133745

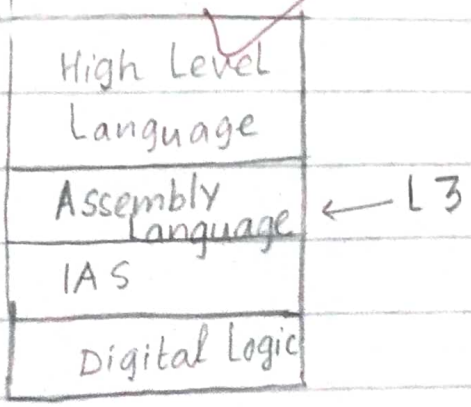
Question # 01

i) When a single instruction is translated into several machine instructions, this relation is termed as one to many. This usually happens when we translate a high level language program into machine language.
On the opposite, one a single program instruction is translated into single machine code instruction, it is termed as one to one. This occurs when we translate assembly language into machine language

2

ii) Virtual machine enables the user to run various ~~mac~~ guest machines under one single machine - the host. It provides an environment that is more user-friendly.
Assembly language is found at level 3 of the machine level.

2



(iii) Registers are built in inside the CPU; whereas memory - RAM - is located outside the CPU which is attached through buses.
Accessing register require a single clock cycle, whereas in order to access memory first the address is placed on data bus; then processor RD pin is change, followed by setting of the memory chips and eventually placing of the data in the operand. Therefore; these steps make ~~CPU~~ memory access 4 times slower than register access.

2

(iv) After the program is loaded into the memory, the execution begins.
As the CPU does the execution it is called process. Operating system generates a track ID of the processes and takes care of any system requests, such as Interrupt. Once the program finishes its execution; the allocated memory is freed.

2

133745

Rough Work

(v) Instruction pointer contains the address of the instruction in the case 00920080h. The CPU fetches the instruction from this address and the address of IP is incremented by 1. CPU then decodes the instruction. After decoding it is revealed that the instruction contains operands. Thereafter; CPU fetches the address of these ~~data~~ operands. Then the execution of the instruction is begun and the address fetches of X1 is then placed in a temporary memory area - register in this case EBX.

(vi)

Names	How they are accessed
Data Segment	CS
Code Segment	DS
Stack Segment	SS
Three Extra Data Segment	ES, FS, GS

The segments are accessed using general purpose segment register.

Rough Work

viii) Example code

```
var1   DWORD 10, 20, 30

mov   eax, offset var1
```

You can use offset operator before the source operand.

2

viii) Code labels are defined by an identifier; such as

```
X1:
    :
    JUMP  X1
```

1.5

(ix)

```
.data
var1   DWORD 54A6B73h

.code
main  PROC

 mov  ax, WORD PTR var1
```

2

The PTR operator is written after the desired size to and then it is followed by the data identifier.

Rough Work

133745

(X) A directive tells the assembler how to perform a certain instruction. for example

MyWORD ~~BYTE~~ db 10 ;

the db directive tells the assembler to allocate 1 byte of memory for a MyWORD variable.

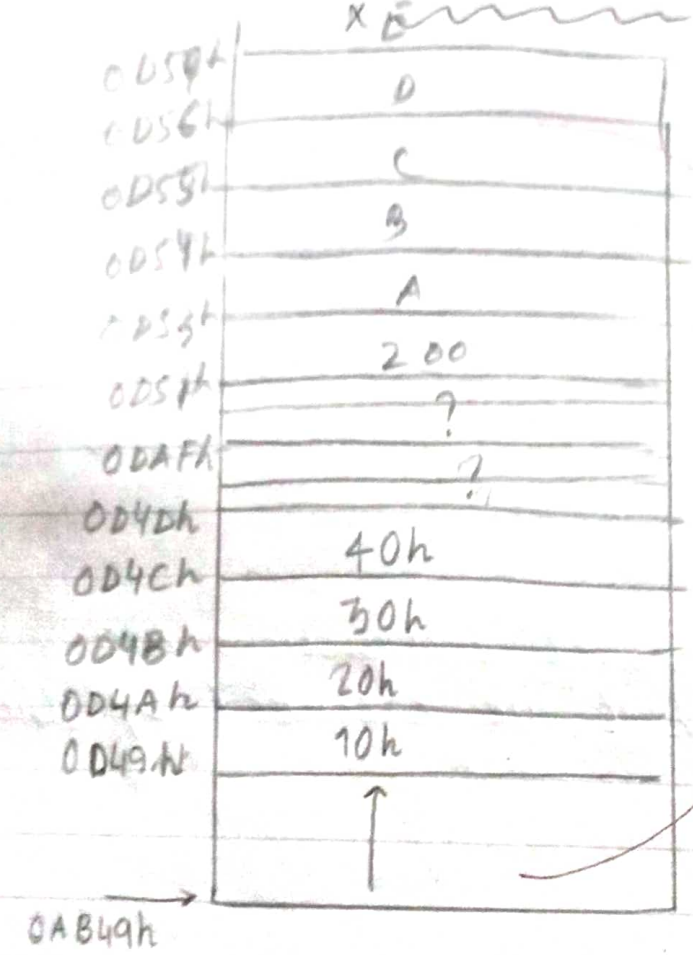
On the other hand instructions are statements that are executed by the program.

2

Rough Work

Question # 02

i)



2

ii)

a. 2 00000002
b. 4 00000004
c. 5 00000005
d 0050

2

iii)

a CF = 1 SF = 0 ZF = 0 OF = 0

b CF = 0 SF = 0 ZF = 0 OF = 1

1.5

133745

Rough Work

7 Question # 03

i) Title Task 3_1

```
.386
.model small
.stack 4096h
.data
dword word 20h, 30h, 50h
ptr offset dword
.code
main PROC

mov si, [$-dword]
mov [si+1], [dword+2]
mov [si], [dword+1]
mov [si-1], [dword]
mov [si-2], 100h

        exit Process
        EndP
End main
```

7

2.5 ★

Task (ii)

```
.386
.model small
.stack 4096h
.data

bArray  BYTE  100 DUP(?)
wArray  WORD  100 DUP(?)
dArray  DWORD 100 DUP(?).
.code
main PROC

mov eax,0
mov ebx,0
mov ecx,100

mov si, offset bArray
mov di, offset wArray
mov dl, offset dArray
X1:
    mov ax,[wArray+di]
    mov bx[bArray+si]
    Sub ax, BYTE PTR bx.
    mov dArray[dl], ax
      LOOP X1
    exit PROCESS
     ENDP
End main
```

3