; TERMINAL.ASM

; Retro UNIX 8086 v1 Terminal Program

; (Standalone program for RUNIX boot utility)

; by Erdogan TAN

; 08/07/2014, 23/07/2014, 27/07/2014

; (06/07/2014, 05/07/2014, 04/07/2014, 02/07/2014)

.8086

CODE\_SEG segment para public

assume CS:CODE\_SEG, DS:CODE\_SEG, SS:CODE\_SEG, ES:CODE\_SEG

start:

mov ax, 0600h ; Scroll up, clear (AL=0)

mov bh, 07h ; Black backround (0),

; Light gray foreground (7)

sub cx, cx ; Left-Upper column, row

mov dx, 184Fh ; Righ-Lower column, row

int 10h

;

mov ah, 2 ; Set cursor position

xor dx, dx ; Row 0 (DH), Column 0 (DL)

xor bh, bh ; 0

int 10h

;

mov si, offset StartMsg

mov bl, 7

call proc\_printmsg

@@:

xor ah, ah

int 16h

;

cmp al, '1'

je short @f

cmp al, '2'

je short \_x

;

mov al, 07h ; BEEP !

mov ah, 0Eh

int 10h

jmp short @b

\_x:

mov si, offset \_3F8h + 1

dec byte ptr [SI] ; 2F8h

add si, 2

dec byte ptr [SI] ; 2F9h

add si, 2

dec byte ptr [SI] ; 2FAh

add si, 2

dec byte ptr [SI] ; 2FCh

mov si, offset \_EFh

mov byte ptr [SI], 0F7h

@@:

sub al, '1'

mov byte ptr [port], al

;

mov si, offset ComSMsg

add byte ptr [SI]+4, al

;mov bx, 7

call proc\_printmsg

;

xor ah, ah

mov al, 0E3h ; Communication parameters

; 9600 baud, parity none, one stop bit

;xor dh, dh

mov dl, byte ptr [port]

int 14h

;

mov cx, 65535

@@:

nop

nop

nop

loop @b

;

mov si, offset AnyKeyMsg

call proc\_printmsg

;

xor ah, ah

int 16h

;

mov ax, 0600h ; Scroll up, clear (AL=0)

mov bh, 17h ; Blue backround (1),

; Light gray foreground (7)

sub cx, cx ; Left-Upper column, row

mov dx, 184Fh ; Righ-Lower column, row

int 10h

;

mov ah, 2 ; Set cursor position

sub dx, dx ; Row 0 (DH), Column 0 (DL)

mov bx, 7

int 10h

;

mov dl, byte ptr [port]

;hook serial port interrupt

xor ax, ax

mov es, ax ; IVT segment (0)

mov bl, 0Bh\*4 ; (COM2) ; IVT offset (2Ch)

and dl, dl

jnz short @f

add bl, 4 ; 0Ch\*4

; (COM1) ; IVT offset (30h)

@@:

;mov ax, cs

mov word ptr ES:[BX], offset serial

mov word ptr ES:[BX]+2, cs

;mov es, ax

;

mov dx, word ptr [\_3FCh]

;modem control register

in al, dx ;read register

or al, 8 ;enable bit 3 (OUT2)

out dx, al ;write back to register

;

mov dx, word ptr [\_3F9h]

;interrupt enable register

in al, dx ;read register

or al, 1 ;receiver data interrupt enable

out dx, al ;write back to register

in al, 21h ;read interrupt mask register

and al, byte ptr [\_EFh]

;enable IRQ 4 (0EFh) (COM1)

; or IRQ 3 (0F7h) (COM2)

out 21h, al ;write back to register

;

sub al, al ; null

jmp short \_1 ; (initialization, wakeup signal)

sendchr:

mov ah, 1

int 16h

jnz short \_0

;hlt

nop

nop

nop

nop

jmp short sendchr

\_0:

xor ah, ah ; 0

int 16h ; Read character

\_1:

push ax

\_2:

xor dh, dh

mov dl, byte ptr [port]

mov ah, 3

int 14h

and ah, 32 ;trasmitter holding register empty

jnz short @f ;yes, ready to send

hlt ;no, check status again

nop

nop

jmp short \_2

@@:

pop ax

mov dx, word ptr [\_3F8h] ;data port

out dx, al ;send on serial port

jmp short sendchr

serial: ;

; INT 0Ch (0Bh) serial port interrupt handler

;

push ds

push ax

push bx

push dx

;

mov ax, cs

mov ds, ax

;

mov dx, word ptr [\_3FAh]

;interrupt identification register

in al, dx ;read register

and al, 0Fh ;leave lowernibble only

xor ah, ah ; 0

cmp al, 4 ;is receiver data available

jne short @f ;no, leave interrupt handler

;

mov dx, word ptr [\_3F8h] ;data register

in al, dx ;read character

;

mov ah, al

@@:

mov al, 20h

out 20h, al ;end of interrupt

;

mov al, ah

mov bx, 7

mov ah, 0Eh

int 10h ; Write character on TTY display

@@:

pop dx

pop bx

pop ax

pop ds

iret

proc\_printmsg:

mov ah, 0Eh

;mov bx, 7

@@:

lodsb

and al, al

jz short @f

int 10h

jmp short @b

@@:

retn

StartMsg:

db 0Dh,0Ah

db 'Terminal program for Retro UNIX 8086 v1... (27/7/2014)'

db 0Dh,0Ah

db 'Press 1 for COM 1 or press 2 for COM2 serial port...'

db 0Dh,0Ah,0h

ComSMsg:

db 07h

db 'COM1 selected...'

db 0Dh, 0Ah, 0

AnyKeyMsg:

db 'Press a key to continue.'

db 0Dh,0Ah,0h

\_3F8h: dw 3F8h

\_3F9h: dw 3F9h

\_3FAh: dw 3FAh

\_3FCh: dw 3FCh

;

\_EFh: db 0EFh

port: db 0

CODE\_SEG ends

end start