**Department of Computer Science and Engineering**

CS2042 Operating Systems

# Programming Assignment – 01

| | |
|---|---|
| Name: | D.A.U.Nanayakkara |
| Index No : | 090342F |

## Table of Contents

Introduction ........ 3
X86 architecture ........ 3
Memory layout........ 3
BIOS Data area........ 3
Interrupts ........ 3
x86 Architecture........ 4
Register types [1]........ 4
General purpose registers........ 4
Segment registers........ 4
Pointer registers........ 4
Other........ 4
Interrupts and BIOS data area........ 6
Getting Hardware Details ........ 7
CPU details [2]........ 7
int 0x11 [4]........ 9
int 0x15 [5]........ 9
Number of Hard drives ........ 10
references........ 10

# Introduction

In this implementation of Josh  I've choose to work on the shell command to print out hardware details of the system. To complete this work I had to learn about the x86 architecture, basic assembly syntax,the BIOS data area, interrupts, etc. In this document I've explained the things I had to learn when implementing the shell command.

### X86 architecture

Under x86 architecture I had to learn how the registers inside a processor work, width of registers, register types, modes of execution of a processor etc.

### Memory layout

How the memory of the system is organized. Which areas in memory stores the program instructions, variables, the program stack etc.

### BIOS Data area

What is  BIOS data area? where it is located? How it is created?

### Interrupts

What are interrupts and how they work and type of interrupts.

## x86 Architecture

x86 architecture designed by Intel is the widely used processor architecture in modern day computers (in 32-bit computing). To use the assembly language effectively we need to have an acute knowledge on processor architecture we are working on.

### *Register types [1]*

In 16-bit processors there are 16 registers.

#### *General purpose registers*

- AX
- BX
- CX
- DX

These are general purpose registers to store any kind of data by the programmer. Even though they are general purpose some registers have their own special tasks, for instance only the CX register can be used as a counter for a loop structure, AX is the accumulator .

#### *Segment registers*

- CS CS stands for code segment. Machine instructions exist at some offset into a code segment. The segment address of the code segment of the currently executing instruction is contained in CS.
- DS DS stands for data segment. Variables and other data exist at some offset into a data segment. There may be many data segments, but the CPU may only use one at a time, by placing the segment address of that segment in register DS.
- SS SS stands for stack segment. the stack has a segment address, which is contained in SS.
- ES ES stands for extra segment. The extra segment is a spare segment that may be used for specifying a location in memory.
- FS & GS FS and GS are clones of ES. They are both additional segments with no specific job or specialty. Their names come from the fact that they were created after ES (E, F, G).

#### *Pointer registers*

- SP
- BP

These are special purpose registers that points to the top of the stack and some other position in the stack.

- IP It contains the offset address of the next machine instruction to be executed in the current code segment.

#### *Other*

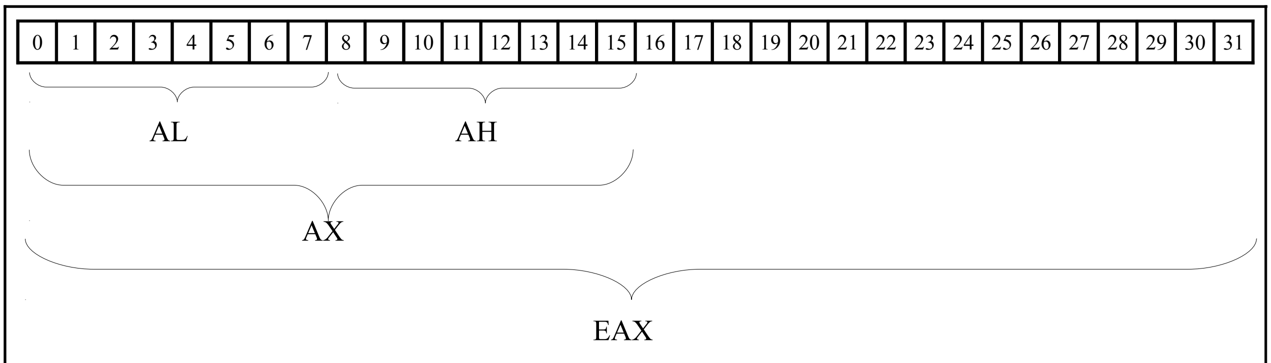
- DI
- SI

These are array indexing registers.

- FLAGS It is 16 bits in size in the 8086, 8088, and 80286, and its formal name is FLAGS. It is 32 bits in size in the 386 and later CPUs, and its formal name in the 32-bit CPUs is EFLAGS. This register is never used as a whole unit, rather it's used to keep track of errors in operations such as division by zero by setting or clearing relevant bits in the register.

** In 16-bit processors these registers were 16 bits wide. But in 32-bit processors the width of these processors were increased to 32 bits. And to accommodate backward compatibility of the processors the newer ones were simply extensions to prior registers. The naming convention of the 32 bit registers were most often can be seen as simply adding an 'E' (extended) to the beginning of the register name. For example AX registers extended version is EAX. (There are some new registers introduced in these newer processors. But knowledge about the basic registers was enough to do this assignment).

Breakdown of a 32 bit register (using EAX register)



AL : Low eight bits of the 16 bit register AH: High eight bits of the 16 bit register

## Interrupts and BIOS data area

*Interrupts*

An interrupt is an event in hardware that triggers the processor to jump from its current program counter to a specific point in the code. Interrupts are designed to be special events whose occurrence cannot be predicted precisely (or at all). There are several kinds of interrupts software interrupts, hardware interrupts etc.

When an interrupt occur the processor saves it current state (its memory pointers, data etc ) and search the interrupt vector table to identify the relevant interrupt handler for the interrupt. Then it passes the control to the interrupt handler to handle the interrupt. After interrupt is handled the processor return to its previous state.

In this assignment I've used BIOS interrupts to get hardware details.  Using BIOS Interrupts we can jump in to the relevant offset of the BIOS data area an read the hardware details from it. Rather than using interrupts we can directly access the BIOS area an read the relevant hardware details. The latter approach is efficient than the former approach. (No overhead in handling interrupts)

*BIOS data area*

BIOS data area is the location in memory where the hardware details and other information regarding the system is stored. When the computer boots up BIOS will probe for hardware devices and store the details of the hardware devices on the BIOS data area of the memory. Usually BIOS data area starts from the memory address 0040h. Using BIOS interrupts or accessing this memory area we can get hardware details of the system.

A detailed memory map of the BIOS data area is available on

http://stanislavs.org/helppc/bios_data_area.html  [Accessed on 2nd December 2010]

## Getting Hardware Details

There are two methods to get hardware details, one method is to use BIOS interrupts and the other method is to access the BIOS data area directly. In this I have used both the methods to get hardware details.

By typing **“sysinfo”** command at the prompt a user can get hardware details of the system. The system information that is shown using this command are,

- CPU details
- Number of floppy disk drives
- Amount of RAM
- Number of serial ports
- Number of parallel ports
- Number of hard drives

### *CPU details [2]*

The method used to get the details of the processor is an instruction called *cpuid.* This is not an interrupt it is a processor dependent instruction. Using this instruction with different values stored in the EAX register gives different information about the processor.

*EAX = 0*

When *cpuid* is called with the EAX register being cleared, the processor gives out the vendor ID string of the processor (which is a unique string for each processor manufacturer) to the EBX, EDX, ECX registers as a string of ASCII characters. Under CPU details of my code I have copied those values into a buffer in memory and printed it out to the screen

Some of the known porcessor ID strings[3]

| | |
|---|---|
| • "AMDisbetter!" | - early engineering samples of AMD K5 processor |
| • "AuthenticAMD" | - AMD |
| • "CentaurHauls" | - Centaur |
| • "CyrixInstead" | - Cyrix |
| • "GenuineIntel" | - Intel |
| • "TransmetaCPU" | - Transmeta |
| • "GenuineTMx86" | - Transmeta |
| • "Geode by NSC" | - National Semiconductor |
| • "NexGenDriven" | - NexGen |
| • "RiseRiseRise" | - Rise |
| • "SiS SiS SiS " | - SiS |
| • "UMC UMC UMC " | - UMC |
| • "VIA VIA VIA " | - VIA |

*EAX = 80000000h [2]*

When EAX = 80000000h it gives the brand string of the processor. This gives the processor type, the clock speed of the processor and the processor name. But some processors doesn't support this option. To determine if the brand string is supported on a processor, software must follow the steps below:

1. Execute the CPUID instruction with EAX=80000000h
2. If ((returned value in EAX) > 80000000h) then the processor supports the extended CPUID functions and EAX contains the largest extended function supported.
3. The processor brand string feature is supported if EAX >= 80000004h

If the brand string is supported it will copy the brand string into the EAX, EBX, ECX and EDX registers. After that like in the vendor ID string the brand ID is printed onto the screen.

*EAX = 80000006h [2]*

From this interrupt I've taken the L2 cache line size, which is given in bytes. ECX register is the output register. Bits 31 to 16 gives the L2 cache size in MB's, 15 to 12 gives the L2cache associativity and 7 to 0 gives the L2 cache line size in bytes.

*EAX = 80000008h [2]*

When called gives the virtual and physical address size supported by the cpu to the EAX register. Bits 15 to 8 gives the virtual address size and 7 to 0 gives the physical address size.

There are several other options in this instruction to get CPU details using different values for EAX

for example:

| | |
|---|---|
| EAX = 1h | Processor Info and Feature Bits |
| EAX = 2h | Cache and TLB Descriptor information |
| EAX = 3h | Processor Serial Number |
| EAX=80000005h | L1 Cache and TLB Identifiers |

## *int 0x11 [4]*

This is a BIOS interrupt that gives some hardware details about the system known as the "*Equipment List*" After clearing the EAX register this interrupt is called. The hardware details will be copied to the AX register.

Different bits in this represent different hardware information. The details I have used in JOSH implementation is as follows.

| Bit(s) | Description |
|---|---|
| 0 | floppy disk(s) installed |
| 2 | pointing device installed (PS) |
| 3 | unused (PS) |
| 5 and 4 | initial video mode.<br>• 00 EGA, VGA, or PGA.<br>• 01 40x25 color.<br>• 10 80x25 color.<br>• 11 80x25 monochrome |
| 7 and 6 | number of floppies installed less 1 (if bit 0 set) |
| 11 to 9 | number of serial ports installed |
| 15 to 14 | number of parallel ports installed |

* Full equipment list is available at Ralf Brown's Interrupt List [4]

## *int 0x15 [5]*

This interrupt is to get the memory map of the ram. There are three methods to get the memory map using this interrupt (by setting the AX register to different values).

*Int 0x15 , AX = e801h*

This is the interrupt that is used in JOSH. This can map up to 4GB of memory. When the interrupt is called the output is given to AX and BX registers. (the same result is give to CX and DX registers).

| | | |
|---|---|---|
| CF | Carry Flag | Non-Carry - indicates no error |
| AX | Extended 1 | Number of contiguous KB between 1 and 16 MB, maximum 0x3C00 = 15 MB. |
| BX | Extended 2 | Number of contiguous 64 KB blocks between 16 MB and 4 GB. |
| CX | Configured 1 | Number of contiguous KB between 1 and 16 MB, maximum 0x3C00 = 15 MB. |
| DX | Configured 2 | Number of contiguous 64 KB blocks between 16 MB and 4 GB. |

- *the memory size given by the AX and CX registers give 1 MB less of the actual memory.*

To map more than 4GB of memory we have to use AX= e820h

### *Number of Hard drives*

The details about number of hard drives installed in the system is stored in the BIOS area. Byte at offset 75h contains this detail. By directly accessing the BIOS area I have taken this data without using an interrupt. The number in hex is converted to decimal before printing the number.

## references

[1] Jeff Duntemann, "*Chapter 4:Locations locations locations*" *in Assembly language step by step,*3rd Ed, Wiley publishing, inc., 2009

[2] Intel, "*Output of the CPUID Instruction" and "Brand ID and Brand String " in Intel® Processor Identification and the CPUID Instruction,* Application Note 485 , August 2009

[3] Wikipedia. (2010, December, 2). *CPUID* [Online]. Available: http://en.wikipedia.org/wiki/CPUID

[4] Ralf Brown's Interrupt List (2010, December, 2). *Int 11* [Online]. Available: http://www.ctyme.com/intr/rb-0575.htm

[5] Erich Boleyn (2010, December, 2). *Int 15* [Online]. Available:

http://www.uruk.org/orig-grub/mem64mb.html

[6] David Jurgens (2010, December, 2). BIOS data area [Online]. Available: http://stanislavs.org/helppc/bios_data_area.html