DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF MORATUWA

CS 2042
OPERATING SYSTEMS

PROGRAMMING ASSIGNMENT 1

AMARASINGHE G N

090018T

1. INTRODUCTION

Two choices were given to us for the programming assignment 1. I chose to implement a command to the given JOSH operating system to display the hardware information of the machine.

First I followed the methods given in the tutorial and then booted the JOSH successfully from a USB drive. But later on I realized that this could get hard once I started adding the modifications, because I had to restart the computer each time. Therefore I decided to follow the method of booting the JOSH in a virtual machine. So I created a floppy drive and mounted the floppy.img file in the drive.

Create the floppy drive,

```
sudo mkdir /media/floppy0
```

mount the image in the drive,

```
sudo mount -o loop -t vfat <location>/josh.img /media/floppy0
```

Later on whenever I modified the kernel I compiled and copied the kernel.bin file to the floppy drive and use that drive to boot the virtual machine.

Copy the kernel.bin to the floppy drive,

```
sudo cp <location>/kernel.bin /media/floppy0
```

For some odd reason this didn't workout exactly as I hoped it would. So I had to go through some tricky process. First by following the given tutorial, I made a bootable flash drive. Then I made an image of the pen drive using the “dd” command.

```
sudo dd if=/dev/sdb of=<location>/JOSH.img
```

Then I used that created image to be mounted on the floppy drive. It worked flawlessly and I was able to use this process to complete the assignment without having to restart my computer.

2. APPROACH

There are two main methods to get the hardware information from the BIOS Data Area.

a. Use BIOS Service interrupts

By using various BIOS interrupts we can get the hardware information that are stored in the BIOS data area. Interrupt call will return the AX register with the corresponding information.

Ex:

```
      xor ax, ax                ;clear the ax register

  int 0x11                      ;send the interrupt, it sets the ax register

      and ax, 0x0e00           ;keep the bits 9-11

      shr ax, 9                   ;shift 9 bits right

      add al, 0x30            ;add 48 and convert to decimal

      mov ah, 0x0e

  int 0x10
```

b. Use BIOS Data area and access it using the offset

When a computer is powered on, the BIOS data area is created at memory location 0040:0000h with a typical size of 255 bytes. This data area can be accessed by getting the relevant offset where the required data is saved.

Ex:

```
        push es           ;save the current values of the es register

        mov ax, 0x40             ;get the address no 0x40 to the ax register

        mov es, ax               ;move ax to the es

        mov al, [es:75h]         ;get the required offset to display no of HDD

        add al, 0x30             ;convert the number to ASCII by adding 48(decimal) = 0x30

        mov ah, 0x0e             ;print the number

  int 0x10

        pop es                   ;restore the value stored in the es register
```

By using the table [4] below we can find almost all the information of the hardware underneath the computer. We can use either,

calling BIOS service interrupts or

referring to BIOS Data area

to get the information needed.

Offset Hex	**BIOS Service**	**Field Size** **[byte(s)]**	**Information**
10h	Int 11h	2	Equipment word
			Bits 15-14 indicate the number of installed parallel ports 00b = 1 parallel port 01b = 2 parallel ports 03b = 3 prallel ports
			Bits 13-12 reserved
			Bits 11-9 indicate the number of installed serial ports 000b = none 001b = 1 serial port 002b = 2 serial ports 003b = 3 serial ports 004b = 4 serial ports
			Bit 8 reserved
			Bits 7-6 indicate the number of installed floppy drives 0b = 1 floppy drive 1b = 2 floppy drives
			Bits 5-4 indicate the video mode 00b = EGA or later 01b = color 40x25 10b = color 80x25 11b = monochrome 80x25
			Bit 3 reserved
			Bit 2 indicate if a PS/2 mouse is installed 0b = not installed 1b = installed
			Bit 1 indicate if a math coprocessor is installed

			0b = not installed 1b = installed
			Bit 0 indicated if a boot floppy is installed 0b = not installed 1b = installed
13h	Int 12h	2	Memory size in Kb
17h	Int 16h	1	Keyboard shift flags 1
18h	Int 16h	1	Keyboard shift flags 2
1Eh	Int 13h	1	Keyboard buffer
41h	Int 13h	1	Floppy disk drive status
42h	Int 13h	1	Hard disk and floppy controller status register 0
43h	Int 13h	1	Floppy drive controller status register 1
44h	Int 13h	1	Floppy drive controller status register 2
49h	Int 10h	1	Active video mode setting
63h	Int 10h	1	I/O port address for the video display adapter
65h	Int 10h	1	Video display adapter internal mode register
67h		2	Adapter ROM offset address
69h		2	Adapter ROM segment address
6Bh		1	Last interrupt (not PC)
74h	Int 13h	1	Status of last hard disk operation
75h	Int 13h	1	Number of hard disk drives
8Ch	Int 13h	1	HDD controller status
F0h		16	Intra-applications communications area

c. Using special commands

In order to view the Processor details, we can use a special command which is provided by the processor manufacturers. The **cpuid** command requires no operands, but takes arguments from the EAX register. The value stored in the EAX register before calling the cpuid command, specifies what information is returned.[10]

Argument given to the EAX register	**Result**
EAX=0	Get vendor ID
EAX=1	Processor Info and Feature Bits
EAX=2	Cache and TLB Descriptor information
EAX=3	Processor Serial Number
EAX=80000000h	Get Highest Extended Function Supported
EAX=80000001h	Extended Processor Info and Feature Bits
EAX=80000002h,80000003h,80000004h	Processor Brand String
EAX=80000005h	L1 Cache and TLB Identifiers
EAX=80000006h	Extended L2 Cache Features
EAX=80000007h	Advanced Power Management Information
EAX=80000008h	Virtual and Physical address Sizes

Ex:

```
_show_CPU_VendorID:
        ;******************read CPU vendor ID
        mov eax,0
        cpuid
        mov [vendorID], ebx
        mov [vendorID+4],edx
        mov [vendorID+8],ecx
        call _display_endl
        mov si, strVendorID
        mov al, 0x01
   int 0x21
        call _display_space
        mov si, vendorID                ;print CPU vender ID
        mov al, 0x01
  int 0x21
        ret
;end
```

3. METHODS IMPLEMENTED

The table below gives a description of methods that were implemented using the above mentioned 3 methodologies.

Method	Action
_display_hardware_info:	Display hardware information invoked by “hinfo” command
_show_CPU_VendorID:	Show CPU vendor ID
_show_Processor_Brand:	Show processor brand
_show_floppy_info:	Show available number of floppy drives
_no_floppyD:	If no floppy drives found print 0
_available_floppyD:	Print the number of floppy drives
_show_serial_info:	Show number of installed serial ports
_show_parallel_info:	Show number of installed parallel ports
_show_memory_info:	Show RAM size
_remove_cx_conflict:	Some bioses return CX=BX=0 if so copy ax to cx and bx to dx
_memCalculate:	Calculate and print the memory size
_MemErr:	If an error occurred print the error
_show_HDD_info:	Show number of installed Hard Disk Drives
_show_Mouse_status:	Show if a PS/2 mouse is installed
_print_true:	Print “true”
_print_false:	Print “false”
_display_custom_msg:	Display a custom message stored in the strTest invoked by “who” command
_display_help:	Display help invoked by “help” command

4. REFERENCE

[1] Hacking JOSH – Operating System Tutorial « Asiri&apos;s Laboratory

http://asiri.rathnayake.org/articles/hacking-josh-operating-system-tutorial/

[2] Operating System design & implementation Tutorial, planning a interrupt service ISR

http://www.mohanraj.info/josh5.jsp

[3] Expanded Main Page - OSDev Wiki

http://wiki.osdev.org/Main_Page

[4] BiosCentral - BIOS Data Area

http://www.bioscentral.com/misc/bda.htm

[5] NASM - OSDev Wiki

http://wiki.osdev.org/NASM

[6] Detecting Memory (x86) - OSDev Wiki

http://wiki.osdev.org/Detecting_Memory_%28x86%29

[7] Detecting CPU Speed - OSDev Wiki

http://wiki.osdev.org/Detecting_CPU_Speed

[8] INT (x86 instruction) - Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/INT_(x86_instruction)

[9] Assembly Tips and Tricks « CyberAsylum

http://cyberasylum.wordpress.com/2010/11/19/assembly-tips-and-tricks/

[10] CPUID - Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/CPUID

[11] Embedded BIOS 4.1 - BIOS User's Manual with BIOS Interrupt Reference

http://www.google.lk/url?sa=t&source=web&cd=1&ved=0CBQQFjAA&url=ftp://ftp.embeddedarm.com/old/saved-downloads-manuals/EBIOS-UM.PDF&rct=j&q=EBIOS-UM&ei=fQD6TM7SFo238QOgxtGmDA&usg=AFQjCNHvPBMUypwtEJ_9_YWxGDgolBr7yA&cad=rja

[12] PC Assembly Language by Paul A. Carter

[November 11, 2003]

http://www.google.lk/url?sa=t&source=web&cd=4&ved=0CDAQFjAD&url=http://net.pku.edu.cn/~course/cs201/2004/pcasm-book.pdf&rct=j&q=pcasm&ei=uAD6TKfKAtSx8QP3wOX4Cw&usg=AFQjCNEjaCOfGUgSDRCfBqT9rH6QtbUISg&cad=rja

5. APPENDIX A

Hexadecimal to decimal conversion

The procedure given below will convert the value stored in the DX register to decimal and print it character by character using decimal to ASCII conversion.

```
_hexToDec:                          ;convert hex to decimal

        push ax

        push bx

        push cx

        push si

        mov ax,dx               ;copy number into AX

        mov si,10               ;SI will be our divisor

        xor cx,cx               ;clean up the CX

_non_zero:

        xor dx,dx               ;clean up the DX

        div si                  ;divide by 10

        push dx                 ;push number onto the stack

        inc cx                  ;increment CX to do it more times

        or ax,ax                ;end of the number?

        jne _non_zero                 ;if not go to _non_zero

_write_digits:

        pop dx                  ;get the digit off DX

        add dl,0x30             ;add 48 to get the ASCII value

        call _print_char        ;print

        loop _write_digits      ;keep going till cx == 0

        pop si                  ;restore SI

        pop cx                  ;restore DX

        pop bx                  ;restore CX

        pop ax                  ;restore AX

        ret

_print_char:

        push ax                 ;save the current AX register
```

```
            mov al, dl
        mov ah, 0x0E            ;BIOS teletype acts on newline
        mov bh, 0x00
        mov bl, 0x07
    int 0x10
            pop ax                  ;restore the AX register
            ret
;end
```

6. APPENDIX B

Help for the modified OS.

Command	Action
ver	view the JOSH version
exit	reboot
hinfo	view hardware information
who	who modified the JOSH to current version

7. APPENDIX C

Modified kernel.asm file is attached with this report.