Ayesha Mehmood
ID # 6832
Semester # 8th
Mid-Term
Degree # BSCS
Subject # Microprocessor and Assembly Language
Date # 13-4-2020

(Q2) Write short note on each of the following:

a) **Embedded Systems:**

Ans) An Embedded System is a combination of computer hardware and software, either fixed in capability or programmable, designed for a specific function or function within a larger system. Industrial machines, agriculture and process industry devices, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines and toys, as well as mobile devices, are possible for an locations for an embedded system.

b) **Device driver:**

Ans) A device driver or hardware driver is a group of files that enable one or more hardware devices to communicate with the computer's operating system. Without, drivers, the computer would not be able to send and recieve data correctly to hardware devices, such as a printer.

c) **Virtual Machine Concept:-**

Ans) A virtual machine is a software program or operating system that not only exhibits the behaviour of a separate computer, but is also capable of performing tasks such as running applications and programs like a separate computer. A virtual machine, usually known as a guest is created within another computing environment referred as a "host".

## d) Instruction Execution Cycle:

Ans) * The time period during which one instruction is fetched from memory and execute when Computer given an instruction in machine language.

* Each instruction is further divided into sequence of phases.

* After the execution of program Counter is incremented to point to the next instruction.

## e) Motherboard Chipset:-

On the motherboard, a PC's chipset controls the communication between the CPU, RAM, storage and other peripherals. The chipset determines how many high-speed components or USB devices your motherboard can support. Chipset are usually comprised of one to four chips and feature controllers for commonly

used peripherals like the your best Gaming keyboard, mouse or monitor.

## f) Access levels of input-output operations:

Ans) There are 4 access levels I/O interaction.

- Level 3 - High level programming Language.
- Level 2 - Operating System API
- Level 1 - BIOS
- Level 0 - Direct Hardware interaction.

* Operating System may reserve Direct access to hardware.

## g) Basic Parts Of an Assembly Language Instruction:

Ans) An assembly instruction has 4 basic parts:

- Label (optional)
- mnemonic (required)
- Operand (depends on the instruction)
- Comment (optional)

Q3) Differentiate b/w each of the following:

## a) Assembly language and high-level language:

Ans) Assembly language is the language between high-level language and machine language. The key difference b/w machine language and assembly language is that, machine

Language executes directly by a computer and assembly language requires an Assembler to convert to machine code or object code to execute by the CPU.

b) Protected mode and Real address mode:

Ans) - Real mode is an operational mode that allows newer intel 286.

- Memory addressing upto 1MB physical memory.
- No virtual memory support.
- Memory protection mechanism is not available.
- Does not support virtual address space.
- Does not support LDT and GDT.
- Segment descriptor cache is not available.

- Support Segmentation.

## Protected Mode:

- Memory addressing upto 16MB of Physical memory.
- Supports up to 64TB of virtual memory.
- Memory protection mechanism is available.
- Gives virtual and physical address space.
- Support LDT and GDT
- Segment description Cache is available.
- Supports Segmentation and paging.

## c) Assembler and Linker:

* An assembler translates assembly language programs into machine code. The output of a assembler is called an object file, which contains a combination of machine instruction as well as the data required to place these instructions in memory.

* Linker is a computer program that links and merge various object files together in order to make an executable file. All these files might have been compiled by separate assembler.

## d) Instruction and directive:

An instruction is a task to be carried out by the processor at run time. Instructions are assembled into machine code and eventually linked into final executable.

* A directive is an instruction to the assembler telling it how to treat the data it is asked to assemble.

e) Code label and data label:

- Data label
  - must be unique
  - Example: myArray
    (not followed by colon)
- Code label
  - target of jump and loop instructions.
  - Example L1:
    (followed by colon)

* Data labels exists in the data segment as variable offsets. Code labels are in the code segment, and are target of transfer of control instructions.

## 9) Equal-Sign directive and EQU directive:

* The equal-sign directives associates a symbol names with an integer expression. This syntax is:

| name = Expression |
|---|

- Expression is a 32-bit integer.
- may be redefined.
- Name is a called a Symbolic constant.

* Define a Symbol as either an integer or ~~text~~ expression.
- Cannot be redefined.

Q4) Give answer to each of the following:

c) Why was unicode invented:

Unicode is a universal computing Standard to represented texts in most writing System. It was invented to Store most of the world's characters. It is started during 1987.

e) Create a truth table to Show all possible input and output for the Boolean function describe by $\neg(A \vee B)$

| A | $\neg A$ | B | $\neg A \vee B$ |
|---|---|---|---|
| F | T | F | T |
| F | T | T | T |
| T | F | F | F |
| T | F | T | T |

a) Explain the concept of Portability as it applies to programming language.

Ans) A language whose source program can be compiled and run on a wide variety of computer system is said to be portable.

b) why would a high-level language can be an ideal tool for writing a program that directly accesses a particular brand of printer?

Ans) A high-level may not be provide for direct hardware access. Even if it does awkword coding techniques possible maintance problem.

f) why does memory access take more machine cees than register access?

Ans) Conventional memory is outside the CPU and its responds more slowly to access registers are kind, inside the CPU.

d) If W = 11101100, X = 00010011 and Y = 00111100, and then find Z = W ∨ X ∧ ¬Y.

Ans) W = 1 1 1 0 1 1 0 0

X = 0 0 0 1 0 0 1 1

Y = 0 0 1 1 1 1 0 0

Soln:-

Z = W ∨ X ∧ ¬Y

$Y = 00111100$

$\neg Y = 11000011$

$\Rightarrow X \wedge \neg Y =$

$00010011$

$11000011$

AND $00000011$

$\Rightarrow W \vee X \wedge \neg Y =$

$11101100$

OR $00000011$

$11101111$

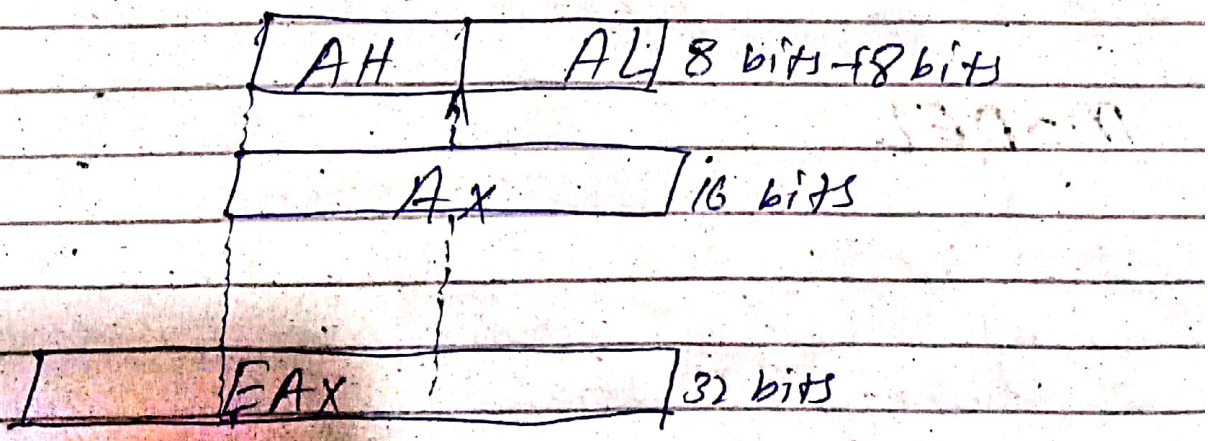
9) Discuss the basic program execution registers used in x86 32-Bit Processor.

Ans) **Basic Program Execution Register:**

Registers are high-speed storage locations directly inside the CPU, designed to be accessed at much higher speed storage than conventional.

**General-Purpose registers.**

The general purpose registers are primilarly used for arithematic and data movement.



Q5) Discuss the following MASM directives in details:

.386

.model flat, stdcall, .stack
4096 ExitProcess PROTO
dwExitCode : DWORD

The .386 directives identifies it as 32-bit program. Line 2 uses the flat memory model and windows. requires the stdcall. Conution to be used. Line 3 sets a side 4096 bytes of storage and line 4 declares a prototypes for the ExitProcess function. This prototype has a PROTO keyword, a Comma, and a list of input parameters.

.MODEL

This tells the assembler which memory model to use - In 32-bit Programs, we use the flat memory model, which is associated with the Processors Proted mode.

## • STACK

The Stack directive tells how many bytes of memory reserve for the runtime stack. 1096 happens to correspond to the size of a memory page in the processor's system for the manging memoriy.

## • CODE

It's the beginning of the code area of the program (meaning what's after-words is usually the main procedure).

## • DATA

The data directive creates a near data segment.

- This data segment contain the frequency used data for your program.
- Data segment an occupy.

# PROTO

The proto directives by using the invoke directive.

Syntax:

Label proto [(distance)] [language-type]
[[parameter]]..tag---]]

Parameter:

distance (32-bit MASM only) (optional) used in 16-bit memory model to override the default and indicate NEAD or FAR calls.

Remarks:

See PROC

# END

Directive for END of the Command that End's the here as you using main you to end with.

- Simply know that the Assembler need END directives to end the file END can be written without main Just end the file with END.

Now ENDP denotes END of PROCEDURE have procedure called "main" so before ending the file, find the "main" procedure - you have to end the procedure.

## PROC

mark start and end of a procedure - block called label.

The Statement in the block can be called with the call instruction or INVOKE directives.

**Syntax:**

label proc [distance] [language-type]
[[PUBLIC | PRIVATE | EXPORT]]

[[ < proto query > ]]

* [[ uses reglist ] [ parameter [[ : tag ]] ... ]]
* [[ FRAME [[ : exhandler - address ]] ]]

Statement
label end.

## ENDP

marks the end of procedure name previously begun with PROC.

### Syntax

name ENDP

Q6) write a program that calculates the following Expression,

$$A = (A+B)-(C+D)$$

Ans)

```
INCLUDE Irvine32.inc ; Contains
Library procedures for IA - 32
.data ; data segment, read and
        write
varA BYTE 10
varB BYTE 20
varC BYTE 30
varD BYTE 40
Final val BYTE ?

.code ; Code segment read-only
main PROC

mov al, varA
mov bl, varB
mov cl, varc
mov dl, varD
ADD al, bl ; compute (A+B)
ADD cl, dl ; compute (C+D)
SUB al, cl ; compute (A+B)-(C+D)
mov finalval, al

call Dumpregs
```

exit
main ENDP

END main

b) Show the order of individual bytes in memory for the following doubleword variable using little endian order:

dval DWORD 12345678h

Ans) The remaining placed in memory at offset 0000, 78h would be stored in the first byte, 56h would be stored in the second byte and the remainier bytes would be at offset 0002 and 0003.

Little endian representation of 12345678h

| | |
|---|---|
| 0000 | 78 |
| 0001 | 56 |
| 0002 | 34 |
| 0003 | 12 |

c) write a statement that causes the assembler to calculate the number of bytes in the following string: and assign the value to a Symbolic constant named StringSize:

String 1 byte " Assembly language is easy", 0

Ans) .data

String1 byte " Assembly language is easy", 0

String byte?

.code

mov eax, SIZEOF String1

mov StringSize, eax

d) Write a program that perform arithmetic operations on different register operands and stores the result in memory. Give stepwise explanation of each statement.

Let the Arithmetic operation is

$x = -a + (b - c)$

- data

```
x DWORD ?          ; Uninitilized variable.
a DWORD 10         ; Initialized variable 'a'
b DWORD 26         ; Initialize variable 'b'
c DWORD 15         ; Initialize variable 'c'
```

- Code

```
main PROC

mov eax, a         ; moves value of 'a' into 'eax'
neg eax            ; EAX = -10
mov ebx, b         ; mov value of 'b' into 'ebx'
sub eax, c         ; EBX = 14
add eax, ebx       ; Performs -10 + 14
mov x, eax         ; 1

call dumpregs

exit

main ENDP
END main
```

Q1. Solve of the following:

a) $64_{10} = (?)_2$

| | |
|---|---|
| 2 | 64 |
| 2 | 32 — 0 |
| 2 | 16 — 0 |
| 2 | 8 — 0 |
| 2 | 4 — 0 |
| 2 | 2 — 0 |
| | 1 — 0 |

$\Rightarrow (1000000)_2 = (64)_{10}$

b) $(011111111)_2 = (?)_{10}$

Ans)

$(011111111)_2 = (?)_{10}$

$\Rightarrow (0\times2)^7 + (1\times2)^6 + (1\times2)^5 + (1\times2)^4 * (1+2)^3 + (1+2)^2 + (1*2)^1 + (1+2)^0$

$= 0 + 64 + 32 + 16 + 8 + 4 + 2 + 1$

$= (127)_{10}$

c) $(4D7F)_{16} = (?)_{10}$

Ans) $(4D7F)_{16} = (?)_{10}$

$= 4 \times 16^3 + D \times 16^2 + 7 \times 16^1 + F \times 16^0$

$= 4 \times 4096 + 13 \times 256 + 7 \times 16 + 15 \times 1$

$= 16384 + 3328 + 112 + 15$

$= (19839)_{10}$

d) $(128)_{10} = (?)_{16}$

Ans) $(128)_{10} = (?)_{16}$

| | |
|---|---|
| 16 | 128 |
| 16 | 8 — 0 |

$= (80)_{16}$

e) $(3A6F)_{16} = (?)_2$

Ans) first convert all to binary (4 bits)

3 A 6 F
0011 1010 0110 1111

$\Rightarrow (0011\ 1010\ 0110\ 1111)_2 = (3A6F)_{16}$

f) $(11\ 00\ 00\ 11111\ 00\ 101)_2 = (?)_{16}$

Ans) $(11\ 0000\ 1111\ 00\ 101)_2 = (?)_{16}$

12 3 14 5E

(C) (3) (E) (5)

$= (C3E5)_{16}$

g) $(-16)_{10} = (?)_2$

Ans) $(-16)_{10} = (?)_2$

As the decimal integer is negative so the MSB is Binary will be 1 now Converting 16 into binary.

| | |
|---|---|
| 2 | 16 |
| 2 | 8 — 0 |
| 2 | 4 — 0 |
| 2 | 2 — 0 |
| | 1 — 0 |

⇒ 00010000

Its the +ve 16 representation taking 2's complement.

⇒ 00010000

11101111

+1

11110000

$(-16)_{10} = (11110000)_2$

2

h) $(01111111)_2 - (00000111)_2$

Ans)

1st complement of 00000111 = 11111000

now add 0000

01111111

11111000

(1)00111111

Carry

Now ~~add~~ add this carry to answers.

$$\begin{array}{r} 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1 \\ +\ 1 \\ \hline 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0)_2 \end{array}$$

I) $(6D)_{16} - (3F)_{16}$

Ans) $(6D)_{16} = 0110\ 1101$

$(3F)_{16} = 0011\ 1111$

Taking 1st Complement of 2nd number:

$$\begin{array}{r} 0\ 1\ 1\ 1\ 1\ 1\ 1 \\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \end{array} \Rightarrow 1^{st}\ \text{Complement}$$

Now add both numbers.

$$\begin{array}{r} 1\ 1\ 0\ 1\ 1\ 0\ 1 \\ +\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline (1)\ 0\ 1\ 0\ 1\ 1\ 0\ 1 \end{array}$$

Carry

Add Carry in result

$$0\ 1\ 0\ 1\ 1\ 0\ 1$$
$$+\ 1$$
$$(0\ 1\ 0\ 1\ 1\ 1\ 0)_2$$

$$= (2E)_{16}$$

J) $(1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)_2 = \pm (?)_{10}$

Ans) Signed integer as MSB is "1" which show number is negative.

$\Rightarrow$ 1.1.1.1.1.1.1.1

$= 1\times2^7 + 1\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

$= -128 + 1\times64 + 1\times32 + 1\times16 + 1\times8 + 1\times4 + 1\times2 + 1\times1$

$= -128 + 64 + 32 + 16 + 8 + 4 + 2 + 1$

$= -1$