NET1004 - Operating System Assignment (x86)

Danny Vuong 040871275 | vuon0023@algonquinlive.com

;---------------------------------------------------

;

; \* AUTHOR (STUDENT NAME):Danny Vuong

;

; \* STUDENT NUMBER:040871275

;

; \* E-MAIL ADDRESS:vuon0023@algonquinlive.com

;

; \* LAB SECTION:A2

;

; \* ASSIGNMENT NUMBER AND NAME:NET1004 - Operating System Assignment

;

; \* PROFESSOR'S NAME:Rob Brandon

;

; \* PURPOSE: A 16-bit operating system with basic

; features and functionality

;

;---------------------------------------------------

;

; Details

; ----------------------------------

; This file is part of a simple 16 bit operating system.

; It resides within the first boot sector.

;

; Copyright 2019 Algonquin College. No part of this file may be

; reproduced, in any form or by any other means, without

; permission in writing from the College.

;

;---------------------------------------------------

.386 ; Compile for a 80386 CPU

option segment:use16 ; Force 16 bit segments instead of default 32 bit

.model tiny ; Tiny memory model

COMMENT \* ╔════════════════════════════╗

CONSTANTS

\* ╚════════════════════════════╝

buffSpace EQU 2000

rowCount EQU 0

colCount EQU 0

.code ; Start of code segment

org 07c00h ; Bootloader entry point

main:

jmp short start

nop ; "No Operation" for one cycle

COMMENT \* ╔════════════════════════════╗ ;Miscellanious strings are at the bottom

/help DEFINITIONS

\* ╚════════════════════════════╝

help db "/help", 0

clear db "clear", 0

reset db "reset", 0

theme db "theme", 0

crash db "crash", 0

;---------------------------------------------------

start:

;

; Summary: Start of the main operating system code.

;

;---------------------------------------------------

cli ; Clear interrupt flag

xor ax, ax ; Set AX to zero

mov ds, ax ; Set data segment to where we are loaded

add ax, 20h ; Skip over the size of the bootloader divided by 16 (512 / 16)

mov ss, ax ; Set segment register to current location (start of the stack)

mov sp, 4096 ; Set ss:sp to the top of the 4k stack

sti ; Set interrupt flag

; Start

COMMENT \* ╔════════════════════════════╗

MAIN CODE (With sound!)

\* ╚════════════════════════════╝

CALL beep ; POST/reset Beep

mov si, OFFSET DanOS ; Move DanOS with offset into source index

mov di,buffSpace ; Destination index address of 2000h

CALL printDanOS ; OS banner and version

mov si, OFFSET helpid ; Move helpid with offset into source index

CALL helpHeader ; OS Help header

empty:

jmp CRLF ; Carriage Return, Line Feed. This also displays a prompt at the beginning of the line (column 0)

; Console user input, loops indefinitely

COMMENT \* ╔════════════════════════════╗

SUBROUTINES

\* ╚════════════════════════════╝

; ================================= Subroutine =================================

; Operating System Banner is displayed at the top of the screen

; ===============================================================================

printDanOS: ; Print OS name and version banner

lodsb ; Goes and gets the first byte from DanOS and places it in "al" and increments to the next byte

or al,al ; Logical operator "or" to compare between al and al to see if they are equal to 0, similar to BNE.l d0,d1 in BDB

jz line ; Branch if zero, to line

cmp dl,2 ; Compare if dl=2h

je helpHeader ; Branch if equal, to helpHeader

mov ah,0Eh ; Move Eh into ahigh, teletype output

mov cx,1 ; Repetition count, print it 1 time(s)

int 10h ; Initialize the interrupt

jmp printDanOS ; Jump back to the printDanOS subroutine

; ================================= Subroutine =================================

; Carriage Return + Line Feed with Prompt '>'

; ===============================================================================

CRLF:

CALL GETSET

CALL printPrompt ; Call subroutine printPrompt

mov ah,2h ; Move 2h into ahigh, set cursor position

mov bh,0 ; Move 0 into bhigh

int 10h ; Init interrupt 10h

mov di,buffSpace ; Reinit memory

jmp writeChar ; Return to caller/subroutine

; ================================= Subroutine =================================

; CRLF without prompt, just an empty line

; ===============================================================================

line:

CALL GETSET

RET ; Return to caller/subroutine

; ================================= Subroutine =================================

; Prompt printing '>' ASCII character

; ===============================================================================

printPrompt:

mov ah,0Ah ; Move 0A into ahigh, write character

mov al,'>' ; Move ASCII '>' into alow

mov bh,0 ; Move 00 into bhigh

mov cx,1 ; Move 01 into cx. 01 into clow, pad chigh with zeros

int 10h ; Init interrupt 10h

add dl,1 ; Add 01 to dlow, this increments one of the arguments for setting the cursor: Increments the column number by 1

RET ; Return to caller/subroutine

; ================================= Subroutine =================================

; User input buffer and display, this loops indefinitely

; ===============================================================================

writeChar:

mov ah,10h ; Move 10h into ahigh, get keystroke

int 16h ; Init interrupt 16h

CALL teleType ; Teletype output

stosb ; Store alow at address ES, post increment by 1 byte

cmp al,0Dh ; Compare if alow==0D (carriage return)

je stringCMP ; Branch if equal, to stringCMP

cmp al,08h ; Compare if alow==08 (backspace)

je bSpace ; Branch if equal, to bSpace

jmp writeChar ; Jumps back to writeChar

; ================================= Subroutine =================================

; Returning cursor position parameters (in dhigh and dlow)

; ===============================================================================

getCursorPos: ; Returns the values of the row and column number to dhigh and dlow, respectively

mov ah,3 ; Move 3h in ahigh, get cursor position

mov bh,0 ; Move 0 into bh, video page 0

int 10h ; Init interrupt 10h

RET ; Return to caller/subroutine

; ================================= Subroutine =================================

; Help text subroutine. This displays the help header at the top.

; ===============================================================================

helpHeader:

lodsb ; Goes and gets the first byte from DanOS and places it in "al" and increments to the next byte

or al,al ; Logical operator "or" to compare between al and al to see if they are equal to 0, similar to BNE.l d0,d1 in BDB

jz line ; Branch if zero, to line

cmp dl,2 ; Compare if dl=2h

je writeChar ; Branch if equal, to writeChar

CALL teleType

jmp helpHeader ; Jump back to the helpHeader

; ================================= Subroutine =================================

; Backspace subroutine. Backspace, it's self explanatory.

; ===============================================================================

bSpace:

CALL getCursorPos ; Return cursor position

cmp dl,0 ; Does dlow=0?

je doorStopper ; Branch if equal, to doorStopper

mov ah,0Ah ; Move 0A into ahigh, write character

mov al,' ' ; Move ASCII '>' into alow

mov bh,0 ; Move 00 into bhigh

mov cx,1 ; Move 01 into cx. 01 into clow, pad chigh with zeros

int 10h ; Init interrupt 10h

jmp writeChar ; Return to caller/subroutine

; ================================= Subroutine =================================

; A ghetto method of "preventing" the prompt from deleting

; ===============================================================================

doorStopper: ; Not the same subroutine as setCursorPosition

CALL printPrompt ; Print '>'

mov ah,02h ; Move 2 into ahigh, set cursor position

mov dl,1h ; Move 1 into dlow

mov bh,0 ; Video page 0

int 10h ; Init interrupt 10h

jmp writeChar ; Jump to writeChar subroutine

; ================================= Subroutine =================================

; Compare string definitions to the user input (in memory location 2000).

; ===============================================================================

stringCMP:

CALL line ; Call to line subroutine

CALL CMPLoad ; Reinitialize buffer space at 2000

mov si,OFFSET help ; Load string offset help into source index

repe cmpsb ; Compare di register and si register

je helpLoad ; Branch if equal, to helpLoad

CALL CMPLoad ; Reinitialize buffer space at 2000

mov si,OFFSET clear ; Load sting offset clear into source index

repe cmpsb ; Compare di register and si register

je clearScreen ; Branch if equal, to clearScreen

CALL CMPLoad ; Reinitialize buffer space at 2000

mov si,OFFSET reset ; Load string offset reste into source index

repe cmpsb ; compare di register and si register

je resetScreen ; Branch if equal, to resetScreen

CALL CMPLoad ; Reinitialize buffer space at 2000

mov si,OFFSET theme ; Load string offset theme into source index

repe cmpsb ; compare di register and si register

je screenTheme ; Branch if equal, to screenTheme

CALL CMPLoad ; Reinitialize buffer space at 2000

mov si,OFFSET crash ; Load string offset crash into source index

repe cmpsb ; Compare di register and si register

je justWhy ; Branch if equal, to justWhy

mov si,OFFSET cmdErr ; If none of the conditions above satisfy, load in a string to si to output as error message

invalidCMD:

lodsb ; Load byte from si, post increment byte

or al,al ; Does this equal to zero?

jz CRLF ; Branch if zero, to CRLF

CALL teleType

jmp invalidCMD ; Jump back to invalidCMD

; ================================= Subroutine =================================

; Displays help information when user inputs "/help" into terminal.

; ===============================================================================

helpLoad:

CALL GETSET ; Gets and sets cursor position

mov si,OFFSET clear ; Load clear string into si register to prepare for display

helpClear:

lodsb ; Load a byte from si, post increment byte

or al,al ; Does this equal to zero?

jz helpLine ; Branch if zero, to helpLine

CALL teleType ; Teletype output

jmp helpClear ; Jump to helpClear subroutine

helpLine:

CALL GETSET ; Gets and sets cursor position

mov si,OFFSET reset ; Load reset string into si register

helpReset:

lodsb ; Load a byte from si, post increment byte

or al,al ; Doe sthis equal to zero?

jz helpLine2 ; Branch if zero, to helpLine2

CALL teleType ; Teletype output

jmp helpReset ; Jump to helpReset subroutine

helpLine2:

CALL GETSET ; Gets and sets the cursor position

mov si,OFFSET theme ; Load reset string into si register

helpTheme:

lodsb ; Load a byte from si, post increment byte

or al,al ; Does this equal to zero?

jz helpLine3 ; Branch if zero, to helpEnd

CALL teleType ; Teletype output

jmp helpTheme ; Jump to helpTheme subroutine

helpLine3:

CALL GETSET ; Gets and sets the cursor position

mov si,OFFSET crash ; Load crash string into si register

helpCrash:

lodsb ; Load a byte from si, post increment byte

or al,al ; Doe sthis equal to zero?

jz helpEnd ; Branch if zero, to helpLine2

CALL teleType ; Teletype output

jmp helpCrash ; Jump to helpReset subroutine

helpEnd:

CALL line ; An empty line

jmp CRLF ; Empty line with a prompt

; ================================= Subroutine =================================

; Clear screen. This clears the screen starting with a prompt at the top.

; ===============================================================================

clearScreen:

CALL blank ; Blank screen

jmp empty ; Print prompt and increment column number of cursor

; ================================= Subroutine =================================

; Reset screen. This resets the screen to initial display.

; ===============================================================================

resetScreen:

CALL blank ; Blank screen

jmp start ; Jump to the very top: OS Banner, copyright boilerplate, help header, and prompt

; ================================= Subroutine =================================

; Black screen. This subroutine by itself does not generate any user input.

; ===============================================================================

blank:

CALL aspect ; Gets the resolution of the program

mov bh,7d ; Foreground color, light gray. Background remains black.

int 10h ; Init interrupt 10h

mov ah,2h ; Set cursor position

mov bh,0h ; Video page 0

mov dx,0000h ; Cursor at location row 0 column 0

int 10h ; Init interrupt 10h

RET ; Return to caller/subroutine

; ================================= Subroutine =================================

; Theme screen. A theme displays on the terminal. Blue bg and red(ish?) fg

; ===============================================================================

screenTheme:

CALL aspect

mov bh,16h ; Pretty colors!

int 10h ; Init interrupt 10h

mov ah,2h ; Set cursor position

mov bh,0h ; Video page 0

mov dx,0001h ; Cursor at location row 0 column 0

int 10h ; Init interrupt 10h

jmp CRLF ; Jump to writeChar subroutine

; ================================= Subroutine =================================

; Beep!

; ===============================================================================

beep:

mov ah,0Eh ; Move 0Eh into ahigh, teletype output

mov al,07h ; Move 07h into alow, "beep" ASCII code

int 10h ; Init interrupt 10h

RET ; Return to caller/subroutine

; ================================= Subroutine =================================

; Teletype (text) output.

; ===============================================================================

teleType:

mov ah,0Eh ; Teletype output

mov cx,1 ; One character

int 10h ; Init interrupt

RET

; ================================= Subroutine =================================

; Screen dimensions: 24x79

; ===============================================================================

aspect:

mov ax,0600h ; Scroll page up, ahigh = 6h, alow = 00h

mov cx,0000h ; Screen dimensions 0x0h = 0x0d

mov dx,184fh ; Screen dimensions 18x4fh = 24x79d

RET

; ================================= Subroutine =================================

; Sets cursor position. (Increment dh "row" and set dl "column" to 0)

; ===============================================================================

setCursorPos:

mov ah,2 ; Set cursor position

add dh,1 ; Increments row number by 1

mov dl,0 ; Column Number (column 0)

mov bh,0 ; Set video page to 0

int 10h ; Init interrupt 10h

RET

; ================================= Subroutine =================================

; Get cursor position and set cursor position.

; ===============================================================================

GETSET:

CALL getCursorPos ; Return cursor position parameterrs

CALL setCursorPos ; Sets cursor position

RET

; ================================= Subroutine =================================

; Character length and buffer space load.

; ===============================================================================

CMPLoad:

mov cx,5 ; Move 5 into cx, check for 5 characters. Conveniently, all strings are 5 chara\

; cters long so they do not need to be specified again.

mov di,buffSpace ; Reinitialize buffer space at 2000

RET

; ================================= Subroutine =================================

; THIS FEATURE HALTS THE CPU FROM RECEIVING ANY INTERRUPTS.

; ===============================================================================

justWhy:

CALL beep ; Crash beep, extends the BIOS beep to 3x length

CALL beep

CALL beep

cli ; Clear interrupt flag

hlt ; Halt the cpu

; End

COMMENT \* ╔════════════════════════════╗

STRING DEFS

\* ╚════════════════════════════╝

; String definitions

DanOS db "DanOS v1.0", 0 ; Define OS banner

cmdErr db "command err", 0 ; Define cmdErr

helpid db "Type /help", 0 ; Define helpid

byte 510 - ($ - main) dup (0) ; Pad remainder of boot sector with zeros

dw 0aa55h ; Boot signature

END main