

```

; UNIXCOPY.ASM (Only for 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 - DOS & UNIX FS file export/import Utility
; (08/12/2012)
;
; [ Last Modification: 14/07/2015 ]
;
; 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

; DTA (PSP+80h= Offset 128)
DTA_Attrib equ 149 ; PDP+21 ; 05/01/2013
DTA_Time   equ 150 ; PSP+22
DTA_Date   equ 152 ; PSP 24
DTA_FileSize equ 154 ; PSP + 26
DTA_FileName equ 158 ; PSP + 30

;err_INVALIDDATA equ 100h
;err_NOFREEBLOCK equ 200h

i_flags      equ 001Eh

.8086

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

          org 100h

START_CODE:

proc_start  proc near
             ; 08/12/2012 (UNIXCOPY)
             ;
             ; 30/11/2012 (UNIXBOOT)
             ;
             mov bx, SizeOfFile+100
             add bx, 15
             shr bx, 1
             shr bx, 1
             shr bx, 1
             shr bx, 1
             mov ah, 4Ah ; modify memory allocation
             ;push cs
             ;pop es
             int 21h

;-----
; see if drive specified
;-----

             mov si, offset 80h           ; PSP command tail
             lodsb
             mov cl, al
             or cl, cl
             jnz short loc_get_args
             dec cl
             jmp loc_unix_welcome

```

```

loc_get_args:
    lodsb
    cmp al, ' '
    jne short loc_check_fd_name

    ;dec cl
    ;jz short loc_unix_welcome

    jmp short loc_get_args

loc_check_fd_name:
    ; 07/07/2015
    mov di, offset img_file_name
    cmp al, "f"
    jne short loc_chk_fname1
    stosb
    lodsb
    cmp al, "d"
    jne short loc_chk_fname1
    stosb
    lodsb
    cmp al, '0'
    jnb short loc_chk_fname2
    cmp al, '1'
    ja short loc_chk_fname2
    stosb
    mov dl, al
    lodsb
    cmp al, 0Dh
    ja short loc_chk_fname2
    mov byte ptr [UNIX_FD_Number], dl
    sub dl, '0'
    mov byte ptr [PhysicalDriveNumber], dl
    jmp load_boot_sector

loc_check_file_name:
    ; 07/07/2015
    lodsb

loc_chk_fname1:
    cmp al, 0Dh
    jna short loc_chk_fname_ok

loc_chk_fname2:
    stosb
    cmp di, offset img_file_name + 12
    jnb short loc_check_file_name
    je short loc_chk_fname_ok

loc_inv_fname:
    mov si, offset msg_inv_file_name
    jmp @f

loc_chk_fname_ok:
    sub al, al
    stosb

loc_cap_file_name:
    ; file name capitalization
    mov si, offset img_file_name
    mov di, si
    mov bx, si

loc_cap_file_name0:
    lodsb
    cmp al, 'a'
    jnb short loc_cap_file_name2
    and al, al
    jz short loc_cap_file_name3
    cmp al, '.'
    jne short loc_cap_file_name1
    mov bx, di ; dot position

loc_cap_file_name1:
    stosb
    jmp short loc_cap_file_name0

loc_cap_file_name2:
    cmp al, 'z'
    ja short loc_cap_file_name1
    and al, 0DFh ; NOT 32
    stosb
    jmp short loc_cap_file_name0

```

```

loc_cap_file_name3:
    mov [di], al
    dec di
    cmp bx, di
    jnb short loc_inv_fname
    sub di, bx
    sub bx, offset img_file_name
    cmp di, 3
    jna short loc_cap_file_name4
    and bx, bx
    jnz short loc_inv_fname
    jmp short loc_find_image_file
loc_cap_file_name4:
    cmp bx, 8
    ja short loc_inv_fname
loc_find_image_file:
    ; 07/07/2015
    mov dx, offset img_file_name
    mov cx, 3Fh ; File Attributes
    mov ah, 4Eh ; MS Dos Function = Find First File
    int 21h
    jnc short loc_chk_image_file_features
    cmp ah, 03h ; dos error number > 3
    ja loc_error
    mov si, offset msg_file_not_found
    jmp @f
loc_chk_image_file_features:
    mov si, DTA_Attrib
    mov al, byte ptr [SI]
    and al, 1Fh ; directory, volume label, system, hidden, read only
    jnz loc_error
    mov si, DTA_FileSize
    lodsw
    cmp word ptr [SI], 16h
    jne short loc_inv_image_file
    cmp ax, 8000h ; 1.44 MB floppy disk image (168000h bytes)
    je short loc_open_image_file
loc_inv_image_file:
    mov si, offset msg_inv_image_file
    jmp short @f
loc_open_image_file:
    mov al, 2 ; open for reading and writing
    ; mov dx, offset img_file_name
    mov ah, 3Dh ; open file
    int 21h
    jc short loc_error
    mov word ptr [img_file_handle], ax
    mov byte ptr [PhysicalDriveNumber], 90h ; image file sign
    ;
    mov bx, ax
    mov cx, 1024 ; read 1024 bytes (2 sectors)
    mov dx, offset BSBuffer ; bootsector (& super block) buffer
    mov ah, 3Fh ; read file
    int 21h
    jc short loc_error
    cmp ax, 1024
    jne short loc_error
    mov bx, dx ; offset BSBuffer
    jmp short load_fd_img_boot_sect_ok
load_boot_sect:
    ; input -> dl = drive number
    xor ah, ah
    int 13h
    jc short loc_drv_read_error
load_boot_sect_ok:
    mov bx, offset BSBuffer
    mov ax, 0202h ; Read boot sector & super block
    mov cx, 1
    xor dh, dh
    int 13h
    jc short loc_drv_read_error
load_fd_img_boot_sect_ok:
    cmp word ptr [BX]+510, 0AA55h
    jne short loc_not_fd_rufs
    cmp word ptr [BX]+bsFSystemID, 'UR'
    jne short loc_not_fd_rufs
    cmp word ptr [BX]+bsFSystemID+2, 'SF'
    je short loc_check_fd_sign

```

```

loc_not_fd_rufs:
    mov si, offset msg_Not_Unix_FS
    jmp short @f

loc_check_fd_sign:
    cmp word ptr [BX]+bsFDSign, 'df'
    jne short loc_not_fd_rufs

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

loc_unix_welcome:
    pushf
    mov si, offset UNIX_Welcome
    call UNIX_PRINTMSG
    popf
    je short loc_call_unix_prompt
    mov si, offset usage

@@:
    call UNIX_PRINTMSG

loc_close_file_then_terminate:
    ; 07/07/2015
    mov bx, [img_file_handle]
    and bx, bx
    jz short terminate

close_img_file:
    mov ah, 3Eh ; close (floppy disk image) file
    int 21h

terminate:
    int 20h

loc_drv_read_error:
    mov si, offset msg_unix_drv_read_error
    jmp short @b

loc_error:
    mov si, offset error_msg
    jmp short @b

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

loc_call_unix_prompt:
    call unix_prompt
    ; 07/07/2015
    jmp short loc_close_file_then_terminate

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
    ; 07/07/2015
    ; 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_0:
    ; 07/07/2015
    cmp     byte ptr [PhysicalDriveNumber], 90h
    jb      short unix_prompt_1
    mov     si, offset img_file_name
    call    unix_printmsg
    mov     si, offset unix_img_cdir
    jmp     short unix_prompt_15
unix_prompt_1:
    mov     si, offset unix_cdrv
    call    unix_printmsg
unix_prompt_2:
    mov     si, offset unix_cdir
unix_prompt_15:
    call    unix_printmsg
unix_prompt_3:
    mov     al, byte ptr [unix_prompt_char]
    ;mov     ah, 0Eh
    ;mov     bx, 07h
    int     10h
unix_prompt_4:
    mov     ah, 03h
    ;mov     bx, 07h
    int     10h
    mov     byte ptr [CursorColumn], dl
unix_prompt_5:
    mov     si, offset CommandBuffer
    call    proc_rw_char
    ;mov     byte ptr [CommandBuffer]+75, 0

    ;mov     si, offset CommandBuffer
    mov     di, si
    xor     bx, bx
    xor     cx, cx
unix_prompt_6:
    mov     al, byte ptr [SI][BX]
    inc     bl
    cmp     al, 20h
    ja      short unix_prompt_8
    jb      short unix_prompt_13
    cmp     bl, 74 ; 75 ?
    jb      short unix_prompt_6
    jmp     short unix_prompt_13
unix_prompt_7:
    mov     al, byte ptr [SI][BX]
    inc     bl
    cmp     al, 20h
    jna     short unix_prompt_9
unix_prompt_8:
    stosb
    inc     cl
    cmp     bl, 74 ; 75 ?
    jb      short unix_prompt_7
    ;jmp     short unix_prompt_12
unix_prompt_9:
    xor     al, al ; 0
unix_prompt_10:
    mov     byte ptr [DI], al
    inc     di
    cmp     bl, 74 ; 75 ?
    jnb     short unix_prompt_12
    mov     al, byte ptr [SI][BX]
    inc     bl
    cmp     al, 20h
    jnb     short unix_prompt_10

```

```

unix_prompt_11: mov     byte ptr [DI], 0
unix_prompt_12: call    command_interpreter

                cmp     byte ptr [program_exit], 1
                jnb     short unix_prompt_14

                mov     cx, 74 ; 75 ?
                mov     di, offset CommandBuffer
                xor     al,al
                rep     stosb
unix_prompt_13: mov     bx,07h
                mov     al,0Dh
                mov     ah,0Eh
                int     10h
                mov     al,0Ah
                int     10h

                jmp     unix_prompt_0 ; loop

unix_prompt_14: 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
        jb      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
; 01/03/2013
; 25/02/2013
; 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   ; 25/02/2013
c14:
        cmp     cl, 4
        ja      cl5
        jb      cl3
; EXIT
loc_cmd_exit:
        cmp     ax, 'xe'
        jne     short loc_cmd_show
        lodsw
        cmp     ax, 'ti'
        jne     short @f
        lodsb
        or      al, al
        jnz     short @f

        mov     byte ptr [program_exit], 1
@@:
        retn
; SHOW
loc_cmd_show:
; 06/01/2013
        cmp     ax, 'hs'
        jne     short loc_cmd_link
        lodsw
        cmp     ax, 'wo'
        jne     short @b
        lodsb
        or      al, al
        jnz     short @b

```

```

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

; LINK
loc_cmd_link:
    cmp ax, 'il'
    jne loc_cmd_iget ; 17/02/2013
    lodsw
    cmp ax, 'kn'
    jne short @b
    lodsb
    or al, al
    jnz short @b

link_sf1:
    mov word ptr [u_namep], si
    lodsb
    cmp al, 20h
    je short link_sf1
    jb short @b

link_sf2:
    lodsb
    cmp al, 20h
    ja short link_sf2
    xor al, al
    mov byte ptr [SI]-1, al

link_df1:
    mov word ptr [arg], si
    lodsb
    cmp al, 20h
    je short link_df1
    jb short @b

link_df2:
    lodsb
    cmp al, 20h
    ja short link_df2
    dec si
    xor al, al
    mov byte ptr [SI], al

link_fsf:
    call name_i
    jc ci_error

link_fdf:
    mov word ptr [uf_i_number], ax

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

    call name_i
    jnc ci_error

    cmp ah, 0FFh
    jne ci_error

    ; [u_dirp] = empty directory entry slot

    mov ax, word ptr [ii]
    mov word ptr [pdir], ax

    mov ax, word ptr [uf_i_number]
    call i_get
    jc ci_error

    call set_imod ; jsr r0,setimod / set modified flag
    inc byte ptr [inode_nlks] ; link count

```

```

        mov ax, word ptr [pdir]
        call i_get
        jc ci_error

        ; name_i -> u_namep points filename
        ;          after the last '/' of the path

        mov ax, word ptr [uf_i_number]
        mov word ptr [u_dirbuf], ax

        call mk_dir ; make directory entry
        jc ci_error

        jmp ci_sync_exit

; IGET
loc_cmd_iget:    ; 17/02/2013, inode/iget
                cmp ax, 'gi'
                jne short loc_cmd_help ; 23/02/2013
                lodsw
                cmp ax, 'te'
                jne short @f
                lodsb
                or al, al
                jnz short @f
ci_iget_getarg:
                mov bx, si
                lodsb
                cmp al, 20h
                ja inode_getarg2
                je short ci_iget_getarg

; HELP
loc_cmd_help:   ; 23/02/2013
                cmp ax, 'eh'
                jne short @f
                lodsw
                cmp ax, 'pl'
                jne short @f
                lodsb
                and al, al
                jnz short @f

ci_?:
                mov si, offset UNIXCOPY_Commands
                call UNIX_PRINTMSG

@@:
                retn

c15:
                cmp cl, 5
                ja c17
                jnb short @f

; CHDIR
loc_cmd_chdir:
                cmp ax, 'hc'
                jne loc_cmd_todos
                lodsw
                cmp ax, 'id'
                jne short loc_cmd_chmod
                lodsb
                cmp al, 'r'
                jne short @f
                lodsb
                or al, al
                jnz short @f

ci_cd_getarg:
                mov word ptr [u_namep], si
                lodsb
                cmp al, 20h
                je short ci_cd_getarg
                jnb 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

```

```

; CHMOD
loc_cmd_chmod: ; 13/01/2013
               ; cmp ax, 'hc'
               ; jne short loc_cmd_todos
               ; lodsw
               cmp ax, 'om'
               jne short loc_cmd_chown
               lodsb
               cmp al, 'd'
               jne short @b
               lodsb
               or al, al
               jnz short @b
ci_chmod_getarg:
               lodsb
               cmp al, 20h
               je short ci_chmod_getarg
               jb short @b
               dec si
               call chmode
               jc ci_error
               mov ax, word ptr [arg]
               or ax, ax
               jz short @b
               xor al, al
               mov byte ptr [arg]+2, al
ci_chown_print:
               mov si, offset msg_arg
               call UNIX_PRINTMSG
               jmp ci_sync_exit

; CHOWN
loc_cmd_chown: ; 13/01/2013
               ; cmp ax, 'hc'
               ; jne short loc_cmd_todos
               ; lodsw
               cmp ax, 'wo'
               jne short @b
               lodsb
               cmp al, 'n'
               jne short @b
               lodsb
               or al, al
               jnz short @b
ci_chown_getarg:
               lodsb
               cmp al, 20h
               je short ci_chown_getarg
               jb short @b
               dec si
               call chowner
               jc ci_error

               and bx, bx
               jnz short ci_chown_print
@@:
               retn

; TODOS
loc_cmd_todos: ; 24/12/2012
               cmp ax, 'ot'
               jne loc_cmd_mkdir ; 30/12/2012
               lodsw
               cmp ax, 'od'
               jne short @b
               lodsb
               cmp al, 's'
               jne short @b
               lodsb
               or al, al
               jnz short @b
todos_uf1:
               mov word ptr [u_namep], si
               lodsb
               cmp al, 20h
               je short todos_uf1
               jb short @b

```

```

todos_uf2:
    lodsb
    cmp al, 20h
    ja short todos_uf2
    xor al, al
    mov byte ptr [SI]-1, al

todos_df1:
    mov word ptr [arg], si
    lodsb
    cmp al, 20h
    je short todos_df1
    jb short @b

todos_df2:
    lodsb
    cmp al, 20h
    ja short todos_df2
    dec si
    xor al, al
    mov byte ptr [SI], al

todos_fuf:
    call name_i
    ;jnc short @f
    jc ci_error

    ;cmp ah, 0FFh
    ;jne ci_error
    ; jmp ci_error ; 'file not found' error

@@:
    mov word ptr [uf_i_number], ax

todos_fdf:
    mov dx, word ptr [arg]
    mov cx, 3Fh ; File Attributes ; 05/01/2013 (3Fh)
    mov ah, 4Eh ; MS Dos Function = Find First File
    int 21h
    ;jnc short todos_afow
    jnc short @f ; 05/01/2013

todos_chk_err:
    cmp ah, 03h ; dos error number > 3
    ja ci_error

    jmp short todos_crdf
    ; 05/01/2013
@@:
    mov si, DTA_Attrib
    mov al, byte ptr [SI]
    and al, 1Fh ; directory, volume label, system, hidden, read only
    jnz ci_error

todos_afow:
    ; overwrite question
    mov si, offset msg_overwrite_question1
    call UNIX_PRINTMSG
    mov si, DTA_FileName
    call UNIX_PRINTMSG
    mov si, offset msg_overwrite_question2
    call UNIX_PRINTMSG
    mov si, offset msg_yes_no
    call UNIX_PRINTMSG

todos_afow_input:
    ; ask for overwrite
    xor ax, ax
    int 16h
    ; wait for keyboard command
    cmp al, 'C'-40h
    je short @f
    cmp al, 27
    je short @f
    and al, 0DFh
    cmp al, 'Y'
    ; Yes?
    je short todos_afow_yes
    ; overwrite
    cmp al, 'N'
    ; No?
    jne short todos_afow_input

todos_afow_no:
    mov si, offset msg_No
    call UNIX_PRINTMSG
    mov si, offset UNIX_CRLF
    call UNIX_PRINTMSG

@@:
    retn

todos_afow_yes:
    mov si, offset msg_YES
    call UNIX_PRINTMSG
    mov si, offset UNIX_CRLF
    call UNIX_PRINTMSG

```

```

todos_crdf:
    ;mov dx, word ptr [arg]
    xor cx, cx ; File Attributes = 0
    mov ah, 3Ch ; MS Dos Function = Create File
    int 21h
    jc ci_error

    mov word ptr [FileHandle], ax

todos_odf:
    mov dx, word ptr [arg]
    mov ah, 3Dh ; MS Dos Function = Open File
    xor al, al
    int 21h
    jc ci_error

todos_ruf_wdf:
    xor ax, ax
    mov word ptr [u_off], ax

todos_wf_msg:
    mov si, offset Msg_writing_file
    call UNIX_PRINTMSG

todos_iget:
    mov ax, word ptr [uf_i_number]
    call i_get
    jc short todos_cdf

    mov ax, word ptr [inode_size]
    mov cx, 512
    cmp ax, cx
    jna short loc_read_unix_sf

    mov ax, cx

@@:
loc_read_unix_sf:
    mov word ptr [u_count], ax

    mov word ptr [u_base], offset ReadBuffer

    mov ax, word ptr [uf_i_number] ; word ptr [u_dirbuf]
    call read_i
    jc short todos_cdf

@@:
;loc_write_dos_df:
    mov ah, 40h ; Write File
    mov cx, word ptr [u_nread] ; 0 -> eof
    mov dx, offset ReadBUFFER
    mov bx, word ptr [FileHandle]
    int 21h
    jc short todos_cdf

    cmp ax, cx ; write count = read count ?
    jne short todos_cdf ; jb short todos_cdf

    or ax, ax ; or cx, cx
    jnz short loc_read_unix_sf

todos_cdf:
    pushf
    jc short @f

todos_set_dfdt:
    mov ax, word ptr [inode_ctim] ; fromdos command ->
    mov dx, word ptr [inode_ctim]+2 ; dos lmdt -> unix ctim

    call convert_from_epoch

    mov dx, word ptr [hour]
    mov cl, 11
    shl dx, cl
    mov ax, word ptr [minute]
    mov cl, 5
    shl ax, cl
    or dx, ax
    mov ax, word ptr [second]
    shr ax, 1
    or ax, dx
    push ax ; time
    mov dx, word ptr [year]
    sub dx, 1980
    mov cl, 9
    shl dx, cl
    mov ax, word ptr [month]

```

```

        mov cl, 5
        shl ax, cl
        or dx, ax
        mov ax, word ptr [day]
        or dx, ax
        pop cx ; time
        mov ax, 5701h ; set lm date&time
        mov bx, word ptr [FileHandle]
        int 21h
@@:
        mov ah, 3Eh ; Close File
        mov bx, word ptr [FileHandle]
        int 21h
        popf
        jc ci_error
todos_retn:
        mov si, offset Msg_OK
        call UNIX_PRINTMSG
        mov si, offset UNIX_CRLF
        call UNIX_PRINTMSG
@@:
        retn
; MKDIR
loc_cmd_mkdir:
        ; 30/12/2012
        cmp ax, 'km'
        jne short loc_cmd_rmdir
        lodsw
        cmp ax, 'id'
        jne short @b
        lodsb
        cmp al, 'r'
        jne short @b
        lodsb
        or al, al
        jnz short @b
ci_mkdir_getarg1:
        mov word ptr [u_namep], si
        lodsb
        cmp al, 20h
        je short ci_mkdir_getarg1
        jnb short @b
ci_mkdir_getarg2: ; 06/01/2013
        lodsb
        cmp al, 20h
        ja short ci_mkdir_getarg2
        dec si
        xor al, al
        mov byte ptr [SI], al

        mov si, offset Msg_Making_Directory
        call UNIX_PRINTMSG

        call make_directory
        jc ci_error

        jmp ci_sync_exit
; RMDIR
loc_cmd_rmdir:
        ; 05/01/2013
        cmp ax, 'mr'
        jne short loc_cmd_namei ; 17/02/2013
        lodsw
        cmp ax, 'id'
        jne short @b
        lodsb
        cmp al, 'r'
        jne short @b
        lodsb
        or al, al
        jnz short @b
ci_rmdir_getarg1:
        mov word ptr [u_namep], si
        lodsb
        cmp al, 20h
        je short ci_rmdir_getarg1
        jnb short @b
        ; 06/01/2013
        mov ah, al

```

```

ci_rmdir_getarg2:
    lodsb
    cmp al, 20h
    ja short ci_rmdir_getarg2
    dec si
    xor al, al
    mov byte ptr [SI], al

    mov al, '.'
    cmp ah, al ; dot
    jne short @f

    mov ah, byte ptr [SI]
    cmp ah, 21h
    jb ci_error

    cmp ah, al ; '.' ; dotdot (parent dir)
    jne short @f

    inc si
    cmp byte ptr [SI], 21h
    jb ci_error
@@:
    ; u_namep = pointer to directory path name

    call name_i
    jc ci_error

    cmp ax, ROOT_DIR_INODE_NUMBER
    je ci_error

    cmp ax, word ptr [u_cdir]
    je ci_error

    push ax
    mov si, offset Msg_Removing_Directory
    call UNIX_PRINTMSG
    pop ax

    call remove_directory
    jc ci_error

    jmp ci_sync_exit

; 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 short @f
    lodsb
    cmp al, 'i'
    jne short @f
    lodsb
    or al, al
    jnz short @f
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
    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 @b
    lodsw
    cmp ax, 'do'
    jne short @b
    lodsb
    cmp al, 'e'
    jne short @b
    lodsb
    or al, al
    jnz short @b
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
cl3:
    cmp cl, 3
    jnb short cl2

; DIR
loc_cmd_dir:
    ; 05/01/2013 @b->@f, dir_print modifications
    cmp ax, 'id'
    jne short @f
    lodsb
    cmp al, 'r'
    jne short @f
    lodsb
    or al, al
    jnz short @f
    mov byte ptr [ls_option], al ; 20/01/2013
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 @b
    cmp ah, 0
    je ci_?
@@:
    retn

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

; CD (CHDIR)
loc_cmd_cd:
    cmp ax, 'dc'
    jne short loc_cmd_ls
    lodsb
    or al, al
    jnz short @b
    jmp ci_cd_getarg

; LS (DIR)
loc_cmd_ls:
    ; 20/01/2013
    cmp ax, 'sl'
    jne short loc_cmd_rm
    lodsb
    or al, al
    jnz short @b
    mov byte ptr [ls_option], 1
ls_getarg1:
    ; 21/01/2013
    lodsb
    cmp al, 20h
    je short ls_getarg1
    jnb short ls_getarg3

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

ls_getarg4:
    lodsb
    inc byte ptr [ls_option]
    cmp al, 20h
    je short dir_getarg
    jnb short ls_getarg3
    dec byte ptr [ls_option]
    jmp short ls_getarg3

; RM
loc_cmd_rm:
    cmp ax, 'mr'
    jne loc_cmd_fs ; 16/02/2013
    lodsb
    or al, al
    jnz short @b

rm_getarg:
    mov word ptr [u_namep], si
    lodsb
    cmp al, 20h
    je short rm_getarg
    jnb short @b

rm_namei:
    call name_i
    jc short ci_error

    ; cmp word ptr [ii], 41    ; i-number of the directory
    ; jne short @f

    mov si, offset BSBuffer + bs_BF_inode_Number
    cmp ax, word ptr [SI]    ; is it i-number of the boot file
    ; je short ci_error
    jne short @f

```

```

        cmp word ptr [ii], 41    ; i-number of root directory
        je ci_error

@@:
        mov word ptr [SI], 0    ; reset wrong boot file configuration
        mov word ptr [uf_i_number], ax ; word ptr [u_dirbuf]
        ; 05/01/2013
        mov dx, word ptr [ii]
        mov word ptr [pdir], dx
        call i_get
        jc ci_error
        mov ax, word ptr [inode_flg]
        test ah, 40h ; 'directory' flag
        jnz ci_error
        test al, 4h
        jz ci_error ; 'write' flag
        ;
rm_move_fn:
        mov si, offset u_dirbuf + 2
        mov di, offset Boot_File_Name
        mov cx, 8

@@:
        lodsb
        and al, al
        jz short @f
        stosb
        loop @b
        xor al, al ; 06/01/2013

@@:
        mov byte ptr [DI], al ; 0

        mov si, offset msg_remove_question1
        call UNIX_PRINTMSG
        mov si, offset Boot_File_Name
        call UNIX_PRINTMSG
        mov si, offset msg_remove_question2
        call UNIX_PRINTMSG
        mov si, offset msg_yes_no
        call UNIX_PRINTMSG
rm_yn_input:
        ; ask for remove
        xor ax, ax
        int 16h                                ; wait for keyboard command
        cmp al, 'C'-40h
        je short @f
        cmp al, 27
        je short @f
        and al, 0DFh
        cmp al, 'Y'                            ; Yes?
        je short rm_a_yes                       ; overwrite
        cmp al, 'N'                            ; No?
        jne short rm_yn_input

rm_a_no:
        mov si, offset msg_No
        call UNIX_PRINTMSG
        mov si, offset UNIX_CRLF
        call UNIX_PRINTMSG

@@:
        retn

rm_a_yes:
        mov si, offset msg_YES
        call UNIX_PRINTMSG
        mov si, offset UNIX_CRLF
        call UNIX_PRINTMSG

rm_unlink:
        mov si, offset Msg_removing_file
        call UNIX_PRINTMSG

        mov ax, word ptr [uf_i_number]
        call unlink
        jc ci_error

        jmp ci_sync_exit

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

```

```

vol_info_print:
fs_info_print:
    call print_volume_info
@@:
    retn
cl6: ; 16/02/2013
    cmp cl, 6
    jne short @b

; VOLUME (fs) ; 16/02/2013
loc_cmd_volume:
    cmp ax, 'ov'
    jne short @b
    lodsw
    cmp ax, 'ul'
    jne short @b
    lodsw
    cmp ax, 'em'
    jne short @b
    lodsb
    or al, al
    jz short vol_info_print
@@:
    retn
; 15/12/2012
cl7:
    cmp cl, 7
    jb cl6 ;16/02/2013
    ja cl8

; FROMDOS
loc_cmd_fromdos:
    cmp ax, 'rf'
    jne short @b
    lodsw
    cmp ax, 'mo'
    jne short @b
    lodsw
    cmp ax, 'od'
    jne short @b
    lodsb
    cmp al, 's'
    jne short @b
    lodsb
    or al, al
    jnz short @b

fromdos_df1:
    mov word ptr [arg], si
    lodsb
    cmp al, 20h
    je short fromdos_df1
    jb short @b

fromdos_df2:
    lodsb
    cmp al, 20h
    ja short fromdos_df2
    xor al, al
    mov byte ptr [SI]-1, al

fromdos_uf1:
    mov word ptr [u_namep], si
    lodsb
    cmp al, 20h
    je short fromdos_uf1
    jb short @b

fromdos_uf2:
    lodsb
    cmp al, 20h
    ja short fromdos_uf2
    dec si
    xor al, al
    mov byte ptr [SI], al

fromdos_fdf:
    mov dx, word ptr [arg]
    mov cx, 27h ; File Attributes
    mov ah, 4Eh ; MS Dos Function = Find First File
    int 21h
    jc ci_error ; file not found

```

```

fromdos_fuf:
    call name_i
    jnc short @f

    cmp ah, 0FFh
    jne ci_error

    xor ax, ax
    mov word ptr [uf_i_number], ax
    jmp fromdos_s_fs_mdt

@@:
    mov word ptr [uf_i_number], ax
    ; 05/01/2013
    mov dx, word ptr [ii]
    mov word ptr [pdir], dx
    call i_get
    jc ci_error
    mov ax, word ptr [inode_flg]
    test ah, 40h ; 'directory' flag
    jnz ci_error
    test al, 4h ; 'write' flag
    jz ci_error
    ;

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

@@:
    lodsb
    and al, al
    jz short @f
    stosb
    loop @b
    xor al, al ; 01/03/2013

@@:
    mov byte ptr [DI], al ; 0

    mov si, offset msg_overwrite_question1
    call UNIX_PRINTMSG
    mov si, Offset Boot_File_Name
    call UNIX_PRINTMSG
    mov si, offset msg_overwrite_question2
    call UNIX_PRINTMSG
    mov si, offset msg_yes_no
    call UNIX_PRINTMSG

fromdos_afow_input:    ; ask for overwrite
    xor ax, ax
    int 16h                ; wait for keyboard command
    cmp al, 'C'-40h
    je short @f
    cmp al, 27
    je short @f
    and al, 0DFh
    cmp al, 'Y'            ; Yes?
    je short fromdos_afow_yes ; overwrite
    cmp al, 'N'            ; No?
    jne short fromdos_afow_input

fromdos_afow_no:
    mov si, offset msg_No
    call UNIX_PRINTMSG
    mov si, offset UNIX_CRLF
    call UNIX_PRINTMSG

@@:
    retn

fromdos_afow_yes:
    mov si, offset msg_YES
    call UNIX_PRINTMSG
    mov si, offset UNIX_CRLF
    call UNIX_PRINTMSG

fromdos_uf_itrunc:
    ; 05/01/2013
    mov ax, word ptr [pdir]
    call i_get
    jc ci_error
    ;
    mov ax, word ptr [uf_i_number]
    call itrunc ; truncate file
    jc ci_error

```

```

fromdos_s_fs_mdt:
    ; 15/12/2012
    ; Derived from UNIXBOOT.ASM (30/11/2012)
    ;mov     si, DTA_FileSize
    mov     si, DTA_FileSize+2
    ;mov     ax, word ptr [SI]
    ;mov     dx, word ptr [SI]+2
    ;or      ax, dx ; 64KB file size limit
    mov     ax, word ptr [SI]
    and     ax, ax
    jnz     ci_error

    ;mov     word ptr [file_Size], ax

    mov     si, DTA_Date
    mov     ax, word ptr [SI]
    push    ax
    and     ax, 00011111b ; Day Mask
    mov     word ptr [day], ax
    pop     ax
    mov     cl, 5
    shr     ax, cl ; shift right 5 times
    push    ax
    and     ax, 00001111b ; Month Mask
    mov     word ptr [month], ax
    pop     ax
    mov     cl, 4
    shr     ax, cl
    ;and     ax, 01111111b ; Result = Year - 1980
    add     ax, 1980
    mov     word ptr [year], ax

    mov     si, DTA_Time
    mov     ax, word ptr [SI]
    push    ax
    and     ax, 0000011111b ; Second Mask
    shl     al, 1
    mov     word ptr [second], ax
    pop     ax
    mov     cl, 5
    shr     ax, cl ; shift right 5 times
    push    ax
    and     ax, 0000111111b ; Minute Mask
    mov     word ptr [minute], ax
    pop     ax
    mov     cl, 6
    shr     ax, cl ; shift right 6 times
    ; (6+5=11)
    mov     word ptr [hour], ax ; ax = hours

    call    convert_to_epoch

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

fromdos_odf:
    mov     dx, word ptr [arg]
    mov     ah, 3Dh ; MS Dos Function = Open File
    xor     al, al
    int     21h
    jc      ci_error

    mov     word ptr [FileHandle], ax

    mov     ax, word ptr [uf_i_number]
    ; 23/02/2013
    and     ax, ax
    jz      short @f
    ; jnz short fromdos_wf_msg ;@f
    xor     ax, ax
    call    fromdos_maknod
    jmp     short fromdos_wf_msg
@@: ; fromdos_mknod:
    mov     ax, i_flags ; 1Eh
    call    mak_nod
    jc      short fromdos_cf

fromdos_wf_msg:
    mov     si, offset Msg_writing_file
    call    UNIX_PRINTMSG
@@:
    ; 16/12/2012
    xor     ax, ax

```

```

        mov word ptr [u_off], ax
@@:
;loc_read_dos_sf:
        mov ah, 3Fh ; Read File
        mov cx, 512
        mov dx, offset ReadBUFFER
        mov bx, word ptr [FileHandle]
        int 21h
        jc short fromdos_cf
        or ax, ax
        jz short fromdos_cf
        mov word ptr [u_count], ax
        mov word ptr [u_base], offset ReadBuffer
        mov ax, word ptr [uf_i_number] ; word ptr [u_dirbuf]
        call write_i
        jnc short short @b ; loc_read_dos_sf

fromdos_cf:
        pushf
        mov ah, 3Eh ; Close File
        mov bx, word ptr [FileHandle]
        int 21h
        popf
        jc ci_error
@@:
        ; 23/02/2013
        xor ax, ax
        mov word ptr [inode_mtim], ax
        mov word ptr [inode_mtim]+2, ax
        call set_imod

ci_sync_exit:
        call sync
        jc ci_error

fromdos_retn:
        mov si, offset Msg_OK
        call UNIX_PRINTMSG
        mov si, offset UNIX_CRLF
        call UNIX_PRINTMSG
@@:
        retn

cl8:
        cmp cl, 8
        ;jb short @b
        ja cl10

; BOOTFILE
loc_cmd_bootfile:
        cmp ax, 'ob'
        jne short @b
        lodsw
        cmp ax, 'to'
        jne short @b
        lodsw
        cmp ax, 'if'
        jne short @b
        lodsw
        cmp ax, 'el'
        jne short @b
        lodsb
        or al, al
        jnz short @b
@@:
        mov word ptr [u_namep], si
        lodsb
        cmp al, 20h
        je short @b
        ja short ci_bf_namei
        mov si, offset BSBuffer + 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
ci_bf_namei:
    call name_i
    jc ci_error

    cmp word ptr [ii], ROOT_DIR_INODE_NUMBER
    jne ci_error

    ; 05/01/2013
    ; ax = i-number of (new) boot file
    call i_get
    jc ci_error
    test word ptr [inode_flg], 4000h ; directory ?
    jnz ci_error
@@:
    mov si, offset BSBuffer + bs_BF_inode_Number
    mov word ptr [SI], ax

    call sync
    jc ci_error

    mov si, offset msg_sf_configuration_set_ok
    call UNIX_PRINTMSG
@@:
    retn
cll10:
    cmp cl, 10
    jne short @f
; NOBOOTFILE
loc_cmd_nobootfile:
    cmp ax, 'on'
    jne short @f
    lodsw
    cmp ax, 'ob'
    jne short @f
    lodsw
    cmp ax, 'to'
    jne short @f
    lodsw
    cmp ax, 'if'
    jne short @f
    lodsw
    cmp ax, 'el'
    jne short @f
    lodsb
    or al, al
    jnz short @f
    mov si, offset BSBuffer + bs_BF_inode_Number
    and ax, ax
    jz ci_no_bootfile
    xor ax, ax
    mov word ptr [SI], ax
    call sync
    jc ci_error
    mov si, msg_sf_configuration_reset_ok
    call UNIX_PRINTMSG
@@:
    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:
    ;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
    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_flg]
    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:
    ; 21/02/2013
    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 ; jnb 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

make_directory proc near
    ; 30/12/2012
    ;
    ; mov word ptr [u.namep], si

    call sys_mkdir
    jc short @f

    ;ax = i-number

    ;mov ax, word ptr [ii]
    ;mov word ptr [u_dirbuf], ax

    mov word ptr [u_namep], offset dot
    xor ax, ax
    mov word ptr [u_dirp], ax ; 0
    call mk_dir      ; make a directory entry
                    ; in current (ii) directory
    jc short @f

    mov word ptr [u_dirp], 10
    mov ax, word ptr [pdir]
    mov word ptr [u_dirbuf], ax
    mov word ptr [u_namep], offset dotdot

    call mk_dir

@@:
    retn

make_directory endp

```

```

sys_mkdir proc near
; 05/01/2013 (bugfix)
; 30/12/2012 unixcopy.asm
; Retro UNIX v1 FS file import/export version
; of sysmkdir function
; Derived from (original) UNIX v1 source code
; PRELIMINARY release of Unix Implementation Document,
; 20/6/1972
;
; RETRO UNIX v1 FS
; sysmkdir:
; make a directory
;
;
; return => if cf=1 error code in AH
; If cf=0 -> AX = I-Number (also in u.dirbuff)

;jsr    r0,arg2 / point u.namep to the file name
;jsr    r0,namei / get the i-number
;      br .+4 / if file not found branch around error
;br     error2 / directory already exists (error)
;tstb   u.uid / is user the super user
;bne    error2 / no, not allowed
;mov    (sp)+,r1 / put the mode in r1
;bic    $!317,r1 / all but su and ex
;bis    $40000,r1 / directory flag
;jsr    r0,maknod / make the i-node for the directory
;br     sysret2 /

mov bx, word ptr [u_namep]
mov si, bx ; 05/01/2013

makdir_1:
lodsb
or al, al
jz short makdir_2
cmp al, '/'
jne short makdir_1
mov bx, si
jmp short makdir_1

makdir_2:
cmp bx, word ptr [u_namep]
je short makdir_3
dec si
dec si ; 05/01/2013
cmp byte ptr [SI], '/' ; is the last char '/'
jne short makdir_3

cmp si, word ptr [u_namep] ; 05/01/2013
je short makdir_3

stc

@@:
retn

makdir_3:
mov word ptr [pdir], bx
sysmkdir_0:
call name_i
jc short sysmkdir_1

stc

@@:
retn
sysmkdir_1:
cmp ah, 0FFh
jne short @@

makdir_4:
mov ax, word ptr [ii]
mov bx, word ptr [pdir]
mov word ptr [pdir], ax
cmp word ptr [u_namep], bx
jb short @b ; parent dir of the new sub dir not found

```

```

sysmkmdir_flags: ; ax = r1 = mode
    mov ax, 0C00Eh ; Flags (1100000000001110b)
sysmkmdir_maknod:
    call mak_nod
    ; ax = I-Number (also in u.dirbuff)
    retn

sys_mkdir endp

remove_directory proc near
    ; 05/01/2013
    ; mov word ptr [u.namep], si
    ; call name_i
    ; jc @f
    ; cmp ax, ROOT_DIR_INODE_NUMBER
    ; je rmdir_stc_retn
    ; cmp ax, word ptr [u_cdir]
    ; je rmdir_stc_retn
    ; INPUT ->
    ; ax = i_number of directory (to be removed)
    ; u_off = directory entry location + 10 (in parent dir)
    ; [ii] = i_number of parent directory
    cmp ax, word ptr [ii] ; '.' entry
    je rmdir_stc_retn
    mov word ptr [uf_i_number], ax ; i_number of dir or file
    mov dx, word ptr [u_off]
    mov word ptr [FileHandle], dx ; directory entry location + 10
    mov dx, word ptr [ii] ; i-number of parent directory
    mov word ptr [pdir], dx
    call i_get
    jc short @f
    mov ax, word ptr [inode_flg]
    test ah, 40h ; 'directory' flag
    jz short rmdir_stc_retn
    test al, 4h ; 'write' flag
    jz short rmdir_stc_retn
    xor ax, ax
    mov word ptr [u_off], ax
    ; mov word ptr [u_fofp], offset u.off
rmdir_readi_loop:
    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
    ; i.e. read a directory entry
    jc short @f
    mov cx, word ptr [u_nread]
    or cx, cx
    ; jna short rmdir_stc_retn
    jna short @f
    ; cmp cx, 10
    ; jnb short @f
    mov bx, word ptr [u_dirbuf]
    and bx, bx
    jz short rmdir_readi_loop
    mov ax, word ptr [u_dirbuf]+2
    cmp al, '.'
    jne short rmdir_stc_retn
    and ah, ah
    jz short rmdir_readi_loop
    cmp ah, '.' ; '..'
    jne short rmdir_stc_retn
    mov ah, byte ptr [u_dirbuf]+4
    or ah, ah
    jnz short rmdir_stc_retn
    mov ax, word ptr [u_off]
    cmp ax, 10 ; protection for removing default system directories
    jna short rmdir_stc_retn ; because, the 1st dir entry of them is '..'
    cmp ax, word ptr [inode_size]
    jnb short rmdir_readi_loop
rmdir_unlink:
    mov ax, word ptr [uf_i_number]
    mov dx, word ptr [FileHandle]
    mov word ptr [u_off], dx
    call unlink
@@:
    retn

```

```

rmdir_stc_retn:
    stc
    retn

remove_directory endp

show_file proc near
    ; 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 word ptr [u_base], offset ReadBuffer
    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, offset ReadBuffer
    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
;     jnb 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 @@
    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_flg], 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
        ; 01/03/2013
        ; 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
        jc short read_inode_3 ; 01/03/2013
        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
        jc short read_inode_3 ; 01/03/2013
        ; 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 ; offset WriteBuffer ; 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 @@ ; 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

rep movsb

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
; 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:
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
; dl = 0 = read
call i_calc

iget_3:
mov ax, word ptr [ii]

iget_4:
retn

i_get endp

i_calc proc near
; 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_4

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, offset WriteBuffer

add si, dx

cmp byte ptr [rw], 0
jna short icalc_3 ; 0 = read (and copy i-node to memory)

icalc_2:
xchg si, di
; over write old i-node (in buffer to be written)
rep movsw

mov ax, word ptr [buff_s] ; 18/11/2012

call dsk_wr

retn

icalc_3:
; copy new i-node into inode area of (core) memory
rep movsw

icalc_4:
retn

i_calc endp

dsk_rd proc near
; 07/07/2015 (floppy disk image file handling)
; 06/03/2013
; 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, offset WriteBuffer ; 28/11/2012

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

```

```

    mov     si, ax

    cmp     byte ptr [PhysicalDriveNumber], 90h ; fd image file sign
    jnb     short image_file_rd

    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

image_file_rd proc near
; 14/07/2015
; 07/07/2015
; reading a block (sector) from floppy disk image file
; INPUTS:
;     ax = si = sector/block number
;     bx = offset WriteBuffer = buffer address
;     [img_file_handle] = file handle
;     number of bytes to be written = 512
;
    mov     dx, 512
    mul     dx
;push    bx
    mov     cx, dx
    mov     dx, ax
    sub     al, al ; specified offset is from the beginning of the file
    mov     ah, 42h ; seek (move file pointer)
    mov     bx, word ptr [img_file_handle]
    int     21h
;pop     bx
    jc     short image_file_rd_err
;mov     dx, bx
    mov     bx, word ptr [img_file_handle]
    mov     cx, 512
    mov     dx, offset WriteBuffer
    mov     ah, 3Fh ; read from file
    int     21h
    jc     short image_file_rd_err
    mov     bx, dx
;xor     dx, dx
;cmp     ax, cx ; ax = actually written bytes
;jb     short image_file_rd_err
    mov     word ptr [buff_s], si ; current buffer sector
image_file_rd_err:
    retn
image_file_rd endp

```

```

m_get  proc near
; 05/03/2013
; 03/03/2013
; 01/03/2013
; 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)

;push bx
;push cx
;push dx
;; 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_5 ; 4f ; large file

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

and bl, 0Eh ; clear all bits but bits 1,2,3
mov ax, word ptr inode_dskp[BX] ; AX = R1, physical block number
or ax, ax
jnz short mget_1 ; if physical block number is zero
; then need a new block for file
call sb_alloc ; allocate a new block for this file
; AX (R1) = Block number
jc short mget_8 ; ; cf -> 1 & ax = 0 -> no free block

mov word ptr inode_dskp[BX], ax

call set_imod

call b_clear

mget_1: ; 2
; AX (R1) = Physical block number

;pop dx
;pop cx
;pop bx

ret

mget_2: ; 3
; adding on block which changes small file to large file
call sb_alloc
; call wslot ; setup I/O buffer for write
; ; R5 points to the first data word in buffer

; push ds
; pop es

mov word ptr [buff_s], ax ; Block/Sector number

;push si
;push di
push ax

mov cx, 8 ; R3, transfer old physical block pointers
; into new indirect block area for the new
; large file
mov di, offset WriteBuffer ; BX = R5

```

```

        mov si, offset inode_dskp

        xor ax, ax ; mov ax, 0
mget_3: ; 1
        movsw
        mov word ptr [SI]-2, ax
        loop mget_3

        mov cl, 256-8 ; clear rest of data buffer

mget_4: ; 1
        rep stosw

        pop ax
        ;pop di
        ;pop si

        ;mov byte ptr [buff_m], 1 ; modified
        call dsk_wr
        jc short mget_1

        mov ax, word ptr [buff_s]

        mov word ptr [inode_dskp], ax
        or word ptr [inode_flg], 4096 ; 1000h

        call set_imod

        jmp short mget_0

mget_9:
        pop ax
mget_8:
        ;mov ax, err_NOFREEBLOCK

        ;pop dx
        ;pop cx
        ;pop bx

        retn

mget_5: ; 4 ; large file
        ; 05/03/2013
        ;mov ax, bx ; bx <= 255 for this file (UNIX v1, RUFS) system
        ;mov cx, 256 ; 01/03/2013 no need a division here
        ;xor dx, dx ; 01/03/2013 no need a division here
        ;div cx ; 01/03/2013 no need a division here
        ;and bx, 1FEh ; zero all bit but 1,2,3,4,5,6,7,8
        ; ; gives offset in indirect block
        ;push bx ; R2
        ;mov bx, ax ; calculate offset in i-node for pointer
        ; ; to proper indirect block
        ;and bx, 0Eh
        ;mov ax, word ptr inode_dskp[BX] ; R1

        and bl, 0FEh ; ah = 0 ; 01/03/2013
        push bx ; i-node pointer offset in indirect block
        ; 01/03/2013 Max. possible AX (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 ; R1
        jnz short mget_6 ; 2f

        call sb_alloc
        jc short mget_9 ; 01/03/2013

        ;mov word ptr inode_dskp[BX], ax ; R1, block number
        mov word ptr [inode_dskp], ax ; 03/03/2013

        call set_imod

        call b_clear

mget_6: ;2
        ; 05/03/2013
        ; ax = R1, block number

```

```

    call dsk_rd ; read indirect block
    pop dx ; R2, get offset
    jc short mget_7
    ; BX = offset WriteBuffer
    add bx, dx ; R5, first word of indirect block
    mov ax, word ptr [BX] ; put physical block no of block
                        ; in file sought in R1 (AX)

    or ax, ax
    jnz short mget_7 ; 2f

    call sb_alloc
    jc short mget_8 ; 01/03/2013

    mov word ptr [BX], ax ; R1

    push ax
    mov ax, word ptr [buff_s]

    ;mov byte ptr [buff_m], 1 ; modified

    ;call wslot
    call dsk_wr
    pop dx ; 18/11/2012
    jc short mget_7

    mov ax, dx ; 18/11/2012
    ; ax = R1, block number of new block

    call b_clear

mget_7: ; 2
    ; ax = R1, block number of new block
    ;pop dx
    ;pop cx
    ;pop bx

    retn

m_get endp

sb_alloc proc near
    ; 14/11/2012 unix boot file configuration version
    ; of "alloc" procedure
    ; 21/8/2012
    ; 18/8/2012
    ; 17/8/2012
    ; 5/8/2012
    ; 21/7/2012
    ; Derived from (original) UNIX v1 source code
    ; PRELIMINARY release of Unix Implementation Document,
    ; 20/6/1972
    ;; input -> AX=R1
    ;; output -> AX=R1
    ; RETRO UNIX v1 FS

    ;push cx
    push bx ; R2
    ;push dx ; R3

    mov bx, offset systm ; SuperBlock
                        ; start of inode and free storage map for disk
alloc_1: ; 1
    mov ax, word ptr [BX] ; first word contains # of bytes
                        ; in free storage map
    shl ax, 1 ; multiply AX (R1) by 8 gives # of blocks
    shl ax, 1
    shl ax, 1
    mov cx, ax ; R1, bit count of free storage map
    xor ax, ax ; 0
alloc_2: ; 1
    inc bx ; 18/8/2012
    inc bx ;
    mov dx, word ptr [BX] ; mov (R2)+, R3
    or dx, dx
    jnz short alloc_3 ; 1f
                        ; branch if any free blocks in this word
    add ax, 16
    cmp ax, cx
    jb short alloc_2 ; 1b

```

```

; jmp short panic ; no free storage

xor ax, ax
stc          ; cf=1 --> error: no free block

jmp short alloc_7

alloc_3: ; 1
        shr dx, 1 ; R3 ; Branch when free block found,
                ; bit for block k is in byte k/8
                ; in bit k (mod 8)
        jc short alloc_4 ; 1f
        inc ax ; R1 ; increment bit count in bit k (mod 8)
        jmp short alloc_3 ; 1b

alloc_4:
        ; call free_3
sb_alloc_free_3:
        mov dx, 1
        mov cx, ax
        and cx, 0Fh
        jz short @f
        shl dx, cl
@@:
        mov bx, ax
        shr bx, 1
        shr bx, 1
        shr bx, 1
        shr bx, 1
free_4: ; 1
        shl bx, 1 ; 21/8/2012
        ; BX (R2) = k/8
        add bx, offset systm+2 ; SuperBlock+2

alloc_5: ; 1
        ; 21/8/2012
        not dx ; masking bit is '0' and others are '1'
        and word ptr [BX], dx ; bic r3, (r2)
        ; 0 -> allocated retn

alloc_6:
        ; inc byte ptr [smod] ; super block modified sign
        ; mov byte ptr [smod], 1

alloc_7:
        ; pop dx ; R3
        pop bx ; R2
        ; pop cx
        ; AX (R1) = Block number
        retn

sb_alloc endp

set_imod proc near
        ; 23/02/2013 (fromdos) file m. date&time modification
        ; 14/11/2012 unix boot file configuration version
        ; of "setimod" procedure
        ; 13/8/2012
        ; 21/7/2012
        ; 14/7/2012
        ; Derived from (original) UNIX v1 source code
        ; PRELIMINARY release of Unix Implementation Document,
        ; 20/6/1972
        ; AX=R0, BX=R1, CX=R3, DX=R5
        ; [SP] = Argument 1, 0 = read, 1 = write
        ; RETRO UNIX v1 FS
        ; initialization/format version
        ;

        ; push dx
        push ax
        mov byte ptr [imod], 1
        ; 23/02/2013
        mov ax, word ptr [inode_ctim]
        mov dx, word ptr [inode_ctim]+2
        and ax, ax
        jnz short setimod_3
        and dx, dx
        jnz short setimod_3

```

```

setimod_1:
; Erdogan Tan 14-7-2012
call epoch
mov word ptr [inode_ctim], ax
mov word ptr [inode_ctim]+2, dx
setimod_2:
mov word ptr [inode_mtim], ax
mov word ptr [inode_mtim]+2, dx
setimod_4:
pop ax
;pop dx
retn
setimod_3:
; 23/02/2013
xor cx, cx
cmp word ptr [inode_mtim], cx
jna short setimod_2
cmp word ptr [inode_mtim]+2, cx
jna short setimod_2
call epoch
jmp short setimod_2

set_imod endp

b_clear proc near
; 18/11/2012
; 14/11/2012 unix boot file configuration version
; of "clear" procedure
; 5/8/2012
; 21/7/2012
; Derived from (original) UNIX v1 source code
; PRELIMINARY release of Unix Implementation Document,
; 20/6/1972
;; input -> AX=R1 (block number)
;; output -> AX=R1
; RETRO UNIX v1 FS
; initialization/format version

;call wslot ; setup I/O buffer for write
; ; R5 points to the first data word in buffer
; BX = R5

mov word ptr [buff_s], ax

;push ds
;pop es

;push di
;push cx
push ax
xor ax, ax
; mov di, bx
mov di, offset WriteBuffer
mov cx, 256
rep stosw

;mov byte ptr [buff_m], 1 ; modified

mov ax, word ptr [buff_s] ; 18/11/2012

call dsk_wr

pop ax
;pop cx
;pop di

retn

b_clear endp

mak_nod proc near
; 01/03/2013
; 23/02/2013
; 15/12/2012 UNIXCOPY.ASM version of maknod
; 02/12/2012 (maknod_imap -> call imap)
; 25/11/2012
; 18/11/2012
; 11/11/2012

```

```

; unixboot.asm (boot file configuration)
; version of 'maknod'
;
; 30/10/2012
; AX = R1, mode
; Derived from (original) UNIX v1 source code
; PRELIMINARY release of Unix Implementation Document,
; 20/6/1972
; RETRO UNIX v1 FS
;
; maknod : create an i-node and make a directory entry
;
; 8086 CPU & IBM PC architecture modifications by Erdogan Tan
;
; return => if cf=1 error code in AH
; If cf=0 -> AX = I-Number (also in u.dirbuff)

or      ah, 80h ; 10000000b, allocate flag set
push    ax      ; put mode on stack
;mov    ax, word ptr [ii] ; move current i-number to AX/r1
;push   ax
;push   word ptr [ii]

maknod_1: mov    ax, 41 ; r1 = 41
; scan for a free i-node
inc     ax      ; r1 = r1 + 1
; 2/12/2012
call    imap    ; get byte address and bit position in inode map in
; r2 (DX) & mq (BX)
; DX (MQ) has a 1 in the calculated bit position
; BX (R2) has byte address of the byte with allocation bit

test    byte ptr [BX], dl ; bitb mq, (r2) / is the i-node active
jnz     short maknod_1    ; bne 1b / yes, try the next one
or      byte ptr [BX], dl ; bisb mq, (r2)
; no, make it active (put a 1 in the bit map)

; ax = i-number
call    i_get    ; jsr r0,iget / get i-node into core
jc      short maknod_3

test    word ptr [inode_flgsl], 8000h ; is i-node already allocated
jnz     short maknod_1    ; 1b / yes, look for another one

mov     word ptr [u_dirbuf], ax ; mov r1, u.dirbuf
; no, put i-number in u.dirbuf

pop     ax ; 15/12/2012 ; get currrent i-number back
call    i_get    ; jsr r0,iget / get i-node in core
jc      short maknod_2

call    mk_dir    ; jsr r0,mkdir
; make a directory entry in current directory
jc      short maknod_2 ; 01/03/2013

mov     ax, word ptr [u_dirbuf] ; mov u.dirbuf, r1
; ax / r1 = new inode number

call    i_get
jc      short maknod_2

; jsr r0,copyz; inode; inode+32. / 0 it out
mov     cx, 16
xor     ax, ax ; 0
mov     di, offset inode
rep     stosw

pop     word ptr [inode_flgsl] ; mov (sp)+,i.flgsl / fill flags
mov     cl, byte ptr [u_uid] ; movb u.uid,i.uid / user id
mov     byte ptr [inode_uid], cl ; 23/02/2013 al -> cl
mov     byte ptr [inode_nlksl], 1 ; movb $1,i.nlksl / 1 link

;call    epoch

;mov     word ptr [s_time], ax
;mov     word ptr [s_time]+2, dx

;mov     word ptr [inode_ctim], ax ; mov s.time,i.ctim / time created
;mov     word ptr [inode_ctim]+2, dx ; mov s.time+2,i.ctim+2

```

```

; 25/11/2012
; 23/02/2013
fromdos_maknod:
; xor    ax, ax
xor    dx, dx
mov    word ptr [inode_mtim], ax ; 0
mov    word ptr [inode_mtim]+2, dx ; 0
test   word ptr [inode_flg], 4000h ; Directory
jnz    short maknod_4
mov    ax, word ptr [uf_make_datetime]
mov    dx, word ptr [uf_make_datetime]+2
maknod_4:
mov    word ptr [inode_ctim], ax
mov    word ptr [inode_ctim]+2, dx

call   set_imod

mov    ax, word ptr [u_dirbuf]
retn

maknod_3:
; 15/12/2012
pop    ax
maknod_2:
pop    ax
retn

mak_nodendp

mk_dir proc near
; 11/11/2012
; unixboot.asm (boot file configuration)
; version of 'mkdir'
;
; 31/10/2012
; Derived from (original) UNIX v1 source code
; PRELIMINARY release of Unix Implementation Document,
; 20/6/1972
; RETRO UNIX v1 FS
;
; mkdir : make a directory entry
;
; 8086 CPU & IBM PC architecture modifications by Erdogan Tan
;
; return => if cf=1 error number in [Error], ax = mode
; If cf=0 -> AX = I-Number (also in u.dirbuff)
;
; input:
; u.namep = file name
; ii = current directory's i-number
; u.dirbuf = directory entry (source) location
; output:
; u.dirbuf+2 to u.dirbuf+10 = file name
; u.off = directory entry offset in current directory
; u.base = start of u.dirbuf
; ;;r1 (AX) = i-number of current directory

mkdir_0:
; jsr r0,copyz; u.dirbuf+2; u.dirbuf+10. / clear this
mov    cx, 4
xor    ax, ax
mov    di, offset u_dirbuf+2
rep    stosw

mov    si, word ptr [u_namep] ; mov u.namep,r2
; r2 points to name of directory entry

mov    di, offset u_dirbuf+2 ; mov $u.dirbuf+2,r3
; r3 points to u.dirbuf+2
mkdir_1: ; 1 / put characters in the directory name in u.dirbuf+2 - u.dirbuf+10
lodsb   ; movb (r2)+,r1 / move character in name to r1
and     al, al
jz      short mkdir_2 ; beq 1f / if null, done
cmp     al, '/' ; cmp r1,$'/' / is it a "/"?
je      short mkdir_stc ; beq error9 / yes, error
cmp     di, offset u_dirbuf+10 ; cmp r3,$u.dirbuf+10.
; have we reached the last slot for
; a char?
je      short mkdir_1 ; beq 1b / yes, go back

```

```

        stosb                                ; movb r1,(r3)+
                                           ; no, put the char in the u.dirbuf
        jmp     short mkdir_1                ; br 1b / get next char

mkdir_2: ;1
        mov     ax, word ptr [u_dirp] ; mov u.dirp,u.off
        mov     word ptr [u_off], ax  ; pointer to empty current directory
                                           ; slot to u.off
wdir:
        mov     word ptr [u_base], offset u_dirbuf
                                           ; mov $u.dirbuf,u.base
                                           ; u.base points to created file name
        mov     word ptr [u_count], 10 ; mov $10.,u.count
                                           ; u.count = 10
        mov     ax, word ptr [ii]      ; mov ii,r1
                                           ; r1 has i-number of current directory

        call    write_i                ; jsr r0,writei / write into directory
@@:
        retn                             ; rts r0

mkdir_stc:
        ; invalid file name, al="/", ah=0
        mov     ah, 1
        stc
        retn

mk_dir endp

write_i proc near
        ; 18/11/2012
        ; 11/11/2012
        ; unixboot.asm (boot file configuration)
        ; version of 'writei'
        ;
        ; 31/10/2012
        ; 18/08/2012
        ; 17/07/2012
        ; BX = R1, i-number
        ; Derived from (original) UNIX v1 source code
        ; PRELIMINARY release of Unix Implementation Document,
        ; 20/6/1972
        ;
        ; RETRO UNIX v1 FS
        ; initialization/format version
        ;
        ; writei: write file
        ;
        ; 8086 CPU & IBM PC architecture modifications by Erdogan Tan
        ;; return => if cf=1 error number in [Error]

        ; input:
        ; AX = R1 = I-Number
        ; u.count = byte count
        ; u.base = user buffer (offset)
        ; u.off (u.fofp) = (pointer to) current file offset

        xor     dx, dx ; 0                ; clr u.nread
        mov     word ptr [u_nread], dx ; clear the number of bytes transmitted during
                                           ; read or write calls
                                           ; tst u.count
        cmp     word ptr [u_count], dx ; test the byte count specified by the user
        ja     short write_1 ; 1f        ; bgt 1f / any bytes to output; yes, branch
        retn                                         ; rts 0 / no, return - no writing to do
        jna     short write_inode_retn

write_1:
        ;push ax                                ; save i-number on stack

        call    i_get                ; jsr r0,iget
                                           ; write i-node out (if modified), read i-node 'r1'
                                           ; into i-node area of core
        mov     dx, word ptr [u_off]
        add     dx, word ptr [u_count]
                                           ; add u.count,r2
                                           ; no. of bytes to be written + file offset is
                                           ; put in r2

```

```

    cmp dx, word ptr [inode_size] ; cmp r2,i.size
                                ; is this greater than the present size of
                                ; the file?
    jna short dskw_1 ; blos      1f / no, branch

    mov word ptr [inode_size], dx ; mov r2,i.size
                                ; yes, increase the file size to file offset +
                                ; no. of data bytes
    call set_imod                ; jsr r0,setimod
                                ; set imod=1 (i.e., core inode has been
                                ; modified), stuff time of modification into
                                ; core image of i-node

dskw_1: ; 1
    call m_get                  ; jsr r0,mget
                                ; get the block no. in which to write the next data
                                ; byte
                                ; AX = R1 = Block Number

    mov bx, word ptr [u_off]
    and bx, 1FFh                ; bit *u.fofp,$777
                                ; test the lower 9 bits of the file offset
    jnz short dskw_2 ; bne 2f
                                ; if its non-zero, branch; if zero, file offset = 0,
                                ; 512, 1024,...(i.e., start of new block)
    cmp word ptr [u_count], 512 ; cmp u.count,$512.
                                ; if zero, is there enough data to fill an
                                ; entire block? (i.e., no. of
    jnb short dskw_3 ; bhis 3f / bytes to be written greater than 512.?
                                ; Yes, branch. / Don't have to read block
; 18/11/2012
    jb short dskw_2

    mov word ptr [buff_s], ax

    jmp short short dskw_3

dskw_2: ; 2
    ; in as no past info. is to be saved (the entire block will be
    ; overwritten).
                                ; AX=R1 (block number)
    call dsk_rd                 ; jsr r0,dskrd
                                ; no, must retain old info.. Hence, read block 'r1'
                                ; into an I/O buffer

; 11/11/2012
    jc short dskw_5
    mov ax,word ptr [buff_s]

dskw_3: ; 3
    ;call wslot

writeinode_sioreg:
    ; call sioreg

    mov di, word ptr [u_off] ; R2
    mov cx, di ; cx = R3, di = R2
    or cx, 0FE00h ; set bits 9...15 of file offset in R3
    and di, 1FFh ; calculate file offset mod 512
    add di, offset WriteBuffer ; di now points to 1st byte in buffer
                                ; where data is to be placed
    mov si, 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 writeinode_sioreg

    ; SI = user data offset (r1)
    ; DI = sector (I/O) buffer offset (r2)
    ; CX = byte count (r3)

```

```

dskw_4: ; 2
        rep movsb

        ; ax = block/sector number

        call dsk_wr ; jsr r0,dskwr / write the block and the i-node
        jc short dskw_5

        cmp word ptr [u_count], 0 ; any more data to write?
        ja short dskw_1 ; 1b ; yes, branch

dskw_5:
        ; pop ax ; i-number

write_inode_retn:
        retn

write_iendp

dsk_wr proc near
        ; 07/07/2015 (floppy disk image file handling)
        ; 11/11/2012
        ; unix boot file configuration version of "dskwr" procedure
        ;
        ; Derived from (original) UNIX v1 source code
        ; PRELIMINARY release of Unix Implementation Document,
        ; 20/6/1972
        ; RETRO UNIX v1 FS
        ;; return => if cf=1 error number in [Error]

        ; ax = sector/block number

        cmp byte ptr [PhysicalDriveNumber], 90h ; fd image file sign
        jnb short image_file_wr ; 07/07/2015

        mov bx, offset WriteBuffer

        xor ch, ch
        mov cl, 4 ; Retry count

dsk_wr_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, 3 ; 3=write
        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_wr_2
        loop dsk_wr_1

dsk_wr_2:
        retn

dsk_wr endp

```

```

image_file_wr proc near
; 14/07/2015
; 07/07/2015
; writing a block (sector) to floppy disk image file
; INPUTS:
;     ax = sector/block number
;     offset WriteBuffer = buffer address
;     [img_file_handle] = file handle
;     number of bytes to be written = 512
;
mov     dx, 512
mul     dx
mov     cx, dx
mov     dx, ax
sub     al, al ; specified offset is from the beginning of the file
mov     ah, 42h ; seek (move file pointer)
mov     bx, word ptr [img_file_handle]
int     21h
;mov    bx, word ptr [img_file_handle]
mov     cx, 512
mov     dx, offset WriteBuffer
mov     ah, 40h ; write to file
int     21h
jc      short image_file_wr_err
mov     bx, dx
xor     dx, dx
cmp     ax, cx ; ax = actually written bytes
image_file_wr_err:
    retn
image_file_wr endp

epoch proc near
; 14/11/2012
; unixboot.asm (boot file configuration)
; version of "epoch" procedure in "unixproc.asm"
; 21/7/2012
; 15/7/2012
; 14/7/2012
; Erdogan Tan - RETRO UNIX v0.1
; compute current date and time as UNIX Epoch/Time
; UNIX Epoch: seconds since 1/1/1970 00:00:00

; 21/7/2012
;push bx
;push cx

mov     ah, 02h                ; Return Current Time
int     1Ah
xchg    ch, cl
mov     word ptr [hour], cx
xchg    dh, dl
mov     word ptr [second], dx

mov     ah, 04h                ; Return Current Date
int     1Ah
xchg    ch, cl
mov     word ptr [year], cx
xchg    dh, dl
mov     word ptr [month], dx

mov     cx, 3030h

mov     al, byte ptr [hour] ; Hour
; AL <= BCD number)
db     0D4h, 10h                ; Undocumented inst. AAM
; AH = AL / 10h
; AL = AL MOD 10h

aad     ; AX= AH*10+AL

mov     byte ptr [hour], al

mov     al, byte ptr [hour]+1 ; Minute
; AL <= BCD number)
db     0D4h, 10h                ; Undocumented inst. AAM
; AH = AL / 10h
; AL = AL MOD 10h

aad     ; AX= AH*10+AL

mov     byte ptr [minute], al

```

```

mov al, byte ptr [second] ; Second
; AL <= BCD number)
db 0D4h,10h ; Undocumented inst. AAM
; AH = AL / 10h
; AL = AL MOD 10h

aad ; AX= AH*10+AL

mov byte ptr [second], al

mov ax, word ptr [year] ; Year (century)
push ax
; AL <= BCD number)
db 0D4h,10h ; Undocumented inst. AAM
; AH = AL / 10h
; AL = AL MOD 10h

aad ; AX= AH*10+AL

mov ah, 100
mul ah
mov word ptr [year], ax

pop ax
mov al, ah
; AL <= BCD number)
db 0D4h,10h ; Undocumented inst. AAM
; AH = AL / 10h
; AL = AL MOD 10h

aad ; AX= AH*10+AL

add word ptr [year], ax

mov al, byte ptr [month] ; Month
; AL <= BCD number)
db 0D4h,10h ; Undocumented inst. AAM
; AH = AL / 10h
; AL = AL MOD 10h

aad ; AX= AH*10+AL

mov byte ptr [month], al

mov al, byte ptr [month]+1 ; Day
; AL <= BCD number)
db 0D4h,10h ; Undocumented inst. AAM
; AH = AL / 10h
; AL = AL MOD 10h

aad ; AX= AH*10+AL

mov byte ptr [Day], al

convert_to_epoch:
; Derived from DALLAS Semiconductor
; Application Note 31 (DS1602/DS1603)
; 6 May 1998

mov dx, word ptr [year]
sub dx, 1970
mov ax, 365
mul dx
xor bh, bh
mov bl, byte ptr [month]
dec bl
shl bl, 1
mov cx, word ptr DMonth[BX]
mov bl, byte ptr [Day]
dec bl

add ax, cx
adc dx, 0
add ax, bx
adc dx, 0
; DX:AX = days since 1/1/1970

mov cx, word ptr [year]
sub cx, 1969
shr cx, 1
shr cx, 1

```

```

        ; (year-1969)/4
add ax, cx
adc dx, 0
        ; + leap days since 1/1/1970

cmp byte ptr [month], 2 ; if past february
jna short @f
mov cx, word ptr [year]
and cx, 3 ; year mod 4
jnz short @f
        ; and if leap year
add ax, 1 ; add this year's leap day (february 29)
adc dx, 0
@@:
        ; compute seconds since 1/1/1970
mov bx, 24
call proc_mul32

mov bl, byte ptr [hour]
add ax, bx
adc dx, 0

mov bx, 60
call proc_mul32

mov bl, byte ptr [minute]
add ax, bx
adc dx, 0

mov bx, 60
call proc_mul32

mov bl, byte ptr [second]
add ax, bx
adc dx, 0

; DX:AX -> seconds since 1/1/1970 00:00:00

; 21/7/2012
;pop cx
;pop bx

ret

epoch 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

sync    proc near
; 14/07/2015
; 07/07/2015
; 18/11/2012 unix boot file configuration version
; of "sync" procedure of retro unix v1.0 by Erdogan Tan
; 12/8/2012
; updates super block and the last i-node on disk
; if modified
; e.g. smod = 1, imod = 1, buffer_m = 1
;
; RETRO UNIX v1 FS

    xor ax, ax ; mov ax, 0
    call i_get ; (write modified i-node)
    jc  short sync_3
sync_1:
; 14/07/2015
; 07/07/2015
    mov dl, byte ptr [PhysicalDriveNumber]
    cmp dl, 90h
    jb  short sync_2
    sub dx, dx ; 0
    mov cx, dx ; 0
    sub al, al ; specified offset is from the beginning of the file
    mov ah, 42h ; seek (move file pointer)
    mov bx, word ptr [img_file_handle]
    int 21h
    jc  loc_error
    mov bx, word ptr [img_file_handle]
    mov cx, 1024 ; write 1024 bytes (2 sectors)
    mov dx, offset BSBuffer ; bootsector (& super block) buffer
    mov ah, 40h ; write file
    int 21h
    jc  loc_error
    cmp ax, 1024
    jne loc_error
    ;mov bx, dx ; offset BSBuffer
    retn
sync_2:
    mov bx, offset BSBuffer
    mov ax, 0302h ; Write boot sector & super block

```

```

        mov cx,1
        xor dh,dh
        mov dl, byte ptr [PhysicalDriveNumber]
        int 13h
sync_3:
        retn

sync     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_dirbuf 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
        ; 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, offset BSBuffer
        mov ax, word ptr [SI]+bs_bf_inode_number

        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 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 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

unlink proc near
; 05/01/2013 UNIXCOPY.ASM modification (pdir -> i_get)
; 16/12/2012 UNIXCOPY.ASM version
; 02/12/2012
; unix boot file configuration version
; of "sysunlink" function of retro unix v1.0 by Erdogan Tan
; Derived from (original) UNIX v1 source code
; PRELIMINARY release of Unix Implementation Document,
; 20/6/1972
; ('sysunlink', unix kernel function)
;
; INPUT -> AX (R1) = inode number
;         [u_off] = Directory Entry Offset + 10
;         ;; [ii] = i-number of current directory
; Return -> CF = 0 -> Succeeded, CF = 1 -> failed
;
;                                     (error code in AX)

        ;jsr r0,arg; u.namep / u.namep points to name
        ;jsr r0,namei / find the i-number associated
        ;                                     with the path name
        ;br error9 / not found

push ax ;mov r1,-(sp) / put its i-number on the stack
        ;jsr r0,isdire / is it a directory
xor ax, ax
mov word ptr [u_dirbuf], ax ; clr u.dirbuf / no, clear
        ;the location that will get written
        ;/ into the i-number portion of the entry
sub word ptr [u_off], 10 ; sub $10.,u.off
        ; / move u.off back 1 directory entry
;mov ax, word ptr [ii]
mov ax, word ptr [pdir] ; 05/01/2013
call i_get
jnc short @f
pop ax
retn
;
@@:
call wdir ;jsr r0,wdir / free the directory entry
pop ax ;mov (sp)+,r1 / get i-number back
jc short @f
call i_get ; jsr r0,iget / get i-node
jc short @f
call set_imod ; jsr r0,setimod / set modified flag
dec byte ptr [inode_nlinks] ; decb i.nlinks
        ; / decrement the number of links
jnz short @f ;bgt sysret9
        ;/ if this was not the last link to file return
call anyi ;jsr r0,anyi / if it was, see if anyone has it open.
        ;Then / free contents of file and destroy it.
        ;br sysret9
@@:
        retn

unlink endp

anyi proc near
; 02/12/2012
; unix boot file configuration version
; of "anyi" procedure of retro unix v1.0 by Erdogan Tan
; Derived from (original) UNIX v1 source code
; PRELIMINARY release of Unix Implementation Document,
; 20/6/1972
; ('anyi' procedure)
;
; INPUT -> AX (R1) = inode number
; Return -> CF = 0 -> Succeeded, CF = 1 -> failed
;
;         ; mov $fsp,r2 / move start of fsp table to r2
anyi_1: ;1
        ; cmp r1,(r2) / do i-numbers match?

```

```

; beq lf / yes, lf
; neg r1 / no complement r1
; cmp r1,(r2) / do they match now?
; beq lf / yes, transfer
; / i-numbers do not match
; add $8,r2 / no, bump to next entry in fsp table
; cmp r2,$fsp+[nfiles*8] / are we at last entry in the table
; blt lb / no, check next entries i-number
; tst r1 / yes, no match
; bge .+4
; neg r1 / make i-number positive
call imap ; jsr r0,imap / get address of allocation bit
; in the i-map in r2
; DX (MQ) has a 1 in the calculated bit position
; BX (R2) has byte address of the byte with allocation bit
push bx ; retro unix modification (not as original unix code)
push dx ; retro unix modification (not as original unix code)
; AX = i-number
call itrunc ; jsr r0,itrunc / free all blocks related to i-node

pop dx ; retro unix modification (not as original unix code)
pop bx ; retro unix modification (not as original unix code)

jc short @f

; (AX=0)
; retro unix modification-> 'call itrunc' moved up for
; keeping super block unmodified if itrunc return with an error

not dx
and byte ptr [BX], dl ; bicb mq,(r2)
; / clear bit for i-node in the imap
; xor ax, ax
mov word ptr [inode_flg], ax ; 0 ; clr i.flgs
; / clear all flags in the i-node
@@:
    retn ; rts r0 / return

;anyi_2: ;1 / i-numbers match
; incb 7(r2) / increment upper byte of the 4th word
; rts r0 / in that fsp entry (deleted flag of fsp entry)

anyi endp

imap proc near
; 02/12/2012
; unix boot file configuration version
; of "imap" procedure of retro unix v1.0 by Erdogan Tan
; Derived from (original) UNIX v1 source code
; PRELIMINARY release of Unix Implementation Document,
; 20/6/1972
; ('imap' procedure)

; 11/11/2012 (maknod_imap location -> imap procedure)
mov bx, ax ; BX = R2, AX = R1 (input, i-number)
sub bx, 41 ; BX has i-41
mov cl, bl ; CX = R3
mov dx, 1 ;
and cl, 7 ; CX has (i-41) mod 8 to get the bit position
jz short @f ; 21/8/2012
shl dx, cl ; DX has 1 in the calculated bit position
@@:
    shr bx, 1
    shr bx, 1
    shr bx, 1 ; BX has (i-41) base 8 of byte number
; from the start of the (inode) map

add bx, word ptr [system] ; superblock free map size + 4
add bx, offset system + 4 ; is inode map offset in superblock

; DX (MQ) has a 1 in the calculated bit position
; BX (R2) has byte address of the byte with allocation bit

retn

imap endp

```

```

itrunc proc near
; 10/03/2013 BugFix
; 01/12/2012
; unix boot file configuration version
; of "itrunc" procedure of retro unix v1.0 by Erdogan Tan
; Derived from (original) UNIX v1 source code
; PRELIMINARY release of Unix Implementation Document,
; 20/6/1972
; ('itrunch' procedure)
;
; INPUT -> AX (R1) = inode number
; Return -> CF = 0 -> Succeeded (AX=0), CF = 1 -> failed
;                                     (error code in AX)

call    i_get ; jsr r0,iget
jc      short itrunc_7 ; 10/03/2013
mov     si, offset inode_dskp ; mov $i.dskp,r2
mov     di, si
itrunc_1:
; 10/03/2013 BugFix ('lodsrb' -> 'lodsw')
lodsw   ; mov (r2)+,r1 / move physical block number into r1
or      ax,ax
jz      short itrunc_5 ; beq 5f
push    si ; mov r2,-(sp)
test    word ptr [inode_flg], 1000h
; bit $10000,i.flgs / test large file bit?
jz      short itrunc_4 ; beq 4f / if clear, branch
push    ax ; mov r1,-(sp) / save block number of indirect block
call    dsk_rd ; jsr r0,dskrd / read in block,
; bx = Buffer offset ; 1st data word pointed to by r5
jnc     short itrunc_6 ; 10/03/2013
pop     si ; 10/03/2013
pop     si ; 10/03/2013
itrunc_7:
retn    ; 10/03/2013
itrunc_6:
mov     cx, 256 ; mov $256.,r3 / move word count into r3
mov     si, bx ; 10/03/2013 (SI more proper here than BX)
itrunc_2:
;mov     ax, word ptr [BX] ; mov (r5)+,r1
; / put 1st data word in r1; physical block number
lodsw   ; 10/03/2013 ; mov ax, word ptr [SI] ; add si, 2
;inc     bx
;inc     bx ; 10/03/2013 (BugFix)
and     ax, ax
jz      itrunc_3 ; beq 3f / branch if zero
push    cx ; mov r3,-(sp) / save r3, r5 on stack
;push    bx ; mov r5,-(sp) ; 10/03/2013, push bx is not needed
call    free ; jsr r0,free / free block in free storage map
;pop     bx ; mov (sp)+,r5 ; 10/03/2013, push bx is not needed
pop     cx ; mov (sp)+,r3
itrunc_3:
loop    itrunc_2 ; dec r3 / decrement word count
; bgt2b / branch if positive
pop     ax ; mov (sp)+,r1
; / put physical block number of indirect block
itrunc_4:
call    free ; jsr r0,free / free indirect block
pop     si ; mov (sp)+,r2
itrunc_5:
cmp     si, offset inode_dskp + 16 ; cmp r2,$i.dskp+16.
jb      short itrunc_1 ; bne 1b
; / branch until all i.dskp entries check
and     word ptr [inode_flg], 0FFFFh ; 11101111111111b
; bic $10000,i.flgs / clear large file bit
mov     cx, 8
xor     ax, ax
mov     word ptr [inode_size], ax
; clr i.size / zero file size
;mov     di, offset inode_dskp
rep     stosw ; jsr r0,copyz; i.dskp; i.dskp+16.
; / zero block pointers
call    set_imod ; jsr r0,setimod
; / set i-node modified flag
;mov     ax, word ptr [ii] ; mov ii,r1
retn    ; rts r0

itrunc endp

```

```

free proc near
    ; 01/12/2012
    ; unix boot file configuration version
    ; of "free" procedure of retro unix v1.0 by Erdogan Tan
    ; Derived from (original) UNIX v1 source code
    ; PRELIMINARY release of Unix Implementation Document,
    ; 20/6/1972
    ; ('free' procedure)
    ;
    ; INPUT -> ax (R1) = physical block number
    ; Return -> CF = 0 -> Succeeded, CF = 1 -> failed
    ;

    ;push  bx      ; mov r2,-(sp) / save r2, r3
    ;push  cx      ; mov r3,-(sp)
    ;push  dx

    ;call  free_3 ; jsr r0,3f
    ; / set up bit mask and word no. in free storage map
    ; / for block
free_3: ; 3
    mov dx, 1
    mov cx, ax    ; mov r1,r2 / block number, k, = 1
    and cx, 0Fh   ; bic $!7,r2 / clear all bits but 0,1,2; r2 = (k) mod (8)
    ; clr r3
    jz short @f
    ; bisb 2f(r2),r3 / use mask to set bit in r3
    ; corresponding to / (k) mod 8
    shl dx, cl
@@:
    mov bx, ax    ; mov r1,r2 / divide block number by 16
    shr bx, 1     ; asr r2
    shr bx, 1     ; asr r2
    shr bx, 1     ; asr r2
    shr bx, 1     ; asr r2
    ; bcc 1f / branch if bit 3 in r1 was 0 i.e.,
    ; bit for block is in / lower half of word
    ; swab r3 / swap bytes in r3; bit in
    ; upper half of word in free / storage map
free_1: ; 1
    shl bx, 1 ; asl      r2 / multiply block number by 2; r2 = k/8
    add bx, offset systm+2 ; add $systm+2,r2
    ; / address of word of free storage map for drum
    ; / with block bit in it
    ; retn ; rts r0 (return from free_3)
@@:
    or word ptr [BX], dx ; bis r3, (r2)
    ; / set free storage block bit; indicates free block
    ; 0 -> allocated, 1 -> free

    ;inc byte ptr [smod] ; incb smod / set super block modified for drum
    ;mov byte ptr [smod], 1 ; / set super block modified for drum
    ;pop dx
    ;pop cx      ; mov (sp)+,r3 / restore r2, r3
    ;pop bx      ; mov (sp)+,r2
    ; AX (R1) = Block number

    retn

free endp

chmode proc near
    ; 13/01/2013
    ; 'change mode' procedure
    ; Format: chmod <octal number> <unix file name>
    ;
    ; output -> cf=1 -> error
    ; -> cf=0 -> word ptr [arg] > 0 -> mode (string, 2 chars)
    ; word ptr [arg] = 0 -> ignored (none is done)
    xor ax, ax
    mov word ptr [arg], ax
chmode_1:
    lodsb
    cmp al, '0'
    jb short chmode_8

    cmp al, '7'
    ja short chmode_stc_retn ; cmc

```

```

chmode_2:
    or ah, ah
    jnz short chmode_3
    mov ah, al
    jmp short chmode_1

chmode_stc_retn:
    cmc
chmode_retn:
    retn

chmode_8:
    or ah, ah
    jz short chmode_stc_retn
    cmp al, 20h
    jne short chmode_stc_retn
    mov al, ah
    mov ah, '0'
    jmp short chmode_4

chmode_3:
    cmp byte ptr [SI], 20h
    ; no error if the 3rd character is a carriage return
    jne short chmode_stc_retn

    inc si
chmode_4:
    mov word ptr [arg], ax
chmode_5:
    mov word ptr [u_namep], si
    lodsb
    cmp al, 20h
    je short chmode_5
    jb short chmode_stc_retn ; no error (carriage return)
chmode_6:
    call name_i
    jc short chmode_retn

    ; ax = i-number
    call i_get
    jc short chmode_retn

    mov ax, word ptr [arg]
    xchg ah, al
    mov word ptr [arg], ax
    sub ax, '00' ; 3030h
    shl al, 1
    shl al, 1
    shl al, 1
    add al, ah

    mov bx, offset inode_flg
    test word ptr [BX], 4000h ; directory ?
    jz short chmode_7
    ; clear 'set user id' and 'executable' flags
    and al, 0Fh ; 1111b
chmode_7:
    mov byte ptr [BX], al
    mov byte ptr [imod], 1
    ;xor ah, ah
    retn

chmode    endp

chowner proc near
    ; 13/01/2013
    ; 'change owner' procedure
    ; Format: chown <decimal number> <unix file name>
    ;
    ; output -> cf=1 -> error
    ;         -> cf=0 ->
    ; BX > 0 -> offset arg == owner (decimal string, 3 chars)
    ;         BX = 0 -> ignored (none is done)

    xor ax, ax
    mov di, offset arg
    mov [di], ax ; 0
    xor bx, bx
    ; mov cx, 3

```

```

        mov cx, '90'
        ; xor dx, dx
        xor dl, dl
chowner_1:
        lodsb
        cmp al, cl ; '0'
        jb short chowner_5
        cmp al, ch ; '9'
        ja short chowner_stc_retn ; cmc
        inc dl
chowner_2:
        or bl, bl
        jnz short chowner_3

        cmp al, cl ; '0'
        je short chowner_1
        jmp short chowner_4

chowner_3:
        push ax
        mov al, 10
        mul bl
        mov bx, ax
        pop ax
chowner_4:
        sub al, cl ; '0'
        add bx, ax
        or bh, bh
        jz short chowner_7
        xor bx, bx

chowner_stc_retn:
        cmc
        retn

chowner_5:
        and dl, dl
        jz short chowner_retn
        cmp al, 20h
        je short chowner_8
chowner_6:
        mov bx, 0
        jmp short chowner_stc_retn

chowner_7:
        add al, cl ; '0'
        stosb
        jmp short chowner_1
        ;loop chowner_1
        ;cmp byte ptr [SI], 20h
        ;; no error if the 4th character is a carriage return
        ;jne short chowner_6

        ;inc si
chowner_8:
        mov word ptr [u_namep], si
        lodsb
        cmp al, 20h
        je short chowner_8
        jb short chowner_6 ; no error (carriage return)

        ;mov byte ptr [u_uid], bl
        push bx
        call name_i
        jc short chowner_9

        ; ax = i-number
        call i_get

chowner_9:
        ;pushf
        ;mov bl, byte ptr [u_uid]
        ;xor bh, bh
        ;mov byte ptr [u_uid], bh ; 0
        ;popf
        pop bx
        jc short chowner_retn

        mov byte ptr [inode_uid], bl

```

```

        mov byte ptr [imod], 1

        or  bl, bl
        jnz short chowner_retn

        mov bh, '0'
        mov byte ptr [arg], bh

chowner_retn:
        retn

chowner 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
    ; 16/02/2013

    mov bx, offset BSBBuffer
    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, offset system ; 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, offset system+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 [system]
    add bx, offset system + 2
    ; 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
    ; 17/02/2013
    ; print inode details
    ; Format: inode <decimal number>, iget <decimal number>
    ; INPUT -> AX <> 0 -> Current Inode [ii]
    ;         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'
    jb 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 [system]
    add bx, offset system+2
    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

PhysicalDriveNumber: db 0
FileHandle: dw 0

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

UNIX_Welcome:
db 0Dh, 0Ah
db 'Retro UNIX 8086 v1 FS File Import/Export Utility'
db 0Dh, 0Ah
db 'UNIXCOPY by Erdogan TAN 2012 - [14/07/2015]'
db 0Dh, 0Ah
db '(Type ', 27h, '?', 27h, ' to see valid commands)'
db 0Dh, 0Ah
db 0Dh, 0Ah, 0

usage:
db 'Usage: unixcopy [Floppy Drive or File Name] '
db 0Dh, 0Ah
db 0Dh, 0Ah
db "Floppy Drive names:"
db 0Dh, 0Ah
db 0Dh, 0Ah
db "fd0      (Floppy Disk 1, A:)", 0Dh, 0Ah
db "fd1      (Floppy Disk 2, B:)", 0Dh, 0Ah
db 0Dh, 0Ah
db "Floppy Disk Image File name examples:"
db 0Dh, 0Ah

```

```

        db 0Dh,0Ah
        db "fd0.img", 0Dh, 0Ah
        db "fd1.img", 0Dh, 0Ah
        db "runixfd.img", 0Dh, 0Ah
        db 0
unix_cdrv:
UNIX_FD_Name:
        db 'fd'
UNIX_FD_Number:
        db '0:', 0
unix_img_cdir: db '!' ; 07/07/2015
unix_cdir:     db '/'
               db 37 dup(0)

CDirOffset:    dw 0

unix_prompt_char: db '>'

CursorColumn:  dw 0

CommandBuffer:      db 74 dup(0)

program_exit: db 0

pdir:            dw 0

msg_arg:         db 0Dh, 0Ah ; 13/01/2012 (chmod)
arg:            dw 0
               dw 0 ; 13/01/2012

msg_yes_no:
        db '(Yes/No)? ', 0

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

msg_inv_file_name: ; 07/07/2015
        db 0dh, 0Ah
        db "Invalid file name !", 0Dh, 0Ah
        db "(File name must fit for 8.3 DOS format) !"
        db 0Dh, 0Ah, 0

msg_file_not_found: ; 07/07/2015
        db 0Dh, 0Ah
        db "File not found !", 0Dh, 0Ah
        db "(File must be in current directory) !"
        db 0Dh, 0Ah, 0

msg_inv_image_file: ; 07/07/2015
        db 0Dh, 0Ah
        db "Invalid floppy disk image file !", 0Dh, 0Ah
        db "(File size must be 1474560 bytes) !"
        db 0Dh, 0Ah, 0

Msg_Not_Unix_FS:
        db 0Dh, 0Ah
        db "Drive has not got a Retro UNIX v1 FS !"
        db 0Dh, 0Ah, 0

Msg_writing_file:
        db 0Dh, 0Ah
        db "Writing file..."
        db 0

Msg_Removing_file:
        db 0Dh, 0Ah
        db "Deleting file..."
        db 0

Msg_DosFile_Name:
        db 0Dh, 0Ah
        db "DOS File Name : ", 0

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

```

```

Msg_3dot_OK:      db "... "
Msg_OK:           db ' OK.', 0Dh, 0Ah, 0

msg_YES:          db ' YES'
                  db 0
msg_NO:           db ' NO'
                  db 0
error_msg:        db 0Dh, 0Ah
                  db 'Error !'

UNIX_CRLF:        db 0Dh, 0Ah, 0

msg_making_directory:
                  db 0Dh, 0Ah
                  db "Making directory..."
                  db 0

msg_removing_directory:
                  db 0Dh, 0Ah
                  db "Removing directory..."
                  db 0

msg_unix_drv_write_error:
                  db 0Dh, 0Ah
                  db 'Drive not ready or write error!'
                  db 0Dh, 0Ah
                  db 0
msg_Startup_File_Not_Exists:
                  db 0Dh, 0Ah
                  db 'Startup File is not configured yet ! '
                  db 0Dh, 0Ah, 0

msg_sf_configuration_set_ok:
                  db 0Dh, 0Ah
                  db "Startup file configuration SET is OK."
                  db 0Dh, 0Ah, 0

msg_sf_configuration_reset_ok:
                  db 0Dh, 0Ah
                  db "Startup file configuration RESET is OK."
                  db 0Dh, 0Ah, 0

msg_overwrite_question1:
                  ; 1/12/2012
                  db 0Dh, 0Ah
                  db 'Do you want to overwrite '
                  db 27h
                  db 0

msg_overwrite_question2:
                  db 27h
                  db ' file '
                  db 0

msg_remove_question1:
                  ; 1/12/2012
                  db 0Dh, 0Ah
                  db 'Do you want to delete '
                  db 27h
                  db 0

msg_remove_question2:
                  db 27h
                  db ' file '
                  db 0

align 2

RetryCount:       dw 0

; 07/07/2015
img_file_name:    db 13 dup(0)
                  db 0
img_file_handle:  dw 0
;img_file_pos:    dd 0 ; file (position) pointer
;
DirFileName:      db 20h ; 06/01/2013

```

```

BOOT_FILE_NAME: db 9 dup(0)

uf_make_datetime: dd 0 ; 25/11/2012

uf_i_number: dw 0 ; 25/11/2012

bootfile_inode:
inode:
inode_flg:      dw 801Eh ; Flags (1000000000011110b)
inode_nlinks:   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

rw: db 0

imod: db 0

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

; 30/11/2012
;imin: dd 0
;ihrs: dd 0
;iday: dw 0
;lday: dw 0
;qday: dw 0
;iyrs: dw 0
;jday: dw 0
;mday: dw 0

; 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) ; 03/02/2013, 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 (RUFS) 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 " 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 " ; 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

UNIXCOPY_Commands: ; 23/02/2013
db 0Dh, 0Ah
db "UNIXCOPY COMMANDS      [", 27h, "/", 27h, " means alternative, ", 27h, "<...>",
27h, " means command argument]", 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 "chdir/cd <directory name> : change directory", 0Dh, 0Ah
db "fromdos <dos file name> <unix file name> : copy/import dos file to unix fs", 0Dh,
0Ah
db "todos <unix file name> <dos file name>   : copy/export unix file to dos fs", 0Dh,
0Ah
db "rm <file name>         : remove/delete/unlink file", 0Dh, 0Ah
db "mkdir <directory name> : make new sub directory", 0Dh, 0Ah
db "rmdir <directory name> : remove/delete sub directory", 0Dh, 0Ah
db "link <source file name> <destination file name> : link file to file", 0Dh, 0Ah
db "exit                  : return to dos", 0Dh, 0Ah
db "show <file name>       : show file, print/display file contents", 0Dh, 0Ah
db "inode/iget <inode number> : print inode details for (decimal) inode number", 0Dh,
0Ah
db "chmod <mode> <file name> : change file mode (according to octal number)", 0Dh, 0Ah
db "chown <owner> <file name> : change file's owner (according to decimal number)", 0Dh,
0Ah
db "namei <file name>      : return/print inode number of file (as decimal)", 0Dh, 0Ah
db "fs/volume              : print (current) unix fs (super block) info", 0Dh, 0Ah
db "bootfile <file name>   : select/configure file as startup/boot file", 0Dh, 0Ah
db "nobootfile             : reset/cancel bootfile configuration", 0Dh, 0Ah
db "?/help                 : print/display UNIXCOPY commands summary (as above)", 0Dh,
0Ah, 0

align 16

; - - - - -
; buffers
; - - - - -
BSBUFFER:      db 512 dup(0)
; superblock
system:        db 512 dup(0)
ReadBUFFER:    db 512 dup(0)
WriteBUFFER:   db 512 dup(0)

SizeOfFile     equ $-100

UNIXCOPY ends

        end START_CODE

```