; 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