300    ; DATA TRANSFER RATE
                             <1> ;--------------------------------------------------------
                             <1> ;1.2 MB MEDIA IN 1.2 MB DRIVE             :
                             <1> ;--------------------------------------------------------
                             <1> MD_TBL3:
00006B14 DF                  <1>  DB    11011111B   ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006B15 02                  <1>  DB    2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006B16 25                  <1>  DB    MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006B17 02                  <1>  DB    2           ; 512 BYTES/SECTOR
00006B18 0F                  <1>  DB    15          ; EOT (LAST SECTOR ON TRACK)
00006B19 1B                  <1>  DB    01BH        ; GAP LENGTH
00006B1A FF                  <1>  DB    0FFH        ; DTL
00006B1B 54                  <1>  DB    054H        ; GAP LENGTH FOR FORMAT
00006B1C F6                  <1>  DB    0F6H        ; FILL BYTE FOR FORMAT
00006B1D 0F                  <1>  DB    15          ; HEAD SETTLE TIME (MILLISECONDS)
00006B1E 08                  <1>  DB    8           ; MOTOR START TIME (1/8 SECONDS)
00006B1F 4F                  <1>  DB    79          ; MAX. TRACK NUMBER
00006B20 00                  <1>  DB    RATE_500    ; DATA TRANSFER RATE
                             <1> ;--------------------------------------------------------
                             <1> ;720 KB MEDIA IN 720 KB DRIVE             :
                             <1> ;--------------------------------------------------------
                             <1> MD_TBL4:
00006B21 DF                  <1>  DB    11011111B   ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006B22 02                  <1>  DB    2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006B23 25                  <1>  DB    MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006B24 02                  <1>  DB    2           ; 512 BYTES/SECTOR
00006B25 09                  <1>  DB    09          ; EOT (LAST SECTOR ON TRACK)
00006B26 2A                  <1>  DB    02AH        ; GAP LENGTH
00006B27 FF                  <1>  DB    0FFH        ; DTL
00006B28 50                  <1>  DB    050H        ; GAP LENGTH FOR FORMAT
00006B29 F6                  <1>  DB    0F6H        ; FILL BYTE FOR FORMAT
00006B2A 0F                  <1>  DB    15          ; HEAD SETTLE TIME (MILLISECONDS)
```

```
00006B2B 08                   <1>  DB    8           ; MOTOR START TIME (1/8 SECONDS)
00006B2C 4F                   <1>  DB    79          ; MAX. TRACK NUMBER
00006B2D 80                   <1>  DB    RATE_250    ; DATA TRANSFER RATE
                              <1> ;--------------------------------------------------------
                              <1> ;720 KB MEDIA IN 1.44 MB DRIVE            :
                              <1> ;--------------------------------------------------------
                              <1> MD_TBL5:
00006B2E DF                   <1>  DB    11011111B   ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006B2F 02                   <1>  DB    2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006B30 25                   <1>  DB    MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006B31 02                   <1>  DB    2           ; 512 BYTES/SECTOR
00006B32 09                   <1>  DB    09          ; EOT (LAST SECTOR ON TRACK)
00006B33 2A                   <1>  DB    02AH        ; GAP LENGTH
00006B34 FF                   <1>  DB    0FFH        ; DTL
00006B35 50                   <1>  DB    050H        ; GAP LENGTH FOR FORMAT
00006B36 F6                   <1>  DB    0F6H        ; FILL BYTE FOR FORMAT
00006B37 0F                   <1>  DB    15          ; HEAD SETTLE TIME (MILLISECONDS)
00006B38 08                   <1>  DB    8           ; MOTOR START TIME (1/8 SECONDS)
00006B39 4F                   <1>  DB    79          ; MAX. TRACK NUMBER
00006B3A 80                   <1>  DB    RATE_250    ; DATA TRANSFER RATE
                              <1> ;--------------------------------------------------------
                              <1> ;1.44 MB MEDIA IN 1.44 MB DRIVE                 :
                              <1> ;--------------------------------------------------------
                              <1> MD_TBL6:
00006B3B AF                   <1>  DB    10101111B   ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
00006B3C 02                   <1>  DB    2           ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
00006B3D 25                   <1>  DB    MOTOR_WAIT  ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
00006B3E 02                   <1>  DB    2           ; 512 BYTES/SECTOR
00006B3F 12                   <1>  DB    18          ; EOT (LAST SECTOR ON TRACK)
00006B40 1B                   <1>  DB    01BH        ; GAP LENGTH
00006B41 FF                   <1>  DB    0FFH        ; DTL
00006B42 6C                   <1>  DB    06CH        ; GAP LENGTH FOR FORMAT
00006B43 F6                   <1>  DB    0F6H        ; FILL BYTE FOR FORMAT
00006B44 0F                   <1>  DB    15          ; HEAD SETTLE TIME (MILLISECONDS)
00006B45 08                   <1>  DB    8           ; MOTOR START TIME (1/8 SECONDS)
00006B46 4F                   <1>  DB    79          ; MAX. TRACK NUMBER
00006B47 00                   <1>  DB    RATE_500    ; DATA TRANSFER RATE
                              <1>
                              <1>
                              <1> ; << diskette.inc >>
                              <1> ; +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                              <1> ;
                              <1> ;----------------------------------------
                              <1> ;ROM BIOS DATA AREAS           :
                              <1> ;----------------------------------------
                              <1>
                              <1> ;DATA        SEGMENT AT 40H          ; ADDRESS= 0040:0000
                              <1>
                              <1> ;----------------------------------------
                              <1> ;FIXED DISK DATA AREAS         :
                              <1> ;----------------------------------------
                              <1>
                              <1> ;DISK_STATUS1:     DB    0           ; FIXED DISK STATUS
                              <1> ;HF_NUM:           DB    0           ; COUNT OF FIXED DISK DRIVES
                              <1> ;CONTROL_BYTE:     DB    0           ; HEAD CONTROL BYTE
                              <1> ;@PORT_OFF   DB    ?           ;  RESERVED (PORT OFFSET)
                              <1>
                              <1> ;----------------------------------------
                              <1> ;ADDITIONAL MEDIA DATA         :
                              <1> ;----------------------------------------
                              <1>
                              <1> ;@LASTRATE   DB    ?           ; LAST DISKETTE DATA RATE SELECTED
                              <1> ;HF_STATUS   DB    0           ; STATUS REGISTER
                              <1> ;HF_ERROR    DB    0           ; ERROR REGISTER
                              <1> ;HF_INT_FLAG DB    0           ; FIXED DISK INTERRUPT FLAG
                              <1> ;HF_CNTRL    DB    0           ; COMBO FIXED DISK/DISKETTE CARD BIT 0=1
                              <1> ;@DSK_STATE  DB    ?           ; DRIVE 0 MEDIA STATE
                              <1> ;      DB    ?           ; DRIVE 1 MEDIA STATE
                              <1> ;      DB    ?           ; DRIVE 0 OPERATION START STATE
                              <1> ;      DB    ?           ; DRIVE 1 OPERATION START STATE
                              <1> ;@DSK_TRK    DB    ?           ; DRIVE 0 PRESENT CYLINDER
                              <1> ;      DB    ?           ; DRIVE 1 PRESENT CYLINDER
                              <1>
                              <1> ;DATA        ENDS              ; END OF BIOS DATA SEGMENT
                              <1> ;
                              <1> ; +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                              <1>
                              <1> ERR_TBL:
00006B48 E0                   <1>  db    NO_ERR
00006B49 024001BB             <1>  db    BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
00006B4D 04BB100A             <1>  db    RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR
                              <1>
                              <1> ; 17/12/2014 (mov ax, [cfd])
                              <1> ; 11/12/2014
00006B51 00                   <1> cfd:         db 0              ; current floppy drive (for GET_PARM)
                              <1> ; 17/12/2014                   ; instead of 'DISK_POINTER'
00006B52 01                   <1> pfd:         db 1              ; previous floppy drive (for GET_PARM)
                              <1>                          ; (initial value of 'pfd
                              <1>                          ; must be different then 'cfd' value
                              <1>                          ; to force updating/initializing
                              <1>                          ; current drive parameters)
00006B53 90                   <1> align 2
```

```
                             <1>
00006B54 F001                <1> HF_PORT:     dw    1F0h  ; Default = 1F0h
                             <1>                          ; (170h)
00006B56 F603                <1> HF_REG_PORT: dw    3F6h  ; HF_PORT + 206h
                             <1>
                             <1> ; 05/01/2015
00006B58 00                  <1> hf_m_s:         db      0     ; (0 = Master, 1 = Slave)
                             <1>
                             <1> ; ****************************************************************************
                                 ;;;

                                 ; 27/08/2014
                                 scr_row:
00006B59 E0810B00                 dd 0B8000h + 0A0h + 0A0h + 0A0h ; Row 3
                                 scr_col:
00006B5D 00000000                 dd 0

                                 ;; 14/08/2015
                                 ;;msgPM:
                                 ;;      db "Protected mode and paging are ENABLED ... ", 0
                                 msgKVER:
00006B61 526574726F20554E49-      db "Retro UNIX 386 v1 - Kernel v0.2.0.16 [09/12/2015]", 0
00006B6A 582033383620763120-
00006B73 2D204B65726E656C20-
00006B7C 76302E322E302E3136-
00006B85 205B30392F31322F32-
00006B8E 3031355D00

00006B93 90                      Align 2

                                 ; 20/08/2014
                                   ; /* This is the default interrupt "handler" :-) */
                                   ; Linux v0.12 (head.s)
                                 int_msg:
00006B94 556E6B6E6F776E2069-      db "Unknown interrupt ! ", 0
00006B9D 6E7465727275707420-
00006BA6 212000

00006BA9 90                      Align 2

                                 ; 21/08/2014
                                 timer_msg:
00006BAA 49525120302028494E-      db "IRQ 0 (INT 20h) ! Timer Interrupt : "
00006BB3 542032306829202120-
00006BBC 54696D657220496E74-
00006BC5 657272757074203A20
                                 tcountstr:
00006BCE 303030303020             db "00000 "
00006BD4 00                       db 0

00006BD5 90                      Align 2
                                  ; 21/08/2014
                                 exc_msg:
00006BD6 435055206578636570-      db "CPU exception ! "
00006BDF 74696F6E202120
                                 excnstr:           ; 25/08/2014
00006BE6 3F3F68202045495020-      db "??h", "  EIP : "
00006BEF 3A20
                                 EIPstr: ; 29/08/2014
00006BF1 00<rept>                 times 12 db 0
                                 rtc_msg:
00006BFD 5265616C2054696D65-      db "Real Time Clock - "
00006C06 20436C6F636B202D20
                                 datestr:
00006C0F 30302F30302F303030-      db "00/00/0000"
00006C18 30
00006C19 20                       db " "
                                 daystr:
00006C1A 44415920                 db "DAY "
                                 timestr:
00006C1E 30303A30303A3030                db "00:00:00"
00006C26 20                       db " "
00006C27 00                       db 0

                                 daytmp:
                                  ; 28/02/2015
00006C28 3F3F3F2053554E204D-      db "??? SUN MON TUE WED THU FRI SAT "
00006C31 4F4E20545545205745-
00006C3A 442054485520465249-
00006C43 2053415420

00006C48 FF                      ptime_seconds: db 0FFh

                                  ; 23/02/2015
                                  ; 25/08/2014
                                 ;scounter:
                                 ;db 5
                                 ;db 19

                                 ; 05/11/2014
                                 msg_out_of_memory:
00006C49 070D0A                   db    07h, 0Dh, 0Ah
```

```
00006C4C 496E73756666696369-            db      'Insufficient memory ! (Minimum 2 MB memory is needed.)'
00006C55 656E74206D656D6F72-
00006C5E 7920212028 4D696E69-
00006C67 6D756D2032204D4220-
00006C70 6D656D6F7279206973-
00006C79 206E65656465642E29
00006C82 0D0A00                   db    0Dh, 0Ah, 0
                                  ;
                                 setup_error_msg:
00006C85 0D0A                     db 0Dh, 0Ah
00006C87 4469736B2053657475-      db 'Disk Setup Error!'
00006C90 70204572726F7221
00006C98 0D0A00                   db 0Dh, 0Ah,0

                                 ; 02/09/2014 (Retro UNIX 386 v1)
                                 ;crt_ulc : db 0 ; upper left column (for scroll)
                                 ;  db 0 ; upper left row (for scroll)

                                 ;crt_lrc : db 79 ; lower right column (for scroll)
                                 ;  db 24 ; lower right row (for scroll)


                                 ; 06/11/2014 (Temporary Data)
                                 ; Memory Information message
                                 ; 14/08/2015
                                 msg_memory_info:
00006C9B 07                       db    07h
00006C9C 0D0A                     db    0Dh, 0Ah
                                  ;db   "MEMORY ALLOCATION INFO", 0Dh, 0Ah, 0Dh, 0Ah
00006C9E 546F74616C206D656D-      db    "Total memory : "
00006CA7 6F7279203A20
                                 mem_total_b_str: ; 10 digits
00006CAD 303030303030303030-      db    "0000000000 bytes", 0Dh, 0Ah
00006CB6 302062797465730D0A
00006CBF 202020202020202020-      db    "               ", 20h, 20h, 20h
00006CC8 202020202020202020
                                 mem_total_p_str: ; 7 digits
00006CD1 303030303030302070-      db    "0000000 pages", 0Dh, 0Ah
00006CDA 616765730D0A
00006CE0 0D0A                     db    0Dh, 0Ah
00006CE2 46726565206D656D6F-      db    "Free memory  : "
00006CEB 727920203A20
                                 free_mem_b_str:  ; 10 digits
00006CF1 3F3F3F3F3F3F3F3F3F-      db    "?????????? bytes", 0Dh, 0Ah
00006CFA 3F2062797465730D0A
00006D03 202020202020202020-      db    "               ", 20h, 20h, 20h
00006D0C 202020202020202020
                                 free_mem_p_str:  ; 7 digits
00006D15 3F3F3F3F3F3F3F2070-      db    "??????? pages", 0Dh, 0Ah
00006D1E 616765730D0A
00006D24 0D0A00                   db    0Dh, 0Ah, 0

                                 dsk_ready_msg:
00006D27 0D0A                     db    0Dh, 0Ah
                                 dsktype:
00006D29 6664                     db    'fd'
                                 dskx:
00006D2B 30                       db    '0'
00006D2C 20                       db    20h
00006D2D 697320524541445920-      db    'is READY ...'
00006D36 2E2E2E
00006D39 00                       db    0
                                 nextline:
00006D3A 0D0A00                   db    0Dh, 0Ah, 0

                                 ; KERNEL - SYSINIT Messages
                                 ; 24/08/2015
                                 ; 13/04/2015 - (Retro UNIX 386 v1 Beginning)
                                 ; 14/07/2013
                                 ;kernel_init_err_msg:
                                 ;db 0Dh, 0Ah
                                 ;db 07h
                                 ;db 'Kernel initialization ERROR !'
                                 ;db 0Dh, 0Ah, 0
                                 ; 24/08/2015
                                 ;;; (temporary kernel init message has been removed
                                 ;;;  from 'sys_init' code)
                                 ;kernel_init_ok_msg:
                                 ;db 0Dh, 0Ah
                                 ;db 07h
                                 ;db 'Welcome to Retro UNIX 386 v1 Operating System !'
                                 ;db 0Dh, 0Ah
                                 ;       db 'by Erdogan Tan - 09/12/2015 (v0.2.0.16)'
                                 ;db 0Dh, 0Ah, 0
                                 panic_msg:
00006D3D 0D0A07                   db 0Dh, 0Ah, 07h
00006D40 4552524F523A204B65-      db 'ERROR: Kernel Panic !'
00006D49 726E656C2050616E69-
00006D52 632021
00006D55 0D0A00                   db 0Dh, 0Ah, 0
                                 etc_init_err_msg:
00006D58 0D0A                     db 0Dh, 0Ah
```

```
00006D5A 07                        db 07h
00006D5B 4552524F523A202F65-       db 'ERROR: /etc/init !?'
00006D64 74632F696E69742021-
00006D6D 3F
00006D6E 0D0A00                    db 0Dh, 0Ah, 0

                                  ; 10/05/2015
                                  badsys_msg:
00006D71 0D0A                      db 0Dh, 0Ah
00006D73 07                        db 07h
00006D74 496E76616C69642053-       db 'Invalid System Call !'
00006D7D 797374656D2043616C-
00006D86 6C2021
00006D89 0D0A                      db 0Dh, 0Ah
00006D8B 4541583A20                db 'EAX: '
                                  bsys_msg_eax:
00006D90 303030303030303068        db '00000000h'
00006D99 0D0A                      db 0Dh, 0Ah
00006D9B 4549503A20                db 'EIP: '
                                  bsys_msg_eip:
00006DA0 303030303030303068        db '00000000h'
00006DA9 0D0A00                    db 0Dh, 0Ah, 0

                                  BSYS_M_SIZE equ $ - badsys_msg


                                  align 2

                                  ; EPOCH Variables
                                  ; 13/04/2015 - Retro UNIX 386 v1 Beginning
                                  ; 09/04/2013 epoch variables
                                  ; Retro UNIX 8086 v1 Prototype: UNIXCOPY.ASM, 10/03/2013
                                  ;
00006DAC B207                     year:  dw 1970
00006DAE 0100                     month: dw 1
00006DB0 0100                     day:   dw 1
00006DB2 0000                     hour:  dw 0
00006DB4 0000                     minute: dw 0
00006DB6 0000                     second: dw 0

                                  DMonth:
00006DB8 0000                      dw 0
00006DBA 1F00                      dw 31
00006DBC 3B00                      dw 59
00006DBE 5A00                      dw 90
00006DC0 7800                      dw 120
00006DC2 9700                      dw 151
00006DC4 B500                      dw 181
00006DC6 D400                      dw 212
00006DC8 F300                      dw 243
00006DCA 1101                      dw 273
00006DCC 3001                      dw 304
00006DCE 4E01                      dw 334

                                  ; 04/11/2014 (Retro UNIX 386 v1)
00006DD0 0000                     mem_1m_1k:   dw 0  ; Number of contiguous KB between
                                                       ; 1 and 16 MB, max. 3C00h = 15 MB.
00006DD2 0000                     mem_16m_64k: dw 0  ; Number of contiguous 64 KB blocks
                                            ;   between 16 MB and 4 GB.

                                  ; 12/11/2014 (Retro UNIX 386 v1)
00006DD4 00                       boot_drv:    db 0 ; boot drive number (physical)
                                  ; 24/11/2014
00006DD5 00                       drv:        db 0
00006DD6 00                       last_drv:    db 0 ; last hdd
00006DD7 00                       hdc:         db 0  ; number of hard disk drives
                                              ; (present/detected)
                                  ;
                                  ; 24/11/2014 (Retro UNIX 386 v1)
                                  ; Physical drive type & flags
00006DD8 00                       fd0_type:    db 0  ; floppy drive type
00006DD9 00                       fd1_type:    db 0  ; 4 = 1.44 Mb, 80 track, 3.5" (18 spt)
                                              ; 6 = 2.88 Mb, 80 track, 3.5" (36 spt)
                                              ; 3 = 720 Kb, 80 track, 3.5" (9 spt)
                                              ; 2 = 1.2 Mb, 80 track, 5.25" (15 spt)
                                              ; 1 = 360 Kb, 40 track, 5.25" (9 spt)
00006DDA 00                       hd0_type:    db 0  ; EDD status for hd0 (bit 7 = present flag)
00006DDB 00                       hd1_type:    db 0  ; EDD status for hd1 (bit 7 = present flag)
00006DDC 00                       hd2_type:    db 0  ; EDD status for hd2 (bit 7 = present flag)
00006DDD 00                       hd3_type:    db 0  ; EDD status for hd3 (bit 7 = present flag)
                                              ; bit 0 - Fixed disk access subset supported
                                              ; bit 1 - Drive locking and ejecting
                                              ; bit 2 - Enhanced disk drive support
                                              ; bit 3 = Reserved (64 bit EDD support)
                                              ; (If bit 0 is '1' Retro UNIX 386 v1
                                              ; will interpret it as 'LBA ready'!)

                                  ; 11/03/2015 - 10/07/2015
00006DDE 000000000000000000-      drv.cylinders: dw 0,0,0,0,0,0,0
00006DE7 0000000000
00006DEC 000000000000000000-      drv.heads:     dw 0,0,0,0,0,0,0
00006DF5 0000000000
```

```
00006DFA 000000000000000000-     drv.spt:       dw 0,0,0,0,0,0,0,0
00006E03 0000000000
00006E08 000000000000000000-     drv.size:      dd 0,0,0,0,0,0,0,0
00006E11 000000000000000000-
00006E1A 000000000000000000-
00006E23 00
00006E24 00000000000000          drv.status:    db 0,0,0,0,0,0,0,0
00006E2B 00000000000000          drv.error:     db 0,0,0,0,0,0,0,0
                                 ;

00006E32 90<rept>                align 16

                                 bss_start:

                                 ABSOLUTE bss_start

                                  ; 11/03/2015
                                  ; Interrupt Descriptor Table (20/08/2014)
                                 idt:
00006E40 <res 00000200>           resb  64*8 ; INT 0 to INT 3Fh
                                 idt_end:

                                 ;alignb 4

                                 task_state_segment:
                                  ; 24/03/2015
00007040 <res 00000002>          tss.link:   resw 1
00007042 <res 00000002>               resw 1
                                 ; tss offset 4
00007044 <res 00000004>          tss.esp0:   resd 1
00007048 <res 00000002>          tss.ss0:    resw 1
0000704A <res 00000002>               resw 1
0000704C <res 00000004>          tss.esp1:   resd 1
00007050 <res 00000002>          tss.ss1:    resw 1
00007052 <res 00000002>               resw 1
00007054 <res 00000004>          tss.esp2:   resd 1
00007058 <res 00000002>          tss.ss2:    resw 1
0000705A <res 00000002>               resw 1
                                 ; tss offset 28
0000705C <res 00000004>          tss.CR3:    resd 1
00007060 <res 00000004>          tss.eip:    resd 1
00007064 <res 00000004>          tss.eflags: resd 1
                                 ; tss offset 40
00007068 <res 00000004>          tss.eax:    resd 1
0000706C <res 00000004>          tss.ecx:    resd 1
00007070 <res 00000004>          tss.edx:    resd 1
00007074 <res 00000004>          tss.ebx:    resd 1
00007078 <res 00000004>          tss.esp:    resd 1
0000707C <res 00000004>          tss.ebp:    resd 1
00007080 <res 00000004>          tss.esi:    resd 1
00007084 <res 00000004>          tss.edi:    resd 1
                                 ; tss offset 72
00007088 <res 00000002>          tss.ES:     resw 1
0000708A <res 00000002>               resw 1
0000708C <res 00000002>          tss.CS:    resw 1
0000708E <res 00000002>               resw 1
00007090 <res 00000002>          tss.SS:    resw 1
00007092 <res 00000002>               resw 1
00007094 <res 00000002>          tss.DS:    resw 1
00007096 <res 00000002>               resw 1
00007098 <res 00000002>          tss.FS:    resw 1
0000709A <res 00000002>               resw 1
0000709C <res 00000002>          tss.GS:    resw 1
0000709E <res 00000002>               resw 1
000070A0 <res 00000002>          tss.LDTR:   resw 1
000070A2 <res 00000002>               resw 1
                                 ; tss offset 100
000070A4 <res 00000002>               resw 1
000070A6 <res 00000002>          tss.IOPB:   resw 1
                                 ; tss offset 104
                                 tss_end:

000070A8 <res 00000004>          k_page_dir:  resd 1 ; Kernel's (System) Page Directory address
                                          ;   (Physical address = Virtual address)
000070AC <res 00000004>          memory_size: resd 1 ; memory size in pages
000070B0 <res 00000004>          free_pages:  resd 1 ; number of free pages
000070B4 <res 00000004>          next_page:   resd 1 ; offset value in M.A.T. for
                                          ;   first free page search
000070B8 <res 00000004>          last_page:   resd 1 ; offset value in M.A.T. which
                                          ;   next free page search will be
                                          ; stopped after it. (end of M.A.T.)
000070BC <res 00000004>          first_page:  resd 1 ;   offset value in M.A.T. which
                                          ; first free page search
                                          ;   will be started on it. (for user)
000070C0 <res 00000004>          mat_size:    resd 1 ; Memory Allocation Table size in pages

                                 ;;;
                                 ; 02/09/2014 (Retro UNIX 386 v1)
                                 ; 04/12/2013 (Retro UNIX 8086 v1)
000070C4 <res 00000002>          CRT_START:   resw 1        ; starting address in regen buffer
                                                ; NOTE: active page only
000070C6 <res 00000010>          cursor_posn: resw 8        ; cursor positions for video pages
```

```
                                 active_page:
000070D6 <res 00000001>          ptty:       resb 1        ; current tty
                                 ; 01/07/2015
000070D7 <res 00000001>          ccolor:     resb 1   ; current color attributes ('sysmsg')
                                 ; 26/10/2015
                                 ; 07/09/2014
000070D8 <res 00000014>          ttychr:      resw ntty+2  ; Character buffer (multiscreen)

                                 ; 21/08/2014
000070EC <res 00000004>          tcount:     resd 1

                                 ; 18/05/2015 (03/06/2013 - Retro UNIX 8086 v1 feature only!)
000070F0 <res 00000004>          p_time:      resd 1     ; present time (for systime & sysmdate)

                                 ; 18/05/2015 (16/08/2013 - Retro UNIX 8086 v1 feature only !)
                                 ; (open mode locks for pseudo TTYs)
                                 ; [ major tty locks (return error in any conflicts) ]
000070F4 <res 00000014>          ttyl:        resw ntty+2 ; opening locks for TTYs.

                                 ; 15/04/2015 (Retro UNIX 386 v1)
                                 ; 22/09/2013 (Retro UNIX 8086 v1)
00007108 <res 0000000A>          wlist:       resb ntty+2 ; wait channel list (0 to 9 for TTYs)
                                 ; 15/04/2015 (Retro UNIX 386 v1)
                                 ;; 12/07/2014 -> sp_init set comm. parameters as 0E3h
                                 ;; 0 means serial port is not available
                                 ;;comprm: ; 25/06/2014
00007112 <res 00000001>          com1p:       resb 1  ;;0E3h
00007113 <res 00000001>          com2p:       resb 1  ;;0E3h

                                 ; 17/11/2015
                                 ; request for response (from the terminal)
00007114 <res 00000002>          req_resp:     resw 1
                                 ; 07/11/2015
00007116 <res 00000001>          ccomport:    resb 1 ; current COM (serial) port
                                            ; (0= COM1, 1= COM2)
                                 ; 09/11/2015
00007117 <res 00000001>          comqr:      resb 1 ; 'query or response' sign (u9.s, 'sndc')
                                 ; 07/11/2015
00007118 <res 00000002>          rchar:      resw 1 ; last received char for COM 1 and COM 2
0000711A <res 00000002>          schar:      resw 1 ; last sent char for COM 1 and COM 2

                                 ; 23/10/2015
                                 ; SERIAL PORTS - COMMUNICATION MODES
                                 ; (Retro UNIX 386 v1 feature only!)
                                 ; 0 - command mode (default/initial mode)
                                 ; 1 - terminal mode (Retro UNIX 386 v1 terminal, ascii chars)
                                 ;;; communication modes for futre versions:
                                 ; // 2 - keyboard mode (ascii+scancode input)
                                 ; // 3 - mouse mode
                                 ; // 4 - device control (output) mode
                                 ; VALID COMMANDS for current version:
                                 ;      'LOGIN'
                                 ;  Login request: db 0FFh, 'LOGIN', 0
                                 ;  ("Retro UNIX 386 v1 terminal requests login")
                                 ;  Login response: db 0FFh, 'login', 0
                                 ;  ("login request accepted, wait for login prompt")
                                 ; When a login requests is received and acknowledged (by
                                 ; serial port interrupt handler (communication procedure),
                                 ; Retro UNIX 386 v1 operating system will start terminal mode
                                 ; (login procedure) by changing comm. mode to 1 (terminal mode)
                                 ; and then running 'etc/getty' for tty8 (COM1) or tty9 (COM2)
                                 ;
                                 ; 'sys connect' system call is used to change communication mode
                                 ; except 'LOGIN' command which is used to start terminal mode
                                 ; by using (COM port) terminal.

                                 ;com1own:     resb 1 ; COM1 owner (u.uno)
                                 ;com2own:     resb 1 ; COM2 owner (u.uno)
                                 ;com1mode:    resb 1 ; communication mode for COM1
                                 ;com1com:     resb 1 ; communication command for COM1
                                 ;com2mode:    resb 1 ; communication mode for COM1
                                 ;com2com      resb 1 ; communication command for COM1
                                 ;com1cbufp:   resb 8 ; COM1 command buffer char pointer
                                 ;com2cbufp:   resb 8 ; COM2 command buffer char pointer
                                 ;com1cbuf:    resb 8 ; COM2 command buffer
                                 ;com2cbuf:    resb 8 ; COM2 command buffer

                                 ; 22/08/2014 (RTC)
                                 ; (Packed BCD)
0000711C <res 00000001>          time_seconds: resb 1
0000711D <res 00000001>          time_minutes: resb 1
0000711E <res 00000001>          time_hours:   resb 1
0000711F <res 00000001>          date_wday:    resb 1
00007120 <res 00000001>          date_day:     resb 1
00007121 <res 00000001>          date_month:   resb 1
00007122 <res 00000001>          date_year:    resb 1
00007123 <res 00000001>          date_century: resb 1

                                 %include 'diskbss.inc'   ; UNINITIALIZED DISK (BIOS) DATA
                             <1> ; Retro UNIX 386 v1 Kernel - DISKBSS.INC
                             <1> ; Last Modification: 10/07/2015
                             <1> ;(Unnitialized Disk Parameters Data section for 'DISKIO.INC')
                             <1> ;
                             <1> ; ****************************************************************************
                             <1> 
                             <1> alignb 2
                             <1> 
                             <1> ;----------------------------------------
                             <1> ;TIMER DATA AREA        :
                             <1> ;----------------------------------------
                             <1> 
                             <1> TIMER_LH:    ; 16/02/205
00007124 <res 00000002>      <1> TIMER_LOW:      resw     1              ; LOW WORD OF TIMER COUNT
00007126 <res 00000002>      <1> TIMER_HIGH:     resw     1              ; HIGH WORD OF TIMER COUNT
00007128 <res 00000001>      <1> TIMER_OFL:      resb     1              ; TIMER HAS ROLLED OVER SINCE LAST READ
                             <1> 
                             <1> ;----------------------------------------
                             <1> ;DISKETTE DATA AREAS          :
                             <1> ;----------------------------------------
                             <1> 
00007129 <res 00000001>      <1> SEEK_STATUS: resb  1
0000712A <res 00000001>      <1> MOTOR_STATUS:resb  1
0000712B <res 00000001>      <1> MOTOR_COUNT: resb  1
0000712C <res 00000001>      <1> DSKETTE_STATUS:    resb  1
0000712D <res 00000007>      <1> NEC_STATUS:  resb  7
                             <1> 
                             <1> ;----------------------------------------
                             <1> ;ADDITIONAL MEDIA DATA        :
                             <1> ;----------------------------------------
                             <1> 
00007134 <res 00000001>      <1> LASTRATE:    resb  1
00007135 <res 00000001>      <1> HF_STATUS:   resb  1
00007136 <res 00000001>      <1> HF_ERROR:    resb  1
00007137 <res 00000001>      <1> HF_INT_FLAG: resb  1
00007138 <res 00000001>      <1> HF_CNTRL:    resb  1
00007139 <res 00000004>      <1> DSK_STATE:   resb  4
0000713D <res 00000002>      <1> DSK_TRK:     resb  2
                             <1> 
                             <1> ;----------------------------------------
                             <1> ;FIXED DISK DATA AREAS        :
                             <1> ;----------------------------------------
                             <1> 
0000713F <res 00000001>      <1> DISK_STATUS1:resb  1          ; FIXED DISK STATUS
00007140 <res 00000001>      <1> HF_NUM:      resb  1          ; COUNT OF FIXED DISK DRIVES
00007141 <res 00000001>      <1> CONTROL_BYTE:resb  1          ; HEAD CONTROL BYTE
                             <1> ;@PORT_OFF   resb  1          ; RESERVED (PORT OFFSET)
                             <1> ;port1_off   resb  1          ; Hard disk controller 1 - port offset
                             <1> ;port2_off   resb  1          ; Hard idsk controller 2 - port offset
                             <1> 
00007142 <res 00000002>      <1> alignb 4
                             <1> 
                             <1> ;HF_TBL_VEC: resd  1          ; Primary master disk param. tbl. pointer
                             <1> ;HF1_TBL_VEC:resd  1          ; Primary slave disk param. tbl. pointer
                             <1> HF_TBL_VEC: ; 22/12/2014
00007144 <res 00000004>      <1> HDPM_TBL_VEC:resd  1          ; Primary master disk param. tbl. pointer
00007148 <res 00000004>      <1> HDPS_TBL_VEC:resd  1          ; Primary slave disk param. tbl. pointer
0000714C <res 00000004>      <1> HDSM_TBL_VEC:resd  1          ; Secondary master disk param. tbl. pointer
00007150 <res 00000004>      <1> HDSS_TBL_VEC:resd  1          ; Secondary slave disk param. tbl. pointer
                             <1> 
                             <1> ; 03/01/2015
00007154 <res 00000001>      <1> LBAMode:     resb  1
                             <1> 
                             <1> ; ****************************************************************************

                                 ;;; Real Mode Data (10/07/2015 - BSS)

                                 ;alignb 2

                                 %include 'ux.s' ; 12/04/2015 (unix system/user/process data)
                             <1> ; Retro UNIX 386 v1 Kernel - ux.s
                             <1> ; Last Modification: 13/11/2015
                             <1> ;
                             <1> ; ///////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS //////////////
                             <1> ; (Modified from
                             <1> ;Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)
                             <1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))
                             <1> ; ----------------------------------------------------------------------------
                             <1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
                             <1> ; (Original) Source Code by Ken Thompson (1971-1972)
                             <1> ; <Bell Laboratories (17/3/1972)>
                             <1> ; <Preliminary Release of UNIX Implementation Document>
                             <1> ; (Section E10 (17/3/1972) - ux.s)
                             <1> ; ****************************************************************************
                             <1> 
00007155 <res 00000001>      <1> alignb 2
                             <1> 
                             <1> inode:
                             <1>  ; 11/03/2013.
                             <1>  ;Derived from UNIX v1 source code 'inode' structure (ux).
                             <1>  ;i.
                             <1> 
00007156 <res 00000002>      <1>  i.flgs:      resw 1
00007158 <res 00000001>      <1>  i.nlks:      resb 1
00007159 <res 00000001>      <1>  i.uid:       resb 1
```

```
0000715A <res 00000002>      <1>         i.size:  resw 1 ; size
0000715C <res 00000010>      <1>  i.dskp:      resw 8 ; 16 bytes
0000716C <res 00000004>      <1>  i.ctim:      resd 1
00007170 <res 00000004>      <1>  i.mtim:      resd 1
00007174 <res 00000002>      <1>  i.rsvd:  resw 1 ; Reserved (ZERO/Undefined word for UNIX v1.)
                             <1>
                             <1> I_SIZE equ $ - inode
                             <1>
                             <1> process:
                             <1>  ; 06/05/2015
                             <1>  ; 11/03/2013 - 05/02/2014
                             <1>  ;Derived from UNIX v1 source code 'proc' structure (ux).
                             <1>  ;p.
                             <1>
00007176 <res 00000020>      <1>         p.pid:   resw nproc
00007196 <res 00000020>      <1>         p.ppid:  resw nproc
000071B6 <res 00000020>      <1>         p.break: resw nproc
000071D6 <res 00000010>      <1>         p.ttyc:  resb nproc ; console tty in Retro UNIX 8086 v1.
000071E6 <res 00000010>      <1>  p.waitc: resb nproc ; waiting channel in Retro UNIX 8086 v1.
000071F6 <res 00000010>      <1>  p.link:      resb nproc
00007206 <res 00000010>      <1>  p.stat:      resb nproc
                             <1>
                             <1>  ; 06/05/2015 (Retro UNIX 386 v1 fetaure only !)
00007216 <res 00000040>      <1>  p.upage: resd nproc ; Physical address of the process's
                             <1>                  ; 'user' structure
                             <1>
                             <1>
                             <1> P_SIZE equ $ - process
                             <1>
                             <1>
                             <1> ; fsp table (original UNIX v1)
                             <1> ;
                             <1> ;Entry
                             <1> ;          15                                       0
                             <1> ;  1     |---|---------------------------------------|
                             <1> ;        |r/w|       i-number of open file           |
                             <1> ;        |---|---------------------------------------|
                             <1> ;        |               device number               |
                             <1> ;        |-------------------------------------------|
                             <1> ;    (*) | offset pointer, i.e., r/w pointer to file |
                             <1> ;        |-------------------------------------------|
                             <1> ;        |  flag that says    | number of processes  |
                             <1> ;        |   file deleted     | that have file open  |
                             <1> ;        |-------------------------------------------|
                             <1> ;  2     |                                           |
                             <1> ;        |-------------------------------------------|
                             <1> ;        |                                           |
                             <1> ;        |-------------------------------------------|
                             <1> ;        |                                           |
                             <1> ;        |-------------------------------------------|
                             <1> ;        |                                           |
                             <1> ;        |-------------------------------------------|
                             <1> ;  3     |                                           |
                             <1> ;        |                                           |
                             <1> ;
                             <1> ; (*) Retro UNIX 386 v1 modification: 32 bit offset pointer
                             <1>
                             <1>
                             <1> ; 15/04/2015
00007256 <res 000001F4>      <1> fsp:    resb nfiles * 10 ; 11/05/2015 (8 -> 10)
0000744A <res 00000018>      <1> bufp:   resd (nbuf+2) ; will be initialized
00007462 <res 00000002>      <1> ii:     resw 1
00007464 <res 00000002>      <1> idev:   resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
00007466 <res 00000002>      <1> cdev:    resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
                             <1> ; 18/05/2015
                             <1> ; 26/04/2013 device/drive parameters (Retro UNIX 8086 v1 feature only!)
                             <1> ; 'UNIX' device numbers (as in 'cdev' and 'u.cdrv')
                             <1> ;0 -> root device (which has Retro UNIX 8086 v1 file system)
                             <1> ;      1 -> mounted device (which has Retro UNIX 8086 v1 file system)
                             <1> ; 'Retro UNIX 8086 v1' device numbers: (for disk I/O procedures)
                             <1> ;0 -> fd0 (physical drive, floppy disk 1), physical drive number = 0
                             <1> ;1 -> fd1 (physical drive, floppy disk 2), physical drive number = 1
                             <1> ;2 -> hd0 (physical drive, hard disk 1), physical drive number = 80h
                             <1> ;3 -> hd1 (physical drive, hard disk 2), physical drive number = 81h
                             <1> ;4 -> hd2 (physical drive, hard disk 3), physical drive number = 82h
                             <1> ;5 -> hd3 (physical drive, hard disk 4), physical drive number = 83h
00007468 <res 00000001>      <1> rdev:   resb 1 ; root device number ; Retro UNIX 8086 v1 feature only!
                             <1>          ; as above, for physical drives numbers in following table
00007469 <res 00000001>      <1> mdev:   resb 1 ; mounted device number ; Retro UNIX 8086 v1 feature only!
                             <1> ; 15/04/2015
0000746A <res 00000001>      <1> active: resb 1
0000746B <res 00000001>      <1>   resb 1 ; 09/06/2015
0000746C <res 00000002>      <1> mnti:   resw 1
0000746E <res 00000002>      <1> mpid:   resw 1
00007470 <res 00000002>      <1> rootdir: resw 1
                             <1> ; 14/02/2014
                             <1> ; Major Modification: Retro UNIX 8086 v1 feature only!
                             <1> ;            Single level run queue
                             <1> ;            (in order to solve sleep/wakeup lock)
00007472 <res 00000002>      <1> runq:   resw 1
00007474 <res 00000001>      <1> imod:   resb 1
00007475 <res 00000001>      <1> smod:   resb 1
```

```
00007476 <res 00000001>      <1> mmod:   resb 1
00007477 <res 00000001>      <1> sysflg: resb 1
                             <1> 
                             <1> alignb 4
                             <1> 
                             <1> user:
                             <1>  ; 18/10/2015
                             <1>  ; 12/10/2015
                             <1>  ; 21/09/2015
                             <1>  ; 24/07/2015
                             <1>  ; 16/06/2015
                             <1>  ; 09/06/2015
                             <1>  ; 11/05/2015
                             <1>  ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit modifications)
                             <1>  ; 10/10/2013
                             <1>  ; 11/03/2013.
                             <1>  ;Derived from UNIX v1 source code 'user' structure (ux).
                             <1>  ;u.
                             <1> 
00007478 <res 00000004>      <1>  u.sp:   resd 1 ; esp (kernel stack at the beginning of 'sysent')
0000747C <res 00000004>      <1>  u.usp:        resd 1 ; esp (kernel stack points to user's registers)
00007480 <res 00000004>      <1>  u.r0:   resd 1 ; eax
00007484 <res 00000002>      <1>  u.cdir:       resw 1
00007486 <res 0000000A>      <1>  u.fp:   resb 10
00007490 <res 00000004>      <1>  u.fofp:       resd 1
00007494 <res 00000004>      <1>  u.dirp:       resd 1
00007498 <res 00000004>      <1>  u.namep:  resd 1
0000749C <res 00000004>      <1>  u.off:        resd 1
000074A0 <res 00000004>      <1>  u.base:       resd 1
000074A4 <res 00000004>      <1>  u.count:  resd 1
000074A8 <res 00000004>      <1>  u.nread:  resd 1
000074AC <res 00000004>      <1>  u.break:  resd 1 ; break
000074B0 <res 00000002>      <1>  u.ttyp:       resw 1
000074B2 <res 0000000A>      <1>  u.dirbuf: resb 10
                             <1>  ;u.pri:       resw 1 ; 14/02/2014
000074BC <res 00000001>      <1>  u.quant:  resb 1 ; Retro UNIX 8086 v1 Feature only ! (uquant)
000074BD <res 00000001>      <1>  u.pri:        resb 1 ;
000074BE <res 00000002>      <1>  u.intr:       resw 1
000074C0 <res 00000002>      <1>  u.quit:       resw 1
                             <1>  ;u.emt:       resw 1 ; 10/10/2013
000074C2 <res 00000002>      <1>  u.ilgins: resw 1
000074C4 <res 00000002>      <1>  u.cdrv:       resw 1 ; cdev
000074C6 <res 00000001>      <1>  u.uid:        resb 1 ; uid
000074C7 <res 00000001>      <1>  u.ruid:       resb 1
000074C8 <res 00000001>      <1>  u.bsys:       resb 1
000074C9 <res 00000001>      <1>  u.uno:        resb 1
000074CA <res 00000004>      <1>         u.upage:  resd 1 ; 16/04/2015 - Retro Unix 386 v1 feature only !
                             <1>  ; tty number (rtty, rcvt, wtty)
000074CE <res 00000001>      <1>  u.ttyn:       resb 1 ; 28/07/2013 - Retro Unix 8086 v1 feature only !
                             <1>  ; last error number
000074CF <res 00000004>      <1>  u.error:  resd 1 ; 28/07/2013 - 09/03/2015
                             <1>                   ; Retro UNIX 8086/386 v1 feature only!
000074D3 <res 00000004>      <1>  u.pgdir:  resd 1 ; 09/03/2015 (page dir addr of process)
000074D7 <res 00000004>      <1>  u.ppgdir: resd 1 ; 06/05/2015 (page dir addr of the parent process)
000074DB <res 00000004>      <1>  u.pbase:  resd 1 ; 20/05/2015 (physical base/transfer address)
000074DF <res 00000002>      <1>  u.pcount: resw 1 ; 20/05/2015 (byte -transfer- count for page)
                             <1>  ;u.pncount: resw 1
                             <1>        ; 16/06/2015 (byte -transfer- count for page, 'namei', 'mkdir')
                             <1>  ;u.pnbase:  resd 1
                             <1>        ; 16/06/2015 (physical base/transfer address, 'namei', 'mkdir')
                             <1>                 ; 09/06/2015
000074E1 <res 00000001>      <1>  u.kcall:  resb 1 ; The caller is 'namei' (dskr) or 'mkdir' (dskw) sign
000074E2 <res 00000001>      <1>  u.brwdev: resb 1 ; Block device number for direct I/O (bread & bwrite)
                             <1>               ; 24/07/2015 - 24/06/2015
                             <1>  ;u.args:  resd 1 ; arguments list (line) offset from start of [u.upage]
                             <1>               ; (arg list/line is from offset [u.args] to 4096 in [u.upage])
                             <1>               ; ([u.args] points to argument count -argc- address offset)
                             <1>               ; 24/06/2015
                             <1>  ;u.core:  resd 1 ; physical start address of user's memory space (for sys exec)
                             <1>  ;u.ecore: resd 1 ; physical end address of user's memory space (for sys exec)
                             <1>               ; 21/09/2015 (debugging - page fault analyze)
000074E3 <res 00000004>      <1>  u.pfcount: resd 1 ; page fault count for (this) process (for sys geterr)
                             <1> 
000074E7 <res 00000001>      <1> alignb 4
                             <1> 
                             <1> U_SIZE equ $ - user
                             <1> 
                             <1> ; 18/10/2015 - Retro UNIX 386 v1 (local variables for 'namei' and 'sysexec')
000074E8 <res 00000004>      <1> pcore:  resd 1 ; physical start address of user's memory space (for sys exec)
000074EC <res 00000004>      <1> ecore:  resd 1 ; physical start address of user's memory space (for sys exec)
000074F0 <res 00000004>      <1> nbase: resd 1      ; physical base address for 'namei' & 'sysexec'
000074F4 <res 00000002>      <1> ncount: resw 1     ; remain byte count in page for 'namei' & 'sysexec'
000074F6 <res 00000002>      <1> argc:  resw 1      ; argument count for 'sysexec'
000074F8 <res 00000004>      <1> argv:  resd 1      ; argument list (recent) address for 'sysexec'
                             <1> 
                             <1> ; 03/06/2015 - Retro UNIX 386 v1 Beginning
                             <1> ; 07/04/2013 - 31/07/2013 - Retro UNIX 8086 v1
000074FC <res 00000001>      <1> rw:     resb 1 ;; Read/Write sign (iget)
000074FD <res 00000001>      <1> rwdsk:  resb 1 ;; Read/Write function number (diskio) - 16/06/2015
000074FE <res 00000001>      <1> retry_count: resb 1 ; Disk I/O retry count - 11/06/2015
000074FF <res 00000001>      <1>   resb 1 ;; Reserved (16/06/2015)
                             <1> 
                             <1> ;alignb 4
                             <1>
                             <1> ; 22/08/2015
00007500 <res 00000820>      <1> buffer: resb nbuf * 520
                             <1>
00007D20 <res 00000008>      <1> sb0:   resd 2
                             <1> ;s:
                             <1> ; (root disk) super block buffer
                             <1> systm:
                             <1>  ; 13/11/2015 (Retro UNIX 386 v1)
                             <1>  ; 11/03/2013.
                             <1>  ;Derived from UNIX v1 source code 'systm' structure (ux).
                             <1>  ;s.
                             <1>
00007D28 <res 00000002>      <1>  resw 1
00007D2A <res 00000168>      <1>  resb 360 ; 2880 sectors ; original UNIX v1 value: 128
00007E92 <res 00000002>      <1>  resw 1
00007E94 <res 00000020>      <1>  resb 32      ; 256+40 inodes ; original UNIX v1 value: 64
00007EB4 <res 00000004>      <1>  s.time:      resd 1
00007EB8 <res 00000004>      <1>  s.syst:      resd 1
00007EBC <res 00000004>      <1>         s.wait_: resd 1 ; wait
00007EC0 <res 00000004>      <1>  s.idlet: resd 1
00007EC4 <res 00000004>      <1>  s.chrgt: resd 1
00007EC8 <res 00000002>      <1>  s.drerr: resw 1
                             <1>
                             <1> S_SIZE equ $ - systm
                             <1>
00007ECA <res 0000005E>      <1>  resb 512-S_SIZE ; 03/06/2015
                             <1>
00007F28 <res 00000008>      <1> sb1:   resd 2
                             <1> ; (mounted disk) super block buffer
                             <1> mount:
00007F30 <res 00000200>      <1>  resb 512  ; 03/06/2015
                             <1>
                             <1> ;/ ux -- unix
                             <1> ;
                             <1> ;systm:
                             <1> ;
                             <1> ;.=.+2
                             <1> ;.=.+128.
                             <1> ;.=.+2
                             <1> ;.=.+64.
                             <1> ;s.time: .=.+4
                             <1> ;s.syst: .=.+4
                             <1> ;s.wait: .=.+4
                             <1> ;s.idlet:.=.+4
                             <1> ;s.chrgt:.=.+4
                             <1> ;s.drerr:.=.+2
                             <1> ;inode:
                             <1> ;i.flgs: .=.+2
                             <1> ;i.nlks: .=.+1
                             <1> ;i.uid:  .=.+1
                             <1> ;i.size: .=.+2
                             <1> ;i.dskp: .=.+16.
                             <1> ;i.ctim: .=.+4
                             <1> ;i.mtim: .=.+4
                             <1> ;. = inode+32.
                             <1> ;mount:.=.+1024.
                             <1> ;proc:
                             <1> ;p.pid:  .=.+[2*nproc]
                             <1> ;p.dska: .=.+[2*nproc]
                             <1> ;p.ppid: .=.+[2*nproc]
                             <1> ;p.break:.=.+[2*nproc]
                             <1> ;p.link: .=.+nproc
                             <1> ;p.stat: .=.+nproc
                             <1> ;tty:
                             <1> ;. = .+[ntty*8.]
                             <1> ;fsp:  .=.+[nfiles*8.]
                             <1> ;bufp: .=.+[nbuf*2]+6
                             <1> ;sb0:  .=.+8
                             <1> ;sb1:  .=.+8
                             <1> ;swp:  .=.+8
                             <1> ;ii:   .=.+2
                             <1> ;idev: .=.+2
                             <1> ;cdev: .=.+2
                             <1> ;deverr: .=.+12.
                             <1> ;active: .=.+2
                             <1> ;rfap: .=.+2
                             <1> ;rkap: .=.+2
                             <1> ;tcap: .=.+2
                             <1> ;tcstate:.=.+2
                             <1> ;tcerrc: .=.+2
                             <1> ;mnti: .=.+2
                             <1> ;mntd: .=.+2
                             <1> ;mpid: .=.+2
                             <1> ;clockp: .=.+2
                             <1> ;rootdir:.=.+2
                             <1> ;toutt:.=.+16.
                             <1> ;touts: .=.+32.
                             <1> ;runq: .=.+6
                             <1> ;
                             <1> ;wlist:.=.+40.
                             <1> ;cc:   .=.+30.
                             <1> ;cf:   .=.+31.
                             <1> ;cl:   .=.+31.
                             <1> ;clist:.=.+510.
                             <1> ;imod: .=.+1
                             <1> ;smod: .=.+1
                             <1> ;mmod: .=.+1
                             <1> ;uquant: .=.+1
                             <1> ;sysflg: .=.+1
                             <1> ;pptiflg:.=.+1
                             <1> ;ttyoch: .=.+1
                             <1> ; .even
                             <1> ; .=.+100.; sstack:
                             <1> ;buffer: .=.+[ntty*140.]
                             <1> ; .=.+[nbuf*520.]
                             <1> ;
                             <1> ; . = core-64.
                             <1> ;user:
                             <1> ;u.sp:    .=.+2
                             <1> ;u.usp:   .=.+2
                             <1> ;u.r0:    .=.+2
                             <1> ;u.cdir:  .=.+2
                             <1> ;u.fp:    .=.+10.
                             <1> ;u.fofp:  .=.+2
                             <1> ;u.dirp:  .=.+2
                             <1> ;u.namep: .=.+2
                             <1> ;u.off:   .=.+2
                             <1> ;u.base:  .=.+2
                             <1> ;u.count: .=.+2
                             <1> ;u.nread: .=.+2
                             <1> ;u.break: .=.+2
                             <1> ;u.ttyp:  .=.+2
                             <1> ;u.dirbuf:.=.+10.
                             <1> ;u.pri:   .=.+2
                             <1> ;u.intr:  .=.+2
                             <1> ;u.quit:  .=.+2
                             <1> ;u.emt:   .=.+2
                             <1> ;u.ilgins:.=.+2
                             <1> ;u.cdev:  .=.+2
                             <1> ;u.uid:   .=.+1
                             <1> ;u.ruid:  .=.+1
                             <1> ;u.bsys:  .=.+1
                             <1> ;u.uno:   .=.+1
                             <1> ;. = core

                                 ;; Memory (swap) Data (11/03/2015)
                                 ; 09/03/2015
00008130 <res 00000002>          swpq_count: resw 1 ; count of pages on the swap que
00008132 <res 00000004>          swp_drv:    resd 1 ; logical drive description table address of the swap 
drive/disk
00008136 <res 00000004>          swpd_size:  resd 1 ; size of swap drive/disk (volume) in sectors (512 bytes).

0000813A <res 00000004>          swpd_free:  resd 1 ; free page blocks (4096 bytes) on swap disk/drive (logical)
0000813E <res 00000004>          swpd_next:  resd 1 ; next free page block
00008142 <res 00000004>          swpd_last:  resd 1 ; last swap page block

00008146 <res 00000002>          alignb 4

                                 ; 10/07/2015
                                 ; 28/08/2014
00008148 <res 00000004>          error_code:  resd 1
                                 ; 29/08/2014
0000814C <res 00000004>          FaultOffset: resd 1
                                 ; 21/09/2015
00008150 <res 00000004>          PF_Count:    resd 1      ; total page fault count
                                                    ; (for debugging - page fault analyze)
                                              ; 'page _fault_handler' (memory.inc)
                                              ; 'sysgeterr' (u9.s)
                                 ;; 21/08/2015
                                 ;;buffer: resb (nbuf*520) ;; sysdefs.inc, ux.s
                                 ;; ((NOTE: nbuf = 6, buffer r/w problem/bug here !? when nbuf > 4))

                                 bss_end:

                                 ; 27/12/2013
                                 _end:  ; end of kernel code (and read only data, just before bss)
```