```
; 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 (RUFS) File System -> Boot Sector Code
; 29/10/2012

BF_BUFFER equ 700h
BF_INODE equ 600h
inode_flgs equ 600h
inode_nlks equ 602h
inode_uid equ 603h
inode_size equ 604h
inode_dskp equ 606h
inode_ctim equ 616h
inode_mtim equ 61Ah
inode_reserved equ 61Eh

boot_file_load_address equ 7E00h
boot_file_segment equ 7E0h


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

                org 7C00h

;±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±
;±
;±              PROCEDURE unixbootsector
;±
;±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±±

unixbootsector proc    near

Start:
                jmp     short @f

; RETRO UNIX 8086 FS v0.1 BootSector Identification (Data) Block
; 29-10-2012 RUFS 1.44MB FD Boot Sector

bsFSystemID:    db 'RUFS'
bsVolumeSerial: dd 0
               db 'fd'
bsDriveNumber:  db 0
bsReserved:    db 0  ; 512 bytes per sector
bsSecPerTrack: db 18
bsHeads:       db 2
bsTracks:      dw 80
bs_BF_I_number:  dw 0
               db '@'
@@:
                mov ax, cs
                mov ds, ax
               mov es, ax

                cli
                mov ss, ax
                mov sp, 0FFFEh
                sti

               mov ax, word ptr [bs_BF_I_number]

               or ax, ax
               jz short loc_no_bootable_disk

               mov byte ptr [bsDriveNumber], DL ; from INT 19h

               call load_boot_file
               jc short loc_unix_bl_error

loc_launch_bootfile:
                mov si, offset msg_CRLF
                call print_string

                mov ax, boot_file_segment ; 7E0h
                mov ds, ax
                mov es, ax
                cli
```

```
                 mov ss, ax
                 ;mov sp, 0FFFEh
                 sti

                mov dl, byte ptr [bsDriveNumber]

               ; MASM.EXE don't accept
               ; jmp 07E0h:0000h
               ; for OP Code: EA0000E007
                 db 0EAh
                 dw 0
                 dw 07E0h

NeverComeHere:   jmp short NeverComeHere

loc_no_bootable_disk:
                mov si, offset msg_press_any_key
                call print_string
                xor ax, ax
                int 16h
                int 19h

loc_unix_bl_error:
                mov si, offset unix_bfl_error_msg
                call print_string
                jmp short NeverComeHere

unixbootsector endp


print_string   proc near

                 mov    BX, 07
                 mov     AH, 0Eh
loc_print:
                 lodsb                          ; Load byte at DS:SI to AL
                 and     AL,AL
                 je      short loc_return        ; If AL = 00h then return

                 int     10h                    ; BIOS Service func ( ah ) = 0Eh
                                                ; Write char as TTY
                                                ;↑AL-char BH-page BL-color
                 jmp     short loc_print
loc_return:
                 retn

print_string   endp


read_i proc near
        ; 28/10/2012
        ; 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 [b_nread], dx ; accumulated number of bytes transmitted
        cmp word ptr [b_count], dx ; is number of byte to read greater than 0
        jna short read_i_retn

read_i_1:
        ; AX = I-Number
        push ax
        call i_get ; get i-node into i-node section of core
        mov bx, inode_size
        mov dx, word ptr [bx] ; file size in bytes in r2 (DX)
        sub dx, word ptr [b_off] ; subtract file offset
        jna short read_i_3
        cmp dx, word ptr [b_count]
                       ; are enough bytes left in file to carry out read
        jnb short read_i_2
        mov word ptr [b_count], dx

read_i_2:
        call m_get  ; returns physical block number of block in file
                   ; where offset points
        ; AX = Physical block number
        call dsk_rd ; read in block, BX points to 1st word of data in
                   ; buffer
        jc short read_i_3

readi_sioreg:
         mov si, word ptr [b_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 Buffer ; si now points to 1st byte in buffer
                   ; where data is to be placed
         mov di, word ptr [b_base] ; R1
        neg cx ; 512 - file offset(mod512) in R3 (cx)
         cmp cx, word ptr [b_count]
        jna short @f ; 2f

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

; end of readi_sioreg

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

         rep movsb

        pop ax

        cmp word ptr [b_count], 0
        ja short read_i_1

        retn

read_i_3:
        pop ax ; i-number

read_i_retn:
        retn

read_i  endp


i_get    proc near
        ; 20/10/2010 (i_i)
        ; 14/10/2012
        ; boot sector version of "iget" procedure
        ; Derived from (original) UNIX v1 source code
        ; PRELIMINARY release of Unix Implementation Document,
        ; 20/6/1972
        ; input -> AX = inode number
        ; RETRO UNIX v1 FS
        ; boot sector version
        ;; return => if cf=1 error number in [Error]

        cmp ax, word ptr [i_i] ; AX (R1) = i-number of current file
        je short i_get_3

        mov di, ax ; i-number

        add ax, 47 ; add 47 to inode number
        push ax ;
        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 i_get_3 ; Error code in AH

        mov word ptr [i_i], di

i_get_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 points to first word in i-node i.

        mov di, BF_INODE
                ; inode is address of first word of current inode
        mov cx, 16 ;

        mov si, bx ; offset Buffer

        add si, dx

i_get_2:
        ; copy new i-node into inode area of (core) memory
        rep movsw

i_get_3:
        retn

i_get   endp


dsk_rd  proc near
        ; 28/10/2012 (bf_buff_s)
        ; 20/10/2012
        ; 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

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

         mov si, ax

         mov bx, BF_BUFFER ; offset Buffer

         xor ch, ch
         mov cl, 4 ; Retry count
dsk_rd_1:
          push  cx
          mov   dx, 18                ; Sectors per track, 18
          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 [bsDriveNumber] ; Physical drive number
          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
dsk_rd_2:
        ;mov word ptr [bf_buff_s], si
dsk_rd_3:
         retn

dsk_rd  endp


m_get   proc near
         ; 28/10/2012
        ; 20/10/2012
        ; Boot sector version of "mget" procedure
        ; Derived from (original) UNIX v1 source code
        ; PRELIMINARY release of Unix Implementation Document,
        ; 20/6/1972
        ;
m_get_0:
         mov bl, byte ptr [b_off]+1
         xor bh, bh
         mov si, inode_flgs
         test word ptr [si], 4096 ; 1000h
                                  ; is this a large or small file
        jnz short m_get_1 ; large file

         test bl, 0F0h ; !0Fh  ; error if BX (R2) >= 16
        jnz short m_get_5

         and bl, 0Eh  ; clear all bits but bits 1,2,3
        mov ax, word ptr inode_dskp[bx] ; AX = R1, physical block number

        jmp short m_get_3

m_get_1: ; large file
        mov ax, bx
        mov cx, 256
        xor dx, dx
        div cx
        and bx, 1FEh  ; zero all bit but 1,2,3,4,5,6,7,8
                     ;gives offset in indirect block
        push bx      ;
        mov bx, ax  ; calculate offset in i-node for pointer
                   ; to proper indirect block
        and bx, 0Eh
         mov ax, word ptr inode_dskp[bx] ;
        or ax, ax
        jz short m_get_4

m_get_2:
        call dsk_rd ; read indirect block
        pop bx
        jc short m_get_5
        add bx, BF_BUFFER     ; R5, first word of indirect block
        mov ax, word ptr [bx] ; put physical block no of block
                              ; in file sought in R1 (AX)
m_get_3: ; 2
        ; ax = R1, block number of new block
        cmp ax, 1
        retn
m_get_4:
        pop bx
m_get_5:
        stc
        retn

m_get endp


load_boot_file proc near
        ; 28/10/2012
        ; 20/10/2012
        ;
        ; RETRO UNIX v1 FS
        ; Boot sector version
        ;
        ; loads boot file
        ;
        ; ax = i-number

load_bf_1:
        call i_get
        jc short load_bf_retn

        mov bx, inode_flgs

        test word ptr [bx], 10h ; executable file attribute bit
        jz short load_bf_stc

        mov bx, inode_size

        cmp word ptr [bx], 0
        jna short load_bf_stc

        mov word ptr [b_base], boot_file_load_address

        xor ax, ax
        mov word ptr [b_off], ax ; u_off is file offset

        ;mov bx, inode_size
        mov ax, word ptr [bx]
        mov word ptr [b_count], ax

        mov ax, word ptr [i_i]
        call read_i
        jc short load_bf_retn

        mov cx, word ptr [b_nread]
        mov bx, inode_size
        cmp cx, word ptr [bx]

        retn

load_bf_stc:
        stc

load_bf_retn:
        retn

load_boot_file endp

unix_bfl_error_msg:
                db 07h, "UNIX boot error!"
msg_CRLF:
               db 0Dh, 0Ah, 0

msg_press_any_key:
                db 07h
                db "Not a bootable floppy disk!"
                db 0Dh,0Ah
b_base: dw 0
b_off: dw 0
b_count: dw 0
b_nread: dw 0

;bf_buff_s: dw 0

i_i:           db 2 dup (0)

                org 7DFEh

bsBootSign:     dw 0AA55h

UNIX_BS ends

        end start
```