

```

; *****
;
; BOOT1.ASM (Only for bootable UNIX v1 file system on 1.44 MB Floppy Disks)
; -----
;
; RETRO UNIX 8086 (Retro Unix == Turkish Rational Unix)
; Operating System Project (v0.1) by ERDOGAN TAN (Beginning: 11/07/2012)
; 1.44 MB Floppy Disk
; Bootable Unix (RUFFS) File System - UNIX Kernel Loader (Boot) File
; 07/03/2013
; Derived from UNIXCOPY.ASM (25/02/2013)
;
; [ Last Modification: 14/07/2013 ]
;
; Retro Unix is a derivation 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>
;
; *****

bsFSystemID      equ 2  ; 'RUFFS'
bsVolumeSerial   equ 6  ; (4 bytes)
bsFDSign         equ 10 ; 'fd'
bsDriveNumber    equ 12 ; fd0 or fd1 (0 or 1)
bsReserved       equ 13 ; 0 (512 bytes per sector)
bsSecPerTrack    equ 14 ; 18 (9 or 15)
bsHeads          equ 15 ; 2
bsTracks         equ 16 ; 80
bs_bf_inode_number equ 18 ; 0 or Boot/Startup File I-Number
bsInfoEndsign    equ 20 ; '@'

ROOT_DIR_INODE_NUMBER equ 41

kernel_loading_segment equ 1000h ; 09/07/2013
    ;; boot file space (segment 7E0h) = 33280 bytes,
    ;; kernel space (segment 1000h) = 65536 bytes
; kernel_loading_segment equ 1800h ; 05/03/2013

.8086

BOOT1  SEGMENT PUBLIC 'CODE'
    assume cs:BOOT1,ds:BOOT1,es:BOOT1,ss:BOOT1

START_CODE:

proc_start  proc near
    ; 07/03/2013 (timer)
    ; 06/03/2013
    ; 05/03/2013
    ; 01/03/2013
    ; 25/02/2013
    ; 24/02/2013 (BOOT1.ASM)
    ; 08/12/2012 (UNIXCOPY)
    ;
    ; 30/11/2012 (UNIXBOOT)
    ;
    mov ax, offset EndOfFile
    mov word ptr [BSBUFFER], ax
    add ax, 512
    mov word ptr [SUPERBLOCK], ax
    add ax, 512
    mov word ptr [DISKBUFFER], ax
    add ax, 512
    mov word ptr [FILEBUFFER], ax

loc_copy_bootsector:
    ; cli
    ;mov ax, cs
    ;mov sp, sizeoffile + 1000h
    ;mov ss, ax
    ;mov es, ax
    ;mov ds, ax
    ;sti
    ; cld
    ;xor cx, cx
    ;mov ds, cx
    mov ax, ds

```

```

        mov cx, 7C0h
        mov ds, cx
        xor si, si
        mov di, offset EndOfFile ; word ptr [BSBUFFER]
        mov cx, 256
        rep movsw

        mov ds, ax
        mov word ptr [EXTRA_SEGMENT], ax ; RESET ; 06/03/2013

;-----
; Read Superblock
;-----
; 25/02/2013
loc_read_superblock:
        ; DL = Drive number
        mov byte ptr [PhysicalDriveNumber], dl
        mov bx, word ptr [SUPERBLOCK]
        mov ax, 0201h ; Read 1 sector
        ;mov cx, 2 ; Read superblock
        mov cl, 2 ; 07/03/2013 (ch=0)
        xor dh, dh
        int 13h
        jnc short loc_unix_welcome

loc_drv_read_error:
        mov si, offset msg_unix_drv_read_error
        call UNIX_PRINTMSG

        xor ah, ah
        int 16h

        int 19h

;-----
; Write message
;-----

loc_unix_welcome:
        mov si, offset UNIX_Welcome
        call UNIX_PRINTMSG

;-----
; Timer
;-----

; 07/03/2013
; Automatic (default) kernel loading with timer tick count

        mov ax, ds ; cs
        mov cx, 40h
        mov es, cx
        mov bx, 6Ch
        mov cx, word ptr ES:[BX]
        add cx, 182*6 ; 60 seconds ?
        mov word ptr [waiting_count], cx
        mov es, ax

;-----
; call command interpreter
;-----

loc_call_unix_prompt:
        call unix_prompt

        int 19h

proc_start endp

```

```

UNIX_PRINTMSG proc near
    ; 20/01/2013 'call unix_printchr'

UNIX_PRINTMSG_LOOP:
    lodsb                    ; Load byte at DS:SI to AL
    and     AL,AL
    jz      short UNIX_PRINTMSG_OK
    mov     AH,0Eh
    mov     BX,07h
    int     10h              ; BIOS Service func ( ah ) = 0Eh
                                ; Write char as TTY
                                ; ↑AL-char BH-page BL-colo
    ;call    unix_printchr    ; 20/01/2013
    jmp     short UNIX_PRINTMSG_LOOP

UNIX_PRINTMSG_OK:
    retn

UNIX_PRINTMSG endp

;unix_printchr proc near
;    ; 20/01/2013
;    mov     AH,0Eh
;    mov     BX,07h
;    int     10h              ; BIOS Service func ( ah ) = 0Eh
;                                ; Write char as TTY
;                                ; ↑AL-char BH-page BL-color
;    retn
;unix_printchr endp

unix_prompt proc near
    ; 06/03/2013
    ; 05/03/2013 (default kernel name: unix)
    ; 25/02/2013 BOOT1 version
    ; 8/12/2012
    ; Derived from
    ; proc_dos_prompt procedure of TRDOS,
    ; MAINPROG.ASM (1/1/2012).
    ;
    ; proc_dos_prompt (15/09/2011)
    ;

    ;push ds
    ;pop  es

unix_prompt_1:
    mov     si, offset Boot_Msg
    call    unix_printmsg

unix_prompt_2:
    mov     ah, 03h
    ;mov     bx, 07h
    int     10h
    mov     byte ptr [CursorColumn],dl
    ;

unix_prompt_14: ; 07/03/2013
    ; automatic kernel loading timer
    ;
    cmp     byte ptr [def_kernel], bh ; 0
    ja      short unix_prompt_3

unix_prompt_15:
    hlt     ; halt cpu until external interrupt

    mov     ah, 1h ; Get keystroke status
    int     16h
    ; ZF = 0 if key pressed
    jnz     short unix_prompt_16

    mov     cx, 40h
    mov     ax, ds
    mov     bx, 6Ch ; mov si, 6Ch
    mov     es, cx
    mov     cx, word ptr ES:[BX] ; ES:[SI]
    mov     es, ax

    cmp     cx, word ptr [waiting_count]
    jb      short unix_prompt_15

    dec     byte ptr [def_kernel] ; FFh

```

```

        mov     si, offset UNIX_CRLF
        call    unix_printmsg

        mov     si, offset CommandBuffer
        mov     di, si
        jmp     short unix_prompt_17
unix_prompt_16:
        inc     byte ptr [def_kernel]
unix_prompt_3:
        mov     si, offset CommandBuffer
        call    proc_rw_char
        mov     di, si
        xor     bx, bx
        xor     cx, cx
unix_prompt_4:
        mov     al, byte ptr [SI][BX]
        inc     bl
        cmp     al, 20h
        ja      short unix_prompt_6
        jb      short unix_prompt_10
        cmp     bl, 74 ; 75 ?
        jb      short unix_prompt_4
unix_prompt_11:
        mov     bx, 07h
        mov     al, 0Dh
        mov     ah, 0Eh
        int     10h
        mov     al, 0Ah
        int     10h
        jmp     unix_prompt_1 ; loop
unix_prompt_5:
        mov     al, byte ptr [SI][BX]
        inc     bl
        cmp     al, 20h
        jna     short unix_prompt_7
unix_prompt_6:
        stosb
        inc     cl
        cmp     bl, 74 ; 75 ?
        jb      short unix_prompt_5
unix_prompt_7:
        xor     al, al ; 0
unix_prompt_8:
        mov     byte ptr [DI], al
        inc     di
        cmp     bl, 74 ; 75 ?
        jnb     short unix_prompt_9
        mov     al, byte ptr [SI][BX]
        inc     bl
        cmp     al, 20h
        jnb     short unix_prompt_8
        mov     byte ptr [DI], 0
unix_prompt_9:
        call    command_interpreter

        xor     al, al ; 07/03/2013

        cmp     byte ptr [unix_reboot], al ; 0
        ja      short unix_prompt_13 ; 06/03/2013
unix_prompt_12:
        mov     cx, 74 ; 75 ?
        mov     di, offset CommandBuffer
        xor     al, al
        rep     stosb
        jmp     short unix_prompt_11 ; 06/03/2013
unix_prompt_10:
        ; 05/03/2013
        xor     al, al
        cmp     byte ptr [def_kernel], al ; 0
        ja      short unix_prompt_11 ; 06/03/2013
        ;mov     di, offset CommandBuffer
unix_prompt_17: ; 07/03/2013 (timer code jumps here)
        mov     ax, 'nu'
        stosw
        mov     ax, 'xi'
        stosw
        xor     al, al

```

```

        stosb
        ;mov     cl, 4
        ;jmp     short unix_prompt_9
        call     loc_load_kernel ; jump/go to kernel
        jmp      short unix_prompt_12 ; error return only

unix_prompt_13: ; 06/03/2013
        retn

unix_prompt endp

proc_rw_char proc near
        ; 8/12/2012 (modification for UNIXCOPY.ASM)
        ; OUTPUT -> DS:SI = Entered String (ASCIIIZ)
        ;
read_next_char:
        xor      ah,ah
        int      16h
        and      al,al
        jz       short loc_arrow
        cmp      al,0E0h
        je       short loc_arrow
        cmp      al,08h
        jne      short char_return

loc_back:
        mov      bl,7
        mov      ah,3
        int      10h
        cmp      dl,byte ptr [CursorColumn]
        ja       short prev_column

loc_beep:
        mov      ah, 0Eh
        mov      al, 7
        int      10h
        jmp      short read_next_char

prev_column:
        dec      dl

set_cursor_pos:
        mov      ah,02h
        int      10h
        mov      bl, dl
        sub      bl,byte ptr [CursorColumn]
        mov      cx,1
        mov      ah,09h
        mov      al,20h
        mov      byte ptr [SI][BX],al

loc_write_it:
        mov      bl,7
        int      10h
        mov      dx,word ptr [CursorColumn]
        jmp      short read_next_char

loc_arrow:
        cmp      AH,4Bh
        je       short loc_back
        cmp      AH,53h
        je       short loc_back
        jmp      short read_next_char

char_return:
        mov      bl,7
        mov      ah,3
        int      10h

        mov      ah, dl
        sub      ah,byte ptr [CursorColumn]
        cmp      al,20h
        jnb      short loc_escape
        cmp      ah, 72 ; limit
        ja       short loc_beep

        mov      bl, ah
        xor      ah, ah
        mov      word ptr [SI][BX],ax
        mov      ah, 0Eh
        mov      bl, 7
        int      10h
        jmp      short read_next_char

```

```

pass_escape:
    cmp     al,0Dh
    jne     short read_next_char
    mov     ah, 0Eh
    mov     bl,7
    int     10h
    mov     al,0Ah
    int     10h
    retn

loc_escape:
    cmp     al,1Bh
    jne     short pass_escape
    stc
    retn

proc_rw_char endp

command_interpreter proc near
    ; 06/03/2013 (loc_load_kernel)
    ; 25/02/2013 BOOT1 version
    ; 23/02/2013 ?/help
    ; 17/02/2013 namei, inode, iget
    ; 16/02/2013 fs, volume
    ; 21/01/2013 'ls -l'
    ; 20/01/2013 ls (dir modifications)
    ; 13/01/2013 chmod, chown, link
    ; 07/01/2013 show tabspace (div) modif.
    ; 06/01/2013 show
    ; 06/01/2013 rm, mkdir, rmdir modifications
    ; 05/01/2013 check file attributes
    ; 30/12/2012
    ; 24/12/2012 todos
    ; 16/12/2012
    ; 08/12/2012
    ;
    lodsw

cl3:
    cmp     cl, 3
    jb     short cl2
    ja     cl5

; DIR
loc_cmd_dir:
    ; 05/01/2013 @b->@f, dir_print modifications
    cmp     ax, 'id'
    jne     loc_load_kernel ; @f
    lodsb
    cmp     al, 'r'
    jne     loc_load_kernel ; @f
    ;lods
    ;or     al, al
    ;jnz     loc_load_kernel ; @f
    mov     byte ptr [ls_option], 0
    inc     si

dir_getarg:
    ; 30/12/2012
    lodsb
    cmp     al, 20h
    je     short dir_getarg
    jnb     short dir_namei

ls_getarg3:
    xor     ax, ax
    jmp     short dir_print

dir_namei:
    ; 30/12/2012
    dec     si
    mov     word ptr [u_namep], si
    call    name_i
    jc     short ci_error
    ; ax = i-number

dir_print:
    call    print_directory_list
    jnc     short @f

ci_error:
    mov     si, offset error_msg
    call    unix_printmsg

@@:
    retn

```

```

; 23/02/2013
c11:
    cmp al, '?'
    jne loc_load_kernel ; @b
    ;cmp ah, 0
    ;jne loc_load_kernel ; @f

    mov si, offset Boot_Commands
    call UNIX_PRINTMSG

@@:
    retn

; 16/12/2012
c12:
    cmp cl, 2
    jb short c11 ; 23/02/2013
    ; jb @b

; LS (DIR)
loc_cmd_ls:
    ; 20/01/2013
    cmp ax, 'sl'
    jne short loc_cmd_cd ; 25/02/2013
    ;lods b
    ;or al, al
    ;jnz short loc_load_kernel ; @b
    mov byte ptr [ls_option], 1
    inc si
ls_getarg1:
    ; 21/01/2013
    lods b
    cmp al, 20h
    je short ls_getarg1
    jb short ls_getarg3

ls_getarg2:
    cmp al, '-'
    jne short dir_namei
    lods b
    cmp al, 'l'
    jne short ls_getarg3

ls_getarg4:
    lods b
    inc byte ptr [ls_option]
    cmp al, 20h
    je short dir_getarg
    jb short ls_getarg3
    dec byte ptr [ls_option]
    jmp short ls_getarg3

; CD (CHDIR)
loc_cmd_cd:
    cmp ax, 'dc'
    jne short loc_cmd_fs ; 25/02/2023
    ;lods b
    ;or al, al
    ;jnz short loc_load_kernel ; @f
    inc si
ci_cd_getarg:
    mov word ptr [u_namep], si
    lods b
    cmp al, 20h
    je short ci_cd_getarg
    jb short @f
    ; dec si

    mov ax, word ptr [u_namep]
    mov word ptr [arg], ax

    call sys_chdir
    jc ci_error

    mov si, word ptr [arg]
    call update_cdir_string

@@:
    retn

; FS (Volume) ; 16/02/2013 (File System / Volume Info)
loc_cmd_fs:
    cmp ax, 'sf'
    jne loc_load_kernel ; @b
    ;lods b
    ;or al, al
    ;jnz short loc_load_kernel ; @b

```

```

fs_info_print:
    call print_volume_info
@@:
    retn

c15:
    cmp cl, 5
    ja c16
    jb c14

; NAMEI          ; 17/02/2013, print i-number of file/directory
loc_cmd_namei:
    cmp ax, 'an'
    jne short loc_cmd_inode
    lodsw
    cmp ax, 'em'
    jne loc_load_kernel ; @f
    lodsb
    cmp al, 'i'
    jne loc_load_kernel ; @f
    ;lods
    ;or al, al
    ;jnz short loc_load_kernel ; @f
    inc si

namei_sf1:
    mov word ptr [u_namep], si
    lodsb
    cmp al, 20h
    je short namei_sf1
    jb short @f

namei_sf2:
    lodsb
    cmp al, 20h
    ja short namei_sf2
    dec si
    xor al, al
    mov byte ptr [SI], al

namei_fsf:
    call name_i
    jnc short namei_iget

namei_unix_stc:
    cmp ah, 0FFh
    jb ci_error
    mov si, offset NotFound_msg
    call UNIX_PRINTMSG

@@:
    retn

namei_iget:
    call i_get

namei_print_inum:
    jc ci_error
    mov cx, ax
    mov si, offset msgINumber
    call UNIX_PRINTMSG
    mov ax, cx
    mov cx, 3
    call print_decimal_number
    mov si, offset UNIX_CRLF
    call UNIX_PRINTMSG
    retn

; INODE          ; 17/02/2013, print inode structure/details
loc_cmd_inode:
    cmp ax, 'ni'
    jne short loc_load_kernel ; @b
    lodsw
    cmp ax, 'do'
    jne short loc_load_kernel ; @b
    lodsb
    cmp al, 'e'
    jne short loc_load_kernel ; @b
    ;lods
    ;or al, al
    ;jnz short loc_load_kernel ; @b
    inc si

```



```

inode_getarg1:
    mov bx, si
    lodsb
    cmp al, 20h
    je short inode_getarg1
    ja short inode_getarg2
    mov ax, word ptr [ii]
    jmp short @f

inode_getarg2:
    lodsb
    cmp al, 20h
    ja short inode_getarg2
    dec si
    xor ax, ax
    mov byte ptr [SI], al
    mov si, bx

@@:
    call show_inode
    jc ci_error

@@:
    retn

c14:
    ;cmp cl, 4
    ;jb c13

; SHOW
loc_cmd_show:
    ; 06/01/2013
    cmp ax, 'hs'
    jne short loc_load_kernel ; loc_cmd_unix ; 05/03/2013
    lodsw
    cmp ax, 'wo'
    jne short loc_load_kernel ; @b
    ;lods
    ;or al, al
    ;jnz short loc_load_kernel ; @b
    inc si

show_uf1:
    mov word ptr [u_namep], si
    lodsb
    cmp al, 20h
    je short show_uf1
    jb short @f

show_uf2:
    lodsb
    cmp al, 20h
    ja short show_uf2
    xor al, al
    mov byte ptr [SI]-1, al

show_uf3:
    call show_file
    jc ci_error

@@:
    retn

; UNIX (default kernel name) ; 06/03/2013
;loc_cmd_unix:
loc_load_kernel: ; 07/03/2013
    ; 06/03/2013
    mov word ptr [u_namep], offset CommandBuffer
    call load_kernel
    jnc short @f

    cmp byte ptr [def_kernel], 0FFh ; auto loading
    jb namei_unix_stc

    ; no error msg when it was auto kernel loading
    retn

@@:
    mov si, offset UNIX_CRLF
    call unix_printmsg

    ; 14/07/2013
    mov dl, byte ptr [PhysicalDriveNumber]
    xor dh, dh

    mov ax, kernel_loading_segment ; 1000h
    mov ds, ax
    mov es, ax

```

```

cli
mov ss, ax
mov sp, 32766 ; 09/07/2013
                ; FFFEH
sti

mov bx, offset EndOfFile ; Relocated BS buffer address

mov bp, sp
mov cx, cs ; 07/03/2013 (CX = Buffer segment)

; MASM.EXE don't accept
; jmp 1000h:0000h
; for OP Code: EA00000010
db 0EAh
dw 0
dw kernel_loading_segment ; 09/07/2013

cl6: ; 16/02/2013
    cmp cl, 6
    ja short cl8

; REBOOT      ; 25/02/201
loc_cmd_reboot:
    cmp ax, 'er'
    jne short loc_load_kernel ; @b
    lodsw
    cmp ax, 'ob'
    jne short loc_load_kernel ; @b
    lodsw
    cmp ax, 'to'
    jne short loc_load_kernel ; @b
    ;lodsb
    ;or al, al
    ;jnz short loc_load_kernel ; @f

    mov byte ptr [unix_reboot], 1

@@:
    retn

cl8:
    cmp cl, 8
    ja short @b ; bad command or file name
    jb short loc_load_kernel ; @b

; BOOTFILE
loc_cmd_bootfile:
    cmp ax, 'ob'
    jne short loc_load_kernel ; @b
    lodsw
    cmp ax, 'to'
    jne short loc_load_kernel ; @b
    lodsw
    cmp ax, 'if'
    jne short loc_load_kernel ; @b
    lodsw
    cmp ax, 'el'
    jne short loc_load_kernel ; @b
    ;lodsb
    ;or al, al
    ;jnz short loc_load_kernel ; @b

    mov si, word ptr [BSBuffer] ; 06/03/2013
    add si, bs_BF_inode_Number
    mov ax, word ptr [SI]
    ; and ax, ax
    ; jnz short @f
;ci_no_bootfile:
;    mov si, offset msg_Startup_File_Not_Exists
;    call UNIX_PRINTMSG
;    retn
@@:
    call find_bfn
    jc ci_error

ci_move_bfn_1:
    mov si, offset u_dirbuf + 2
    mov di, offset Boot_File_Name
    mov cx, 8

ci_move_bfn_2:
    lodsb

```

```

        and al, al
        jnz short @f
        mov byte ptr [DI], al ; 0
@@:
        stosb
        loop ci_move_bfn_2

        call proc_display_startupfile_info

        retn

command_interpreter endp

update_cdir_string proc near
    ; 13/01/2013 bugfix
    ; 10/12/2012
    ; 09/12/2012
    ; input -> SI= chdir argument
ucds_0:
    mov bx, offset unix_cdir
    inc bx ; 13/01/2013
    mov di, bx
    lodsb
    cmp al, '/'
    jne short @f
    xor dx, dx
    mov word ptr [CDirOffset], dx
    jmp short ucds_6
@@:
    mov dx, word ptr [CDirOffset]
    ; 13/01/2013
    or dx, dx
    jz short @f
    add di, dx
    mov byte ptr [DI], '/'
    inc di
    ;
    jmp short @f
ucds_8:
    inc di
ucds_6:
    lodsb
    cmp al, '/'
    je short ucds_6
@@:
    or al, al
    jz short ucds_5
    cmp al, '.'
    jne short ucds_3
    lodsb
    cmp al, '.'
    je short ucds_2 ; dotdot
ucds_1: ;dot
    cmp al, '/'
    je short ucds_6
    or al, al
    jz short ucds_5
    mov ah, '.'
    xchg ah, al
    stosw
    jmp short ucds_6
ucds_2: ; dotdot
    cmp di, bx
    ja short @f
    xor dx, dx
    mov byte ptr [DI], dl ; 0
    jmp short ucds_7
@@: ; 13/01/2013
    dec di
@@: ; move back
    dec di ; 13/01/2013
    mov al, byte ptr [DI]
    cmp al, '/'
    jne short @b ; 13/01/2013
    jmp short ucds_8

```

```

ucds_4:
    stosb
    jmp short ucds_6
ucds_3:
    stosb
    lodsb
    cmp al, '/'
    je short ucds_4
    and al, al
    jnz short ucds_3
ucds_5: ; 13/01/2013
    cmp di, bx
    jna short ucds_9
    dec di
    cmp byte ptr [DI], '/'
    je short ucds_9
    inc di
ucds_9:
    ; 13/01/2013
    mov byte ptr [DI], al ; 0
    mov dx, di
    sub dx, bx
ucds_7:
    mov word ptr [CDirOffset], dx

    retn

update_cdir_string endp

print_directory_list proc near
    ; 23/02/2013 long list printing (list_count)
    ; 03/02/2013
    ; 22/01/2013 ls -l command feature
    ; 21/01/2013 dir/ls options
    ; 20/01/2013 directory sign ("/")
    ; 30/12/2012
    or ax, ax ; i-number of directory
    jnz short @f

    ; 09/12/2012
pdl_0:
    mov ax, word ptr [u_cdir]
@@:
    call i_get
    jc short @f ; 20/01/2013 ; jc short pdl_9

    test word ptr [inode_flg], 4000h ; directory i-node ?
    jnz short pdl_2
pdl_1:
    mov ah, 0FFh ; error number
    stc
@@: ; 20/01/2013
    ; jmp short pdl_9
    retn
pdl_2:
    ; 25/02/2013
    mov si, offset unix_cdrv ; print current directory
    call unix_printmsg
    ;
    ; mov ax, word ptr [inode_size]
    ; mov word ptr [u_dirp], ax ; put size of directory in u.dirp

    ; xor ax, ax
    xor ah, ah
    mov word ptr [u_off], ax ; u.off is file offset used by user
    ; mov word ptr [u_fofp], offset u.off
    ; ; u.fofp is a pointer to the offset portion
    ; ; of fsp entry
    mov byte ptr [list_count], al ; 0 ; 23/02/2013
pdl_3:
    mov word ptr [u_base], offset u_dirbuf
    ; ; u.dirbuf holds a file name copied from
    ; ; a directory
    mov word ptr [u_count], 10
    ; ; u.dirbuff holds a file name copied from
    ; ; a directory
    mov ax, word ptr [ii]

```

```

    call read_i ; read 10 bytes of file with i-number (R1)
                ; i.e. read a directory entry
    jc short @b ; jc short pdl_9

    mov cx, word ptr [u_nread]
    or cx, cx
    jna short pdl_1 ; gives error return

    mov bx, word ptr [u_dirbuf]
    and bx, bx
    jz pdl_8
pdl_4:
    mov si, offset u_dirbuf + 2 ; r3, points to file name of directory entry
    mov cx, 8 ; max. file name length
    mov di, offset DirFileName + 1 ; boot_File_Name
pdl_5:
    lodsb ; mov al, byte ptr [SI], inc si
    or al, al
    jz short pdl_6 ; 3f. If char is nul, then the last char in string has
                ; been compared
    stosb ; mov byte ptr [DI], al, inc di
    loop pdl_5
pdl_6:
    ; 21/01/2013
    mov si, offset UNIX_CRLF
    call unix_printmsg
    cmp byte ptr [ls_option], 1
    je short pdl_7
    ;mov al, 0
    mov byte ptr [DI], al
    jb short pdl_13
pdl_7:
    ; 20/01/2013
    push di
    mov ax, word ptr [ii]
    mov word ptr [pdir], ax
    mov ax, word ptr [u_dirbuf]
    call i_get
    pop di
    jc pdl_9

    ; 22/01/2012
    cmp byte ptr [ls_option], 1
    jna short @f

pdl_11: ; 21/01/2013 ; Inode number
    mov ax, word ptr [u_dirbuf]
    mov cx, 3 ; 03/02/2013
    call print_decimal_number
    jmp short pdl_10
@@:
    mov ax, word ptr [inode_flgsl]
    test ah, 40h ; 'directory' flag
    jz short pdl_10

    mov si, offset u_dirbuf + 2
    lodsb
@@:
    cmp al, '.' ; '.'
    jne short @f
    lodsb
    or al, al
    jz short pdl_10
    jmp short @b
@@:
    mov al, '/'
    mov byte ptr [DI], al
    inc di
pdl_10:
    ; 21/03/2013
    xor al, al
    mov byte ptr [DI], al
pdl_13: ; File/Directory name
    inc byte ptr [list_count] ; 23/02/2013
    mov si, offset DirFileName
    call unix_printmsg

```

```

; 22/01/2013
cmp byte ptr [ls_option], 1
je pdl_12 ; 03/02/2013 short -> near
jb pdl_8 ; 23/02/2013

; 03/02/2013
@@: ; Owner (uid)
;xor bh, bh ; mov bh, 0
mov ah, 03h ; get cursor position and size.
int 10h
cmp dl, 13
jnb short @f
mov al, 20h
call putc
jmp short @b

@@:
xor ah, ah
mov al, byte ptr [inode_uid]
mov cx, 3
call print_decimal_number

@@:
mov al, 20h
call putc

mov al, 20h
call putc

@@: ; Flags/Attributes
mov dx, word ptr [inode_flg]
mov cl, '-'
shl dh, 1
shl dh, 1
jnc short @f
add al, 'd'-'-'

@@:
add al, cl
call putc
shl dl, 1
shl dl, 1
shl dl, 1
shl dl, 1
jnc short @f
add al, 'x'-'-'

@@:
add al, cl
call putc
shl dl, 1
jnc short @f
add al, 'r'-'-'

@@:
add al, cl
call putc
shl dl, 1
jnc short @f
add al, 'w'-'-'

@@:
add al, cl
call putc
shl dl, 1
jnc short @f
add al, 'r'-'-'

@@:
add al, '-'
call putc
shl dl, 1
jnc short @f
add al, 'w'-'-'

@@:
add al, cl
call putc

mov al, 20h
call putc

@@: ; File Size ; 03/02/2013
mov ax, word ptr [inode_size]
;mov cx, 5
mov cl, 5

```

```

        call print_decimal_number
@@:
        mov al, 20h
        call putc

        mov al, 20h
        call putc

@@: ; 03/02/2013 ; File creation date & time
        ;mov ax, word ptr [inode_ctim]
        ;mov dx, word ptr [inode_ctim]+2

        ; 23/02/2013 ; File last modification date & time
        mov ax, word ptr [inode_mtim]
        mov dx, word ptr [inode_mtim]+2

        call convert_from_epoch
        ; cx = day

        mov ax, cx ; word ptr [day]
        mov si, offset dec_num
        mov bx, si
        add bx, 2
        ; mov cx, 2
        mov cl, 2
        call proc_bin_to_decimal
        mov byte ptr [BX], '/'
        mov si, bx
        inc si
        mov ax, word ptr [month]
        ; mov cx, 2
        mov cl, 2
        call proc_bin_to_decimal
        add bx, 3
        mov byte ptr [BX], '/'
        mov si, bx
        inc si
        mov ax, word ptr [year]
        ;mov cx, 4
        mov cl, 4
        call proc_bin_to_decimal

        mov si, offset dec_num
        call unix_printmsg

        mov al, 20h
        call putc

        mov si, offset dec_num
        mov bx, si
        mov ax, word ptr [hour]
        ; mov cx, 2
        mov cl, 2
        call proc_bin_to_decimal
        add bx, 2
        mov byte ptr [BX], ':'

        mov si, bx
        inc si
        mov ax, word ptr [minute]
        ; mov cx, 2
        mov cl, 2
        call proc_bin_to_decimal
        add bx, 3
        ;mov byte ptr [BX], ':'
        ;mov si, bx
        ;inc si
        ;mov ax, word ptr [second]
        ;mov cx, 2
        ;mov cl, 2
        ;call proc_bin_to_decimal
        ;add bx,
        xor al, al
        mov byte ptr [BX], al

        mov si, offset dec_num
        call unix_printmsg

```

```

pdl_12:
    mov ax, word ptr [pdir]
    call i_get
    jc pdl_9
pdl_8:
    ; 30/12/2012
    mov ax, word ptr [u_off]
    cmp ax, word ptr [inode_size]
    jnb short @f ; 22/02/2013 ; jb pdl_3
    ; 23/02/2013
    cmp byte ptr [list_count], 21
    jb pdl_3
    xor ah, ah
    mov byte ptr [list_count], ah ; 0
    int 16h
    cmp al, 1Bh ; ESC key
    jne pdl_3
@@:
    mov si, offset UNIX_CRLF
    call unix_printmsg
pdl_9:
    retn

putc: ; 22/01/2013
    mov ah, 0Eh
    ;mov bx, 07h
    int 10h
    xor al, al

    retn

print_directory_list endp

sys_chdir proc near
    ; 09/12/2012 unixcopy.asm
    ; Retro UNIX v1 FS file import/export version
    ; of syschdir function
    ; Derived from (original) UNIX v1 source code
    ; PRELIMINARY release of Unix Implementation Document,
    ; 20/6/1972
    ;
    ; RETRO UNIX v1 FS
    ; syschdir:
    ; makes the directory specified in the argument
    ; the current directory

    ; mov word ptr [u_namep], si

syschdir_0:
    call name_i
    jc short syschdir_5

syschdir_1:
    call i_get
    jc short syschdir_5
syschdir_2:
    test word ptr [inode_flg], 4000h ; directory i-node ?
    jnz short syschdir_4
syschdir_3:
    mov ah, 0FFh
    stc
    retn
syschdir_4:
    mov word ptr [u_cdir], ax
    ; mov dx, word ptr [cdev]
    ; mov word ptr [u_cdev], dx

syschdir_5:
    retn

sys_chdir endp

```



```

show_file proc near
    ; 05/03/2013
    ; 07/01/2013
    ; 06/01/2013
    ; derived from TRDOS command interpreter file (CMDINTR.ASM)
    ; 'show' procedure (13/09/2011)

    call name_i
    jc short suf_4

    call i_get
    jc short suf_4

    test word ptr [inode_flg], 4000h ; Directory
    jnz short suf_4

    mov si, offset UNIX_CRLF
    call UNIX_PRINTMSG

    mov ax, word ptr [inode_size]

    mov dx, 512
    cmp ax, dx
    jna short suf_1

    mov ax, dx
suf_1:
    xor dx, dx
    mov word ptr [u_off], dx
    mov cx, 22
suf_2:
    push cx
    mov word ptr [u_count], ax
    mov ax, word ptr [FILEBUFFER]
    mov word ptr [u_base], ax
    mov ax, word ptr [ii] ; word ptr [u_dirbuf]
    call read_i
    pop cx
    jc short suf_4

    mov di, word ptr [u_nread]

    or di, di
    jz short suf_4

    mov si, word ptr [FILEBUFFER]

    jmp short suf_6
suf_3:
    and cx, cx
    jnz short suf_6
    xor ah, ah
    int 16h
    cmp al, 1Bh ; ESCAPE Key
    jne short suf_5
suf_4:
    mov si, offset UNIX_CRLF
    call UNIX_PRINTMSG

    retn
suf_5:
    mov cx, 20
suf_6:
    xor bh, bh ; mov bh, 0
    mov bl, 7

    lodsb
    cmp al, 0Dh ; ENTER/RETURN Char
    jne short suf_7
    dec cx
    jmp short suf_8
suf_7:
    cmp al, 09h ; TAB Space Char
    je short suf_10

```

```

suf_8:
    mov ah, 0Eh
    ;xor bh, bh ; mov bh, 0
    ;mov bl, 7
    int 10h
suf_9:
    dec di
    jnz short suf_3
    mov ax, word ptr [u_nread]
    jmp short suf_2
suf_10:
    push cx
    ;xor bh, bh ; mov bh, 0
    mov ah, 03h ; get cursor position and size.
    int 10h
    mov al, dl
    mov cx, 8
; suf_11a:
;     cmp al, cl
;     jb short suf_11b
;     sub al, cl
;     jmp short suf_11a
; suf_11b:
;     sub cl, al
suf_11:
    ; 07/01/2013
    xor ah, ah
    div cl
    sub cl, ah
    ;
    mov al, 20h
    mov ah, 0Eh
    ;mov bl, 7 ; char color attribute
suf_12:
    int 10h
    loop suf_12
    pop cx
    jmp short suf_9

show_file endp

name_i proc near
    ; 05/01/2013
    ; 09/12/2012 unixcopy.asm
    ;     Retro UNIX v1 FS file import/export version
    ; 31/10/2012
    ; 14/10/2012
    ; 07/10/2012
    ; Derived from (original) UNIX v1 source code
    ; PRELIMINARY release of Unix Implementation Document,
    ; 20/6/1972
    ;
    ; RETRO UNIX v1 FS
    ;
    ; return i-number of file (in AX)
    ;
    ; input:
    ; u_namep = pointer to file path name
    ; u_cdir = i-number of users directory
    ; ;u_cdev = device number
    ; output:
    ; cf= 0 -> no error, i-number in AX (R1)
    ; cf= 1 -> error code in AX
    ;

    mov si, word ptr [u_namep]

    cmp byte ptr [SI], '/' ; is first char in file name a /
    jne short @f
    mov ax, ROOT_DIR_INODE_NUMBER ; 41
        ; Put i-number of root directory in R1
    ; xor dx, dx
    inc si ; go to next char
    mov word ptr [u_namep], si
    jmp short namei_0

```

```

@@:
    ;mov dx, word ptr [u_cdev]
    mov ax, word ptr [u_cdir]
        ; put i-number of current directory in R1
namei_0:
    ;mov word ptr [cdev], dx
        ; device file for users directory into cdev
; 1
    cmp byte ptr [SI], 0 ; is the character in file name a nul
    jna short namei_7 ;nig

namei_1: ; 1
    ; get i-node with i-number r1
    call i_get
    jc short namei_7

    test word ptr [inode_flgsl], 4000h ; directory i-node ?
    ;jz short namei_6 ; got an error
    jnz short @f

;nib:
namei_6:
    mov ah, 0FFh ; Error code
    stc

;nig:
namei_7:
    retn
@@:
    mov ax, word ptr [inode_size]
    mov word ptr [u_dirp], ax ; put size of directory in u.dirp

    xor ax, ax
    mov word ptr [u_off], ax ; u.off is file offset used by user
    ;mov word ptr [u_fofp], offset u.off
        ; u.fofp is a pointer to the offset portion
        ; of fsp entry
namei_2: ; 2
    mov word ptr [u_base], offset u_dirbuf
        ; u.dirbuf holds a file name copied from
        ; a directory
    mov word ptr [u_count], 10

    mov ax, word ptr [ii]

    call read_i ; read 10 bytes of file with i-number (R1)
        ; i.e. read a directory entry
    jc short namei_7

    mov cx, word ptr [u_nread]

    or cx, cx
    jna short namei_6 ; nib ; gives error return

    mov bx, word ptr [u_dirbuf]
    and bx, bx
    jnz short namei_3 ; 3f. branch when active directory entry
        ; (i-node word in entry non zero)
    mov ax, word ptr [u_off]
    sub ax, 10
    mov word ptr [u_dirp], ax
    jmp short namei_2 ; 2b

namei_3: ; 3
    mov si, word ptr [u_namep] ; r2, u.namep points into a file name string
    mov di, offset u_dirbuf + 2 ; r3, points to file name of directory entry
    mov dx, offset u_dirbuf + 10
@@:
    ; 3
    lodsb ; mov al, byte ptr [SI], inc si (al = r4)
    or al, al
    jz short namei_4 ; 3f. If char is nul, then the last char in string has
        ; been compared
    cmp al, "/" ; is char a "/"
    je short namei_4 ; 3f
    cmp di, dx ; offset u_dirbuf + 10 ; r3,
        ; have i checked all 8 bytes of file name
    je short @b ; 3b

```

```

scasb                ; cmpb (r3)+, r4    (DI=R3, AL=R4)
                    ; compare char in u.namep string to file name char
                    ; read from
je short @b ; directory; brach if chars match

jmp short namei_2 ; 2b
                    ; File names do not match, go to next directory entry
namei_4: ; 3
    cmp di, dx ; offset u_dirbuf + 10 ; r3,
                    ; if equal all 8 bytes were matched
    je short namei_5 ; 3f

    mov ah, byte ptr [DI]
    ;inc di ; 05/01/2013
    and ah, ah ; tstb (r3)+, bne 2b
    jnz short namei_2 ; 2b

namei_5: ; 3
    mov word ptr [u_namep], si ; r2
                    ; u.namep points to char following a "/" or nul
    ;mov bx, word ptr [u_dirbuf] ; r1

    and al, al      ; r4. If r4=0 the end of file name reached,
                    ; if r4="/" then go to next directory
    mov ax, bx

    jnz namei_1 ; 1b

    retn

name_i  endp

read_i proc near
    ; 06/03/2013 (kernel loading segment)
    ; 05/03/2013
    ; 14/10/2012
    ; Boot sector version of "readi" procedure
    ; Derived from (original) UNIX v1 source code
    ; PRELIMINARY release of Unix Implementation Document,
    ; 20/6/1972
    ; AX (R1) = i-number
    ; RETRO UNIX v1 FS
    ; Boot sector version
    ;
    ; read from an i-node
    ;

    xor dx, dx ; 0
    mov word ptr [u_nread], dx ; accumulated number of bytes transmitted
    cmp word ptr [u_count], dx ; is number of byte to read greater than 0
    jna short read_inode_retn

read_inode_1:
    ; AX = I-Number
    push ax
    call i_get ; get i-node into i-node section of core
    mov dx, word ptr [inode_size] ; file size in bytes in r2 (DX)
    sub dx, word ptr [u_off] ; subtract file offset
    jna short read_inode_3
    cmp dx, word ptr [u_count]
                    ; are enough bytes left in file to carry out read
    jnb short read_inode_2
    mov word ptr [u_count], dx

read_inode_2:
    call m_get ; returns physical block number of block in file
                    ; where offset points
    ; AX = Physical block number
    call dsk_rd ; read in block, BX points to 1st word of data in
                    ; buffer
    jc short read_inode_3

```

```

readinode_sioreg:
    mov si, word ptr [u_off] ; R2
    mov cx, si ; cx = R3, si = R2
    or cx, 0FE00h ; set bits 9...15 of file offset in R3
    and si, 1FFh ; calculate file offset mod 512
    add si, bx ; word ptr [DISKBUFFER] ; si now points to 1st byte in buffer
                    ; where data is to be placed
    mov di, word ptr [u_base] ; R1
    neg cx ; 512 - file offset(mod512) in R3 (cx)
    cmp cx, word ptr [u_count]
    jna short @f ; 2f

    mov cx, word ptr [u_count]

@@:
    add word ptr [u_nread], cx ; r3 + number of bytes
                    ; xmitted during write is put into
                    ; u_nread
    sub word ptr [u_count], cx
    add word ptr [u_base], cx ; points to 1st of remaining
                    ; data bytes
    add word ptr [u_off], cx ; new file offset = number
                    ; of bytes done + old file offset

; end of readinode_sioreg

    ; DI = file (user data) offset
    ; SI = sector (I/O) buffer offset
    ; CX = byte count

    ; 06/03/2013
    mov ax, word ptr [EXTRA_SEGMENT] ; kernel loading segment or CS/DS
    mov es, ax

    rep movsb

    mov ax, ds ; 06/03/2013
    mov es, ax

    pop ax

    cmp word ptr [u_count], 0
    ja short read_inode_1

    retn

read_inode_3:
    pop ax ; i-number

read_inode_retn:
    retn

read_i endp

i_get    proc near
    ; 02/03/2013
    ; 24/02/2013 BOOT1 version
    ; 18/11/2012 unix boot file configuration version
    ; of "iget" procedure.
    ; 16/9/2012
    ; 14/7/2012
    ; Derived from (original) UNIX v1 source code
    ; PRELIMINARY release of Unix Implementation Document,
    ; 20/6/1972
    ; AX=R1
    ; RETRO UNIX v1 FS
    ; return => if cf=1 error number in [Error]

    cmp ax, word ptr [ii] ; AX (R1) = i-number of current file
    je short iget_4
iget_1:
    ; 24/02/2013
    ;mov dl, byte ptr [imod]
    ;and dl, dl ; has i-node of current file been modified ?
    ;jz short iget_2
    ;xor dl, dl ; mov al, 0
    ;mov byte ptr [imod], dl
    ;push ax

```

```

        ;mov ax, word ptr [ii]
        ;inc dl ; mov dl, 1
        ;; dl = 1 = write
        ;call i_calc
        ;pop dx
        ;jc short iget_4
        ;mov ax, dx
iget_2:
        and ax, ax
        jz short iget_3
        mov word ptr [ii], ax
        ;xor dl, dl ; 02/03/2013
        ; dl = 0 = read
        call i_calc
iget_3:
        mov ax, word ptr [ii]
iget_4:
        retn

i_get   endp

i_calc  proc near
        ; 05/03/2013
        ; 24/02/2013 BOOT1 version
        ; 18/11/2012 unix boot file configuration version
        ; of "icalc" procedure.
        ; 17/8/2012
        ; 14/7/2012
        ; Derived from (original) UNIX v1 source code
        ; PRELIMINARY release of Unix Implementation Document,
        ; 20/6/1972
        ;; AX=R1
        ; 0 = read, 1 = write
        ; RETRO UNIX v1 FS
        ;
        ; i-node is located in block (i+47)/16 and
        ; begins 32*(i+47) mod 16 bytes from its start
        ;; return => if cf=1 error number in [Error]

        ;; input -> dl = 0 -> read, 1 = Write

        ;mov byte ptr [rw], dl

        add ax, 47 ; add 47 to inode number
        push ax ; R1 -> -(SP)
        shr ax, 1 ; divide by 16
        shr ax, 1
        shr ax, 1
        shr ax, 1
                ; ax contains block number of block in which
                ; inode exists
        call dsk_rd
        pop dx
        jc short icalc_2

icalc_1:
        and dx, 0Fh      ; (i+47) mod 16
        shl dx, 1
        shl dx, 1
        shl dx, 1
        shl dx, 1
        shl dx, 1
                ; DX = 32 * ((i+47) mod 16)
                ; DX (R5) points to first word in i-node i.

        mov di, offset inode
                ; inode is address of first word of current inode
        mov cx, 16 ; CX = R3

        mov si, word ptr [DISKBUFFER]

        add si, dx

        ; copy new i-node into inode area of (core) memory
        rep movsw

```

```

icalc_2:
    retn

i_calc  endp

dsk_rd  proc near
    ; 06/03/2013
    ; 05/03/2013
    ; 28/11/2012 BugFix
    ; 20/10/2012 (buff_s)
    ; 14/10/2012
    ; fd boot sector version of "dskrd" procedure
    ; Derived from (original) UNIX v1 source code
    ; PRELIMINARY release of Unix Implementation Document,
    ; 20/6/1972
    ; RETRO UNIX v1 FS
    ; floppy disk boot sector version
    ; return => if cf=1 error number in [Error]

    ; ax = sector/block number

    mov bx, word ptr [DISKBUFFER]

    cmp ax, word ptr [buff_s] ; buffer sector
    je short dsk_rd_3

    mov si, ax

    xor ch, ch
    mov cl, 4 ; Retry count
dsk_rd_1:
    push cx
    mov dx, 18 ; Sectors per track
    div dl
    mov cl, ah ; Sector (zero based)
    inc cl ; To make it 1 based
    shr al, 1 ; Convert Track to Cylinder
    adc dh, 0 ; Heads (0 or 1)

    mov dl, byte ptr [PhysicalDriveNumber]
    mov ch, al

    mov ah, 2 ; 2=read
    mov al, 01h
    int 13h ; BIOS Service func ( ah ) = 2
    ; Read disk sectors
    ; BIOS Service func ( ah ) = 3
    ; Write disk sectors
    ; ↑AL-sec num CH-cyl CL-sec
    ; DH-head DL-drive ES:BX-buffer
    ; ↓CF-flag AH-stat AL-sec read

    pop cx
    jnc short dsk_rd_2
    loop dsk_rd_1
    retn ; 06/03/2013
dsk_rd_2:
    mov word ptr [buff_s], si
dsk_rd_3:
    retn

dsk_rd  endp

```

```

m_get    proc near
; 05/03/2013
; 03/03/2013
; 01/03/2013
; 24/02/2013 BOOT1 version
; 18/11/2012
; 14/11/2012 unix boot file configuration version
; of "mget" procedure
; 31/10/2012
; 20/10/2012
; 19/8/2012
; 13/8/2012
; 27/7/2012
; 21/7/2012
; Derived from (original) UNIX v1 source code
; PRELIMINARY release of Unix Implementation Document,
; 20/6/1972
;; return -> AX=R1
; RETRO UNIX v1 FS
; initialization/format version
; cf -> 1 = error (no free block)

; ; contents of bx, cx, dx will be destroyed
mget_0:
mov bl, byte ptr [u_off]+1
xor bh, bh
; BX = R2
test word ptr [inode_flg], 4096 ; 1000h
; is this a large or small file
jnz short mget_4 ; 4f ; large file

test bl, 0F0h ; !0Fh ; error if BX (R2) >= 16
jnz short mget_2 ; 28/02/2013

and bl, 0Eh ; clear all bits but bits 1,2,3
mov ax, word ptr inode_dskp[BX] ; AX = R1, physical block number
mget_3:
; 24/02/2013
cmp ax, 1

mget_1: ; 2
retn

mget_4: ; 4 ; large file
; 05/03/2013
and bl, 0FEh
push bx
; 01/03/2013 Max. possible BX (offset) value is 127 (65535/512)
; for this file system (offset 128 to 255 not in use)

; There is always 1 indirect block for this file system
mov ax, word ptr [inode_dskp] ; inode_dskp[0]

or ax, ax
jnz short mget_5
; 28/02/2013
mget_2:
stc
mget_7:
pop bx
retn

mget_5: ;2
; ax = R1, block number
call dsk_rd ; read indirect block
jc short mget_7
mget_6:
pop ax ; R2, get offset
add bx, ax ; first word of indirect block
mov ax, word ptr [BX] ; put physical block no of block
; in file sought in R1 (AX)
jmp short mget_3 ; 24/02/2013

m_get endp

```



```

convert_from_epoch proc near
    ; 30/11/2012
    ; Derived from DALLAS Semiconductor
    ; Application Note 31 (DS1602/DS1603)
    ; 6 May 1998
    ;
    ; INPUT:
    ; DX:AX = Unix (Epoch) Time
    mov cx, 60
    call proc_div32
    ;mov word ptr [imin], ax    ; whole minutes
    ;mov word ptr [imin]+2, dx ; since 1/1/1970
    mov word ptr [second], bx  ; leftover seconds
    ; mov cx, 60
    call proc_div32
    ;mov word ptr [ihrs], ax    ; whole hours
    ;mov word ptr [ihrs]+2, dx ; since 1/1/1970
    mov word ptr [minute], bx  ; leftover minutes
    ; mov cx, 24
    mov cl, 24
    call proc_div32
    ;mov word ptr [iday], ax    ; whole hours
                                ; since 1/1/1970
    ; mov word ptr [iday]+2, dx ; DX = 0
    mov word ptr [hour], bx    ; leftover hours
    add ax, 365+366            ; whole day since
                                ; 1/1/1968
    ; adc dx, 0                ; DX = 0
    ; mov word ptr [iday], ax
    push ax
    mov cx, (4*365)+1          ; 4 years = 1461 days
    call proc_div32
    pop cx
    ;mov word ptr [lday], ax    ; count of quadyrs (4 years)
    push bx
    ;mov word ptr [qday], bx    ; days since quadyr began
    cmp bx, 31 + 29            ; if past feb 29 then
    cmc                        ; add this quadyr's leap day
    adc ax, 0                  ; to # of qadyrs (leap days)
    ;mov word ptr [lday], ax    ; since 1968
    ;mov cx, word ptr [iday]
    xchg cx, ax                ; CX = lday, AX = iday
    sub ax, cx                  ; iday - lday
    mov cx, 365
    ;xor dx, dx                ; DX = 0
    ; AX = iday-lday, DX = 0
    call proc_div32
    ;mov word ptr [iyrs], ax    ; whole years since 1968
    ; jday = iday - (iyrs*365) - lday
    ;mov word ptr [jday], bx    ; days since 1/1 of current year
    add ax, 1968                ; compute year
    mov word ptr [year], ax
    mov dx, ax
    ;mov ax, word ptr [qday]
    pop ax
    cmp ax, 365                ; if qday <= 365 and qday >= 60
    ja short @f                ; jday = jday + 1
    cmp ax, 60                 ; if past 2/29 and leap year then
    cmc                        ; add a leap day to the # of whole
    adc bx, 0                  ; days since 1/1 of current year
@@:
    ;mov word ptr [jday], bx
    mov cx, 12                 ; estimate month
    xchg cx, bx                ; CX = jday, BX = month
    mov ax, 366                ; mday, max. days since 1/1 is 365
    and dx, 11b                ; year mod 4 (and dx, 3)
@@:
    ; Month calculation        ; 0 to 11 (11 to 0)
    cmp cx, ax                  ; mday = # of days passed from 1/1
    jnb short @f
    dec bx                      ; month = month - 1
    shl bx, 1
    mov ax, word ptr DMonth[BX] ; # elapsed days at 1st of month
    shr bx, 1                  ; bx = month - 1 (0 to 11)
    cmp bx, 1                  ; if month > 2 and year mod 4 = 0
    jna short @b                ; then mday = mday + 1
    or dl, dl                  ; if past 2/29 and leap year then
    jnz short @b                ; add leap day (to mday)
    inc ax                      ; mday = mday + 1

```

```

        jmp short @b
@@:
        inc bx                ; -> bx = month, 1 to 12
        mov word ptr [month], bx
        sub cx, ax            ; day = jday - mday + 1
        inc cx
        mov word ptr [day], cx

        ; ax, bx, cx, dx is changed at return
        ; output ->
        ; [year], [month], [day], [hour], [minute], [second]
        ;

        retn

convert_from_epoch endp

```

```
proc_mul32 proc near
```

```

        ; push cx

        mov cx, bx
        mov bx, dx

        mul cx

        xchg ax, bx

        push dx

        mul cx

        pop cx

        add ax, cx
        adc dx, 0

        xchg bx, ax
        xchg dx, bx

        ; pop cx

        retn

```

```
proc_mul32 endp
```

```
proc_div32 proc near
```

```

        ; 1999
        ; (Rx_Dos_Div32) 32 bit divide procedure
        ; by Erdogan Tan
        ; Input -> DX_AX = 32 bit dividend
        ;         CX = 16 bit divisor
        ; output -> DX_AX = 32 bit quotient
        ;         BX = 16 bit remainder
        mov bx, dx
        xchg ax, bx
        xor dx, dx
        div cx                ; at first, divide DX
        xchg ax, bx           ; remainder is in DX
                                ; now, BX has quotient
                                ; save remainder
        div cx                ; so, DX_AX divided and
                                ; AX has quotient
                                ; DX has remainder
        xchg dx, bx           ; finally, BX has remainder

        retn

```

```
proc_div32 endp
```

```

find_bfn proc near
    ; 26/11/2012
    ; 25/11/2012
    ;
    ; find boot file name by i-number (ax)
    ;
    ; cf -> 1 means error, ax = 0 -> not found

    mov word ptr [uf_i_number], ax
    push si

    mov ax, ROOT_DIR_INODE_NUMBER ; 41
    call i_get
    jc short loc_find_bfn_retn

    ; test word ptr [inode_flg], 4000h ; directory i-node ?
    ; jnz short @f

    ; mov ah, 0FFh ; error number
    ; stc
    ; jmp short loc_find_bfn_retn
;:@:
    xor ax, ax
    mov word ptr [u_off], ax ; u_off is file offset used by user

loc_find_bfn_1:
    mov word ptr [u_base], offset u_dirbuf
                                ; u.dirbuff holds a file name copied from
                                ; a directory
    mov word ptr [u_count], 10

    mov ax, ROOT_DIR_INODE_NUMBER

    call read_i ; read 10 bytes of file with i-number
                ; i.e. read a directory entry
    jc short loc_find_bfn_retn

    mov ax, word ptr [u_nread]

    or ax, ax
    jz short loc_find_bfn_2 ; gives error return

    mov ax, word ptr [u_dirbuf]

    cmp ax, word ptr [uf_i_number] ; Check i-number of directory entry
    jne short loc_find_bfn_1       ; if same with specified uf_i_number
                                    ; it is the boot file

loc_find_bfn_3:
    call i_get
loc_find_bfn_retn:
    pop si
    retn

loc_find_bfn_2:
    stc
    jmp short loc_find_bfn_retn

find_bfn endp

```

```

proc_display_startupfile_info proc near
    ; 06/03/2013
    ; 30/11/2012
    ; 29/11/2012 ; @@
    ; 25/11/2012

    mov si, offset Msg_StartupFile_Name
    call UNIX_PRINTMSG

    mov si, offset Boot_File_Name
    call UNIX_PRINTMSG

    mov si, offset Str_Inode_Number
    call UNIX_PRINTMSG

    mov si, word ptr [BSBuffer] ; 06/03/2013
    add si, bs_bf_inode_number
    mov ax, word ptr [SI]

    mov si, offset Decimal_i_no_str
    mov cx, 5
    call proc_bin_to_decimal

    mov si, offset Decimal_i_no_str

    mov cx, 4
@@:
    cmp byte ptr [SI], '0'
    ja short @f
    inc si
    loop @b
@@:
    call UNIX_PRINTMSG

    mov si, offset Str_startup_file_size
    call UNIX_PRINTMSG

    mov ax, word ptr [Inode_size]
    mov si, offset Decimal_size_str
    ;mov cx, 5
    mov cx, 5
    mov cl, 5
    call proc_bin_to_decimal

    mov si, offset Decimal_size_str

    mov cl, 4
@@:
    cmp byte ptr [SI], '0'
    ja short @f
    inc si
    loop @b
@@:
    call UNIX_PRINTMSG

    mov si, offset Str_Bytes
    call UNIX_PRINTMSG

    ; 30/11/2012

    mov ax, word ptr [Inode_ctim]
    mov dx, word ptr [Inode_ctim]+2

    call convert_from_epoch

    mov ax, word ptr [year]
    mov si, offset str_cyear
    ;mov cx, 4
    mov cx, 4
    mov cl, 4
    call proc_bin_to_decimal

    mov ax, word ptr [month]
    mov si, offset str_cmonth
    mov cl, 2
    call proc_bin_to_decimal

    mov ax, word ptr [day]
    mov si, offset str_cday
    mov cl, 2

```

```

    call proc_bin_to_decimal

    mov ax, word ptr [hour]
    mov si, offset str_chour
    mov cl, 2
    call proc_bin_to_decimal

    mov ax, word ptr [minute]
    mov si, offset str_cminute
    mov cl, 2
    call proc_bin_to_decimal

    mov ax, word ptr [second]
    mov si, offset str_csecond
    mov cl, 2
    call proc_bin_to_decimal

    mov ax, word ptr [Inode_mtim]
    mov dx, word ptr [Inode_mtim]+2

    call convert_from_epoch

    mov ax, word ptr [year]
    mov si, offset str_myear
    ;mov cx, 4
    mov cl, 4
    call proc_bin_to_decimal

    mov ax, word ptr [month]
    mov si, offset str_mmonth
    mov cl, 2
    call proc_bin_to_decimal

    mov ax, word ptr [day]
    mov si, offset str_mday
    mov cl, 2
    call proc_bin_to_decimal

    mov ax, word ptr [hour]
    mov si, offset str_mhour
    mov cl, 2
    call proc_bin_to_decimal

    mov ax, word ptr [minute]
    mov si, offset str_mminute
    mov cl, 2
    call proc_bin_to_decimal

    mov ax, word ptr [second]
    mov si, offset str_msecond
    mov cl, 2
    call proc_bin_to_decimal

    mov si, offset Str_SF_date_Time
    call UNIX_PRINTMSG

    retn

proc_display_startupfile_info endp

proc_bin_to_decimal proc near
    ; 30/11/2012 (CX input)
    ; 25/11/2012 unixboot.asm version
    ; 6-5-2009
    ; Erdogan Tan
    ; INPUT: DS:SI = Target location
    ;        AX = Binary Number
    ;        CX = Number of digits
    ; OUTPUT: Decimal chars at DS:SI
    ; CX, AX, DX will be changed.

    ;push bp
    ;push si
loc_reset_str_NumberInput:
    mov byte ptr [SI], "0"
    inc si
    loop loc_reset_str_NumberInput

```

```

        mov bp, sp
        xor dx, dx
        mov cx, 10
loc_rediv_NumberInput:
        div cx
        add dl, '0'
        push dx
        xor dx, dx
        dec si
        or ax, ax
        jnz short loc_rediv_NumberInput
loop_popcx_NumberInput:
        pop dx
        mov byte ptr [SI], dl
        inc si
        cmp bp, sp
        jne short loop_popcx_NumberInput
        ;pop si
        ;pop bp

        retn

proc_bin_to_decimal endp

print_decimal_number proc near
    ; 03/02/2013
    ; 21/01/2013
    ; print decimal number
    ;
    ; INPUT -> AX = Integer
    ; 32/02/2013 CX = Number of decimal digits
    ; OUTPUT -> decimal number as string
pdn0:
    mov si, offset dec_num
    ;
    mov bx, si
    add si, cx ; 03/02/2013
    mov di, si
    ;mov cx, 10
    mov cl, 10
    mov dl, '0'

@@:
    mov byte ptr [BX], dl
    inc bx
    loop @b
    ;
    ;xor dl, dl
    ;mov byte ptr [BX], dl
    mov bx, 10
    xor dx, dx
pdn_itoa:
    div bx
    ; 03/02/2013
    add byte ptr [SI], dl ; 03/02/2013
    and dl, dl
    jnz short @f
    and al, al
    jz short pdn_14
@@:
    dec si
    xor dl, dl
    jmp short pdn_itoa
pdn_14:
    mov si, offset dec_num
    mov bx, si
@@:
    ; leading zeros will not be printed
    mov al, byte ptr [BX] ; 03/02/2013
    cmp al, '0'
    ja short @f
    cmp bx, di
    jnb short @f
    mov al, 20h
    mov byte ptr [BX], al
    inc bx
    jmp short @b
@@:
    mov ah, 0Eh

```

```

        mov bx, 07h
@@:
        lodsb
pdn_putc:
        int 10h

        cmp si, di
        jna short @b

        ;mov al, 20h
        ;int 10h

        retn

print_decimal_number endp

print_volume_info proc near
        ; 06/03/2013
        ; 05/03/2013
        ; 16/02/2013

        mov bx, word ptr [BSSBuffer] ; 06/03/2013
        add bx, bsVolumeSerial+2
        mov cx, 2
        mov di, offset msgVolume_Serial
@@:
        mov ax, word ptr [BX]
        call proc_hex_double
        stosw
        mov ax, dx
        stosw
        dec cx
        jz short @f
        inc di
        sub bx, 2
        jmp short @b
@@:
        mov si, offset msgVolume_Info
        call UNIX_PRINTMSG
@@:
        mov bx, word ptr [SUPERBLOCK] ; SuperBlock
        ; start of free storage map for disk
@@:
        mov ax, word ptr [BX] ; first word contains # of bytes
        ; in free storage map
        shl ax, 1 ; multiply AX by 8 gives # of blocks
        shl ax, 1
        shl ax, 1
        push ax
        mov si, offset msgVol_Size_Hdr
        call UNIX_PRINTMSG
        pop ax
        push ax
        mov cl, 4 ; mov cx, 4
        call print_decimal_number
        mov si, offset msgVolume_Size
        call UNIX_PRINTMSG
        pop cx ; cx = bit count of free storage map
        xor dx, dx ; mov dx, 0
        xor bl, bl ; xor bx, bx
        mov si, word ptr [SUPERBLOCK]
        add si, 2
        mov di, 16
pvi_size_loop1:
        lodsw
        or ax, ax
        jz short pvi_size_loop3
        push cx
        mov cx, di
pvi_size_loop2:
        shr ax, 1
        jnc short @f
        inc bx
@@:
        loop pvi_size_loop2
        pop cx

```

```

pvi_size_loop3:
    add dx, di
    cmp dx, cx
    jb short pvi_size_loop1

    push bx
    mov si, offset msgVol_freeblocks_Hdr
    call UNIX_PRINTMSG
    pop ax ; # of free blocks
    mov cx, 4
    call print_decimal_number
    mov si, offset msgVolume_freeblocks
    call UNIX_PRINTMSG

@@:
    mov bx, word ptr [SUPERBLOCK]
    mov ax, word ptr [BX]
    add ax, 2
    add bx, ax ; 06/03/2013
                ; start of inode map for disk
@@:
    mov ax, word ptr [BX] ; first word contains # of bytes
                ; in inode map
    shl ax, 1        ; multiply AX by 8 gives # of inodes
    shl ax, 1
    shl ax, 1
    push bx
    push ax
    mov si, offset msgVol_icount_Hdr
    call UNIX_PRINTMSG
    pop ax
    push ax
    mov cl, 4 ; mov cx, 4
    call print_decimal_number
    mov si, offset msgVolume_icount
    call UNIX_PRINTMSG
    pop cx        ; cx = bit count of inode map
    pop si        ; inode map offset
    xor dx, dx ; mov dx, 0
    xor bl, bl ; xor bx, bx
    mov di, 16
pvi_icount_loop1:
    lodsw
    ; cmp ax, 0FFFFh
    ; je short pvi_icount_loop3
    inc ax
    jz short pvi_icount_loop3
    dec ax
    push cx
    mov cx, di
pvi_icount_loop2:
    shr ax, 1
    jc short @f
    inc bx
@@:
    loop pvi_icount_loop2
    pop cx
pvi_icount_loop3:
    add dx, di
    cmp dx, cx
    jb short pvi_icount_loop1

    push bx
    mov si, offset msgVol_free_icount_Hdr
    call UNIX_PRINTMSG
    pop ax ; # of free inodes
    mov cx, 4
    call print_decimal_number
    mov si, offset msgVolume_free_icount
    call UNIX_PRINTMSG

    retn

print_volume_info endp

```



```

proc_hex_double proc near
    ; 16/02/2013 (AX:DX)
    ; 28/01/2002 (DX:AX)
    ; From binary (word) to hexadecimal (character) converter
    ;
    ; input -> AX = word (binary number) to be converted
    ; output -> AX = First 2 characters of hexadecimal number
    ; output -> DX = Last 2 characters of hexadecimal number

    push cx
    xor dx, dx
    mov cx, 10h
    div cx      ; Q in AX, R in DX (DL)
    push dx     ; DH= 0, R in DL <- CX= 10h
    xor dl, dl
    div cx      ; DH= 0, R in DL, AX <= FFh
    div cl      ; AL <= 0Fh
                ; R in AH, Q in AL
    pop cx      ; R in CL
    mov dh, cl

    or dx, '00'

    cmp dl, '9'
    jna short pass_cc_dl
    add dl, 7
pass_cc_dl:
    cmp dh, '9'
    jna short pass_cc_dh
    add dh, 7
pass_cc_dh:
    or ax, '00'

    cmp al, '9'
    jna short pass_cc_al
    add al, 7
pass_cc_al:
    cmp ah, '9'
    jna short pass_cc_ah
    add ah, 7
pass_cc_ah:
    pop cx

    retn

proc_hex_double endp

show_inode proc near
    ; 05/03/2013
    ; 17/02/2013
    ; print inode details
    ; Format: inode <decimal number>, iget <decimal number>
    ; INPUT -> AX <> 0 -> Current Inode [iil]
    ;          AX = 0 -> use inode number input
    ;
    and ax, ax
    jnz short show_inode_7
    mov word ptr [arg], ax ; 0
    xor dx, dx
show_inode_1:
    lodsb
    cmp al, '0'
    jnb short show_inode_4
    cmp al, '9'
    ja short show_inode_stc_retn ; cmc
    sub al, '0'
show_inode_2:
    or dx, dx
    jnz short show_inode_5
show_inode_3:
    mov dx, ax
    jmp short show_inode_1
show_inode_4:
    or dx, dx
    jz short show_inode_stc_retn
    cmp al, 20h
    jna short show_inode_6

```

```

show_inode_stc_retn:
    cmc
show_inode_retn:
    retn
show_inode_5:
    cmp dx, 256
    jnb short show_inode_stc_retn
    mov ah, dl
    mov dl, al
    mov al, 10
    mul ah
    add dx, ax
    jmp short show_inode_1
show_inode_6:
    mov bx, word ptr [SUPERBLOCK]
    mov ax, word ptr [BX]
    add ax, 2
    mov bx, ax
    mov ax, word ptr [BX] ; inode map bytes
    shl ax, 1
    shl ax, 1
    shl ax, 1 ; inode count
    add ax, 40 ; + device file inodes
    cmp ax, dx
    jb short show_inode_retn ; not a valid i-number
    mov ax, dx
    mov word ptr [arg], ax
    ; ax = i-number
    call i_get
    jc short show_inode_retn
show_inode_7:
    ;mov ax, word ptr [ii]
    call proc_hex_double
    mov word ptr [txt_inode_number], ax
    mov word ptr [txt_inode_number]+2, dx
    mov ax, word ptr [inode_flg]
    push ax
    call proc_hex_double
    mov word ptr [txt_inode_flags_h], ax
    mov word ptr [txt_inode_flags_h]+2, dx
    pop dx
    mov di, offset txt_inode_flags_b
    mov cx, 16
@@:
    xor al, al ; 0
    shl dx, 1
    adc al, '0'
    stosb
    loop @b
    mov ax, word ptr [inode_nlks] ; & uid
    call proc_hex_double
    mov word ptr [txt_inode_nlks], dx
    mov word ptr [txt_inode_uid], ax
    mov ax, word ptr [inode_size]
    call proc_hex_double
    mov word ptr [txt_inode_size], ax
    mov word ptr [txt_inode_size]+2, dx
    mov cl, 8
    mov si, offset inode_dskp
    mov di, offset txt_inode_dskp
@@:
    lodsw
    call proc_hex_double
    stosw
    mov ax, dx
    stosw
    dec cl
    jz short @f
    inc di
    inc di
    jmp short @b
@@:
    ;mov si, offset inode_ctim
    mov ax, word ptr [SI]
    mov dx, word ptr [SI]+2
    push dx
    push ax
    push dx

```

```

call proc_hex_double
mov word ptr [txt_inode_ctim_h]+4, ax
mov word ptr [txt_inode_ctim_h]+6, dx
pop ax
call proc_hex_double
mov word ptr [txt_inode_ctim_h], ax
mov word ptr [txt_inode_ctim_h]+2, dx
pop ax
pop dx
call convert_from_epoch
mov ax, word ptr [year]
mov si, offset txt_inode_cyear
;mov cx, 4
mov cl, 4
call proc_bin_to_decimal
mov ax, word ptr [month]
mov si, offset txt_inode_cmonth
mov cl, 2
call proc_bin_to_decimal
mov ax, word ptr [day]
mov si, offset txt_inode_cday
mov cl, 2
call proc_bin_to_decimal
mov ax, word ptr [hour]
mov si, offset txt_inode_chour
mov cl, 2
call proc_bin_to_decimal
mov ax, word ptr [minute]
mov si, offset txt_inode_cminute
mov cl, 2
call proc_bin_to_decimal
mov ax, word ptr [second]
mov si, offset txt_inode_csecond
mov cl, 2
call proc_bin_to_decimal
mov si, offset inode_mtim
mov ax, word ptr [SI]
mov dx, word ptr [SI]+2
push dx
push ax
push dx
call proc_hex_double
mov word ptr [txt_inode_mtim_h]+4, ax
mov word ptr [txt_inode_mtim_h]+6, dx
pop ax
call proc_hex_double
mov word ptr [txt_inode_mtim_h], ax
mov word ptr [txt_inode_mtim_h]+2, dx
pop ax
pop dx
call convert_from_epoch
mov ax, word ptr [year]
mov si, offset txt_inode_myear
;mov cx, 4
mov cl, 4
call proc_bin_to_decimal
mov ax, word ptr [month]
mov si, offset txt_inode_mmonth
mov cl, 2
call proc_bin_to_decimal
mov ax, word ptr [day]
mov si, offset txt_inode_mday
mov cl, 2
call proc_bin_to_decimal
mov ax, word ptr [hour]
mov si, offset txt_inode_mhour
mov cl, 2
call proc_bin_to_decimal
mov ax, word ptr [minute]
mov si, offset txt_inode_mminute
mov cl, 2
call proc_bin_to_decimal
mov ax, word ptr [second]
mov si, offset txt_inode_msecond
mov cl, 2
call proc_bin_to_decimal
mov ax, word ptr [inode_reserved]
call proc_hex_double

```

```

        mov word ptr [txt_inode_reserved], ax
        mov word ptr [txt_inode_reserved]+2, dx
@@:
        mov si, offset msg_inode_details
        call UNIX_PRINTMSG
        retn

show_inode endp

load_kernel proc near
        ; 06/03/2013
        ;
        ; loads unix kernel file
        ;
        ; INPUT -> u_namep = unix kernel/file name address
        ; unix kernel will be loaded at 'kernel_loading_segment'
        ;

load_k_1:
        call name_i
        jc short @f
load_k_2:
        call i_get
        jc short @f

        mov bx, inode_flg

        test word ptr [BX], 10h ; executable file attribute bit
        jz short load_k_stc

        mov bx, inode_size

        xor ax, ax

        cmp word ptr [BX], ax ; 0
        jna short load_k_stc

        mov word ptr [u_off], ax ; 0
        mov word ptr [u_base], ax ; 0

        ;mov bx, inode_size
        mov ax, word ptr [BX]
        mov word ptr [u_count], ax

        mov ax, kernel_loading_segment
        mov word ptr [EXTRA_SEGMENT], ax

        mov ax, word ptr [ii]
        call read_i
        jc short load_k_retn

        mov cx, word ptr [u_nread]
        mov bx, inode_size
        cmp cx, word ptr [BX]

load_k_retn:
        mov ax, ds
        mov word ptr [EXTRA_SEGMENT], ax
@@:
        retn

load_k_stc:
        stc
        retn

load_kernel endp

align 2 ; 05/03/2013
PhysicalDriveNumber: db 0
db 0

```

```

;- - - - -
; messages
;- - - - -

UNIX_Welcome:
    db 'Retro UNIX 8086 v1', 0

Boot_Msg:
    db 0Dh, 0Ah
    db 'Boot: '
    db 0
align 2 ; 05/03/2013
unix_cdrv:
    db 0Dh, 0Ah

UNIX_FD_Name:
    db 'fd'

UNIX_FD_Number:
    db '0:'

unix_cdir:
    db '/'
    db 37 dup(0)

CDirOffset:
    dw 0

CursorColumn:
    dw 0

pdir:
    dw 0

arg:
    dw 0

msg_unix_drv_read_error:
    db 0Dh, 0Ah
    db "Drive not ready or read error!"
    db 0Dh, 0Ah, 0

Msg_StartupFile_Name:
    db 0Dh, 0Ah
    db "Startup File Name : ", 0

error_msg:
    db 0Dh, 0Ah
    db 'Error !'

UNIX_CRLF:
    db 0Dh, 0Ah, 0

RetryCount:
    dw 0

DirFileName:
    db 20h ; 06/01/2013

BOOT_FILE_NAME: db 9 dup(0)

uf_i_number: dw 0 ; 25/11/2012

bootfile_inode:
inode:
inode_flg:    dw 801Eh ; Flags (1000000000011110b)
inode_nlnks:  db 1      ; number of links
inode_uid:    db 0      ; user ID (0 = root)
inode_size:   dw 0      ; file size
inode_dskp:   dw 8 dup(0) ; indirect or contents blocks
inode_ctim:   dd 0      ; creation date & time
inode_mtim:   dd 0      ; modification date & time
inode_reserved: dw 0     ; unused

align 2 ; 05/03/2013
db 0FFh

U:
u_uid: db 0
u_cdir: dw ROOT_DIR_INODE_NUMBER
u_namep: dw 0
u_dirp: dw 0
u_base: dw 0
u_off: dw 0
u_count: dw 0
u_nread: dw 0
u_dirbuf: db 10 dup(0)

```

```

ii: dw 0
buff_s: dw 0

year: dw 1970
month: dw 1
day: dw 1
hour: dw 0
minute: dw 0
second: dw 0

DMonth:
dw 0
dw 31
dw 59
dw 90
dw 120
dw 151
dw 181
dw 212
dw 243
dw 273
dw 304
dw 334

; 25/11/2012
str_inode_number:
    db 0Dh, 0Ah
    db 'Startup File I-Number: ', 0
Decimal_i_no_str:
    db 6 dup (0)

Str_startup_file_size:
    db 0Dh, 0Ah
    db 'Startup File Size : ', 0
Str_Bytes:
    db ' bytes', 0

Decimal_size_str: db 6 dup (0)

Str_sf_date_time:
    db 0Dh, 0Ah
    db 'Creating Date & Time      : '
Str_cday:
    db '00'
    db '/'
Str_cmonth:
    db '00'
    db '/'
Str_cyear:
    db '0000'
    db 20h, 20h
Str_chour:
    db '00'
    db ':'
Str_cminute:
    db '00'
    db ':'
Str_csecond:
    db '00'
    db 0Dh, 0Ah
    db 'Last Modif. Date & Time : '
Str_mday:
    db '00'
    db '/'
Str_mmonth:
    db '00'
    db '/'
Str_myear:
    db '0000'
    db 20h, 20h
Str_mhour:
    db '00'
    db ':'
Str_mminute:
    db '00'
    db ':'
Str_msecond:
    db '00'
    db 0Dh, 0Ah, 0

;23/02/2013
list_count: db 0FFh
; 20/01/2013
ls_option: db 0
; 21/01/2013
dec_num: db 10 dup(20h) ; 02/03/2012, 3 bytes -> 10 bytes
db 0

```

```

;30/12/2012
DotDot:
db '.'
Dot:
db '.'
db 0

;16/02/2013
msgVolume_Info:
    db 0Dh, 0Ah
    db "Retro UNIX 8086 v1 (RUFFS) File System", 0Dh, 0Ah
    db "by Erdogan Tan (2013)"
    db 0Dh, 0Ah, 0Dh, 0Ah
    db "Volume Serial No: "
msgVolume_Serial:
    db "0000-0000h"
    db 0Dh, 0Ah, 0
msgVol_Size_Hdr:db "Volume Size : ", 0
msgVolume_Size: ; db "0000"
    db " blocks", 0Dh, 0Ah, 0
msgVol_freeblocks_Hdr:db "Free Count : ", 0
msgVolume_freeblocks : ;db "0000"
    db " blocks", 0Dh, 0Ah, 0
msgVol_icount_Hdr:
    db "# of Inodes : ", 0
msgVolume_icount: ; db "0000"
    db "+40", 0Dh, 0Ah, 0
msgVol_free_icount_Hdr:db "Free Inodes : ', 0
msgVolume_free_icount : ;db "0000"
    db 0Dh, 0Ah, 0
NotFound_msg:
    db 0Dh, 0Ah
    db "Not found !"
    db 0Dh, 0Ah, 0
msgINumber:
    db 0Dh, 0Ah
    db "Inode Number : ", 0

msg_inode_details:
    db 0Dh, 0Ah
    db "UNIX V1 I-NODE STRUCTURE DETAILS OF I-NODE "
txt_inode_number:
    db "0000h"
    db 0Dh, 0Ah, 0Dh, 0Ah
    db "Flags : "
txt_inode_flags_h:
    db "0000h"
    db 20h, 20h
    db "["
txt_inode_flags_b:
    db "0000000000000000b"
    db "]"
    db 0Dh, 0Ah
    db "# of Links : "
txt_inode_nlnks:
    db "00h"
    db 0Dh, 0Ah
    db "User ID : "
txt_inode_uid:
    db "00h"
    db 0Dh, 0Ah
    db "Size : "
txt_inode_size:
    db "0000h"
    db 0Dh, 0Ah
    db "Disk Blocks : "
txt_inode_dskp:
    db "0000h 0000h 0000h 0000h "
    db "0000h 0000h 0000h 0000h"
    db 0Dh, 0Ah
    db "Creation Time : "
txt_inode_ctim_h:
    db "00000000h"
    db 20h, 20h
    db "["
txt_inode_cday:

```

```

        db "00"
        db "/"
txt_inode_cmonth:
        db "00"
        db "/"
txt_inode_cyear:
        db "0000"
        db ","
txt_inode_chour:
        db "00"
        db ":"
txt_inode_cminute:
        db "00"
        db ":"
txt_inode_csecond:
        db "00"
        db "]"
        db 0Dh, 0Ah
        db "Modification Time : "
txt_inode_mtim_h:
        db "00000000h"
        db 20h, 20h
        db "["
txt_inode_mday:
        db "00"
        db "/"
txt_inode_mmonth:
        db "00"
        db "/"
txt_inode_myear:
        db "0000"
        db ","
txt_inode_mhour:
        db "00"
        db ":"
txt_inode_mminute:
        db "00"
        db ":"
txt_inode_msecond:
        db "00"
        db "]"
        db 0Dh, 0Ah
        db "Unused : "
txt_inode_reserved:
        db "0000h"
        db 0Dh, 0Ah, 0

Boot_Commands: ; 25/02/2013
db 0Dh, 0Ah
db "BOOT COMMANDS", 0Dh, 0Ah
db "dir <directory name>      : print directory entries without details", 0Dh, 0Ah
db "ls <directory name>      : print directory entries, ", 27h, "/", 27h, " means entry is
directory", 0Dh, 0Ah
db "ls -l <directory name> : print directory entries with details", 0Dh, 0Ah
db "cd <directory name>      : change directory", 0Dh, 0Ah
db "show <file name>         : show file, print/display file contents", 0Dh, 0Ah
db "inode <inode number>     : print inode details for (decimal) inode number", 0Dh, 0Ah
db "namei <file name>        : print inode number of file (as decimal)", 0Dh, 0Ah
db "fs                       : print (current) unix fs (super block) info", 0Dh, 0Ah
db "bootfile                 : print startup/boot file details", 0Dh, 0Ah
db "reboot                   : reboot (int 19h)", 0Dh, 0Ah
db "?"                       : print boot commands summary (as above)", 0Dh, 0Ah, 0

```



```
align 2 ; 05/03/2013
CommandBuffer: db 74 dup(0)
unix_reboot:   db 0
def_kernel:    db 0
BSBUFFER:      dw 0
SUPERBLOCK:    dw 0
DISKBUFFER:    dw 0
FILEBUFFER:    dw 0
EXTRA_SEGMENT: dw 0 ; 06/03/2013
; 07/03/2013
waiting_count: dw 182*3 ; 30 seconds
```

```
align 16 ; 05/03/2013
```

```
EndOfFile:
```

```
;- - - - -
; buffers
;- - - - -
;BSBUFFER:    db 512 dup(0)
;SUPERBLOCK:  db 512 dup(0)
;DISKBUFFER:  db 512 dup(0)
;FILEBUFFER:  db 512 dup(0)
;;;
;;BootStack:
```

```
BOOT1 ends
```

```
end START_CODE
```