NAME SAEEDA NAZ

ID 14332

Department BS(cs)

Semester 5th

teacher Sir Amin

Subject Microprocessor AND Assembly Language.

Question No 2

Question No 3 :-

Ans :- Virtual machine concept :-

A virtual machine (VM) is a software Program or operating System that not only exhibits the behavior of a separate computer, but is also capable of performing task such as running application and Program like a seperate computer.

Question No 2.

Ans :- Device drive :-

Device drive are programs that translate general operating

System command into specific reference to hardware detail that only the manufacture knows. Drives are hardware dependent and operating system - specific. A device drive is a file that tells the computer know the configuration and specification of a certain hardware device

Question No 1 :-

Ans Embedded System :-

A embedded system is a computer system. a combination of a computer processor, computer memory and input-/output peripheral devices. that has a dedicated function within a larger machanical or electrical system.

Question No 4 :-

Ans :- Instruction execution cycle :-

There three execution cycle :-

1) fetch
2) decode
3) execute

Question No 7 :-

Basic part of assembly language instruction.

There are four basic part of instruction.

e.

Ans :- [label :] mnemonic [operands] [; comment]

Question No 5 :-

Motherboard chipset

The chipset is the "glue" that connect the microprocessor to the rest of the motherboard and therefore to the rest of the computer. on a pc it consists of two basic part : the north bridge and the south bridge All of the various component of the computer communicate with the cpu through the chipset.

Question No 6 :-

Access level for input-output operation.

Ans Question No 6 :-

Access level for input-output operation.

Ans In computer input/output or I/O (or, informally, io or IO) is the communication b/w an information processing system such as a computer and the outside world, possibly human or another information processing system. Input are the signals or data received by the system and output are the signals or data sent from it. The term can also be used as part of an action; to "perform I/O" is to perform an input or output.

Question No 3

Part No 1 :-

Different b/w Assembly language and high level language.

Ans: An Assemenably language directly control the physical hardware. A high level language is much more abstract; which must be compiled / translated is a step up from this. High level language act the middle - man between human thinking and machine language.

Part No 2:-

Different b/w Protected mode and real mode

Real mode	Protected mode.
Real mode is an operational mode that allows newer intel 986 processor to acquire the characteristic of the reduced 8086 or 8088 processors, Real mode provide a higher clock speed but only restricts that processor to a 16-bit and a minimum of 1	The Protected mode is a 32-bit operating mode that is identified on Intel 80786 or later on. This mode allow the termination of a failed program without restarting the computer or running programs it offer access to

mb RAM instruction	addressing by virtual memory.), expanded memory, and) multiple task) while it protecting program against memory over writing.

Part No 3 :-

Assembler and linker.

Ans The main output produce by assembler an input assembly language source file is the translation of the file into an object file in (ELF). ELF files produced by the assembler are relocatable files that hold code and for data. they are input file for linker.

Part No 5 :-

Different b/w code label and data label.

code label	Data label
code label is the label that we have on code as we see in case of conditional jump (label 11) and is normally used for loop control statement.	DATA Label is the label that we use to define data as we define memory location num 1, num 2 ..... etc in our programs.

Part No 7 :-

Different b/w Equal-sign directive and EQU directive.

Ans. Equal-sign Directive	EQU - directive

The Equal-sign directive associates a symbol name with an integer expression.

This syntax is

name = Expression

Expression is a 32-bit integer.

- May be redefined.
- Name is called a Symbolic constant.

Define a Symbol as either an integer or text expression.

- connot be redefined.

- The EQU directive give a Symbolic name to a numeric constant a register-relative value or a PC-relative value.

Part No 6 :-

Different b/w Line comment and block comment :-

Line comment	block comment
A single line comment and as implied	A block comment and refers usually

only applied to a single line in the "source code" (the program).	refers to a paragraph of text. A block comment has a start symbol and an end symbol and everything between is ignored by the computer.

Part No 4 :-

Different b/w instruction and directive.

Instruction	directive.
An instruction is a task to be carried out by the processor at run time. Instruction are assembled into machine code and eventually linked into the final executable.	A directive is an instruction to the assembler telling it how to treat the data it is asked to assemble. Directives only used at assembly time and although they may affect the way the code is generated, they don't result in any code

generation themselves.

Question No 4

Part No 1 :-

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

Part No 2 :- A high-level-language may not be provide for direct hardware access. Even if it does awkward coding techniques possible maintance problem.

Part No 3 :-

Unicode is a character encoding standard that has widespread acceptance. They store letter and other characters by assigning a number for each one. Before unicode was invented, there was ~~standard~~ hundreds of different encoding system for assigning these numbers. No single encoding could contain enough characters.

Part No 5 :-

$\neg (A \vee B)$

Create table.

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

Part No 6 :-

Ans :- conventional memory is out side the CPU and it responeds more slowly to access request. Register are hard-wired inside the CPU.

Part No 4 :-

W = 11101100

X = 00010011

Y = 00111100

W = 11101100
X = 00010011
Y = 00111100

Sol :-

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

Y = 00111100
$\neg$Y = 11000011

=> X $\wedge \neg$Y = 00010011
11000011
AND 00000011 Ans

=> $W \wedge X \wedge \neg Y$:-

```
1 1 1 0 1 1 0 0
0 0 0 0 0 0 1 1      OR
1 1 1 0 1 1 1 1
```

Part No 7 :-

Ans :- XMM Registers :-

The x86 architecture also contain eight 128-bit register called XMM registers. they are used by Streaming SIMD extension to the instruction.

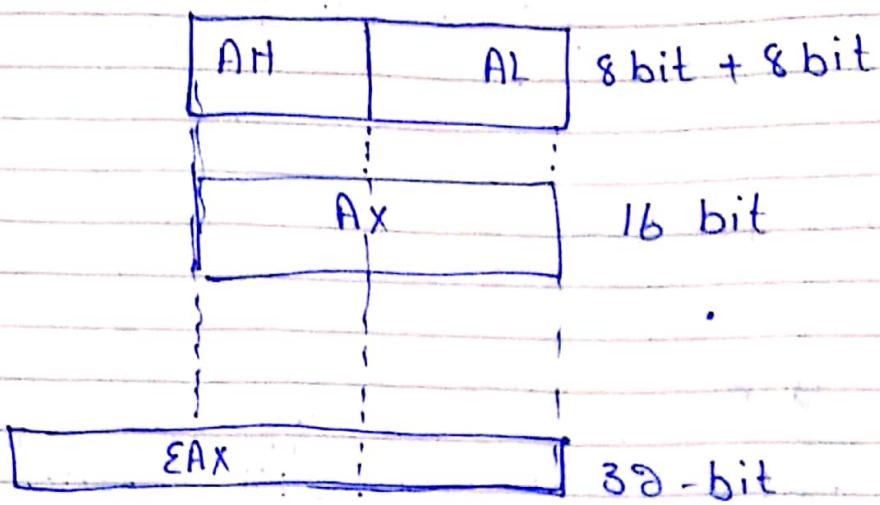
a) MMX Registers :-

MMX technology improves the performance of Intel processor when implementing advanced multimedia and communication application. The 64 bit MMX register support special instruction called SIMD (single-instruction) multiple-DATA).

3) Segment Register :-

In real-address mode, 16-bit segment register indicate base addresses of preassigned memory areas. named segments.

*) General-purpose register :-



* Basic Programming Execution Register :-

Registers are high-speed storage location directly inside the CPU designed to be accessed at much higher speed than conventional memory. when a processing loop is optimized for speed.

Question No 4

Question No 5

Part No 1 :-

- **MODEL :-** This tells the assembler which memory model to use. In 32-bit program, we use the flat memory model, which is associated with the processor's protected mode.

- **DATA :-**
  - DATA directive creates a near data segment.
  - This segment contain the frequency used to data for your program.
  - DATA segment can occupy
    - up to 64k in MS-Dos
    - or up to 512 mega byte under flat model in window NT.

• PROTO :-

The PROTO directives by using the INVOKE directive

Syntax :-

Label PROTO [[distance]]
[[language-type]][[parameter]:tag....]

Parameter :-

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

• code :-

.code
main PROC

it is the beginning of the code area of a program (meaning what's afterward is using usually the main procedure.

- .386 :-

```
model flat.
.stack 4096
ExitProcess PROTO
dw Exit code : DWORD
```

The .386 directive identifies it as a 32-bit program line 2 uses the flat memory model, and window. requires the Model converstion to be used.

Line 3 set aside 4096 bytes of storage
Line 4 declares a photo type for the Exit process function.

- Stack :-

The stack direction tell how many bytes of memory to reserve for the run time stacks.

4096 happen to correspond to the size of a memory page in this processor system for managing memory.

PROC :-

MasM Start and end of a

of a procedure block called label.

Syntax :-

label PROC [distance] [language_type]
[[PUBLIC | PRIVATE | EXPORT]]
[[< protoguearg >]]

* [[user regist]] [[parameter [[:tag]]]]
* [[FRAME [[:ehandler - address]]]]

Statement
label END.

④ ENDP :-

Marks the end of prod. procedure name previously begun with PROC

Syntax :-

name ENDP

Remarks :-

See PROC

4) END :-

directive for END of files command. That's END of file. Here as you are using "main" you have to end it with.

- Simply know that the Assembler need END directive to end the file END can be written without Main just end the file with END.

4) Now ENDP denoted END of procedure here Procedure id "main" So before ending the file, END the "main" procedure you have to end the procedure

Question No 1

Solve each of the following.

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

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

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

2) $(01111111)_2 = (?)_{10}$

Ans $(01111111)_2 = (?)_{10}$

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

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

$= (127)_{10}$ Ans.

3) $(4D7F)_{16} = (?)_{10}$

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

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

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

$\Rightarrow 16384 + 3328 + 112 + 15$

$\Rightarrow (19839)_{10}$ Ans

4) $(198)_{10} = (?)_{16}$

Ans :- $(198)_{10} = (?)_{16}$

16	128
16	8 — 0

$= (80)_{16}$.

5) $(3A6F)_{16} = (?)_2$.

Ans: First convert all to binary (4 bits)

3 A 6 F

0011 1010 0110 1111

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

6) $(11\ 0000\ 1111\ 00\ 101)_2 = (?)_{16}$.

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

12 → (C) 3 → (3) 14 → (E) 5 → (5)

$\Rightarrow (C3E5)_{16}$ Ans

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

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
2	1 — 0

$\Rightarrow$ 00010000

its the +ve 16 representation Taking 2's complement.

$\Rightarrow$ 00010000
11101111
+1

11110000

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

8) $(01111111)_2 - (00000111)_2$ (2's compliment)

Ans :-

1st complement of 00000111 = 11111000

Now add

```
  1 1 1 1
  0 1 1 1 1 1 1 1
+ 1 1 1 1 1 0 0 0
-----------------
1 0 1 1 1 0 1 1 1
```

carry

Now add this carry to answers.

```
    1 1 1
  0 1 1 1 0 1 1 1
        + 1
-----------------
  (0 1 1 1 0 0 0 0)2.
```

Ans ⇒

7) $(6D)_{16} - (3F)_{16}$.

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

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

Taking 1st complement of 2nd Number.

0 111111

1 000000 => 1st complement

Now add both numbers.

```
  1101101
+ 1000000
-----------
(1) 0101101
```

carry

add carry in result.

```
  0101101
        +1
-----------
 (0101110)2   Ans
```

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

10) $(11111111)_2 = \pm(?)_{10}$

Sol :- $(11\ 11\ 11\ 11)_2 = \pm(?)_{10}$

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

$\Rightarrow$ 11111111

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

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

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

$\Rightarrow 127 = -1$

$= \boxed{\pm127}$ Ans $= -1$ Ans

Question No 6.

(1) Write a program the calculate the following expression A = (A+B)-(C+D)

Ans :- A = (A+B)-(C+D)

```
.data ; data segment , read and write
var A   BYTE 10
var B   BYTE 20
var C   BYTE 30
var D   BYTE 40

Final. val BYTE ?

.code ; code. segment , read-only
main PROC

Mov al , var A
Mov bl , var B
Mov cl , var C
Mov dl , var D

ADD al , bl ; compute (A+B)
ADD cl , dl ; compute (A+B)
SUB al , cl ; compute (A+B)-(C+D)
Mov final val , al
```

```
call DumpRegs
exit
main ENDP
END main
```

Part No B :- when placed in memory at offset 0000, 78h would be stored in the first byte 56h would be stored in the second byte and the remain bytes would be at offset 0002 and 0003 as show in figure 3.14

Little-endian representation of 12345678h.

Part No C :-

String / data

```
.data
String byte "Assembly language is
easy ",0
StringSize byte ?

.code

mov eax SizeOf String1
mov StringSize, eax
```

Part No D :-

Let the Arithematic operation is $x = -a + (b-c)$

```
INCLUDE Irvine32.inc

.data
x DWORD ?    ; Uninitialized variable.
a DWORD 10   ; initialize variable 'a'
b DWORD 26   ; initialize variable 'b'.
c DWORD 15   ; initize variable 'c'
```

- code

```
main    PROC
mov     eax, a      ; move value of 'a' into 'eax'
neg     eax         ; EAX = -10
mov     ebx, b      ; move value of 'b' into 'ebx'
sub     ebx, c      ; EBX = 11
add     eax, ebx    ; perform -10+11
mov     x, eax      ; 1

call    DumpRegs

exit

main    ENDP

END     main.
```