ASHMAL ANIS 19K-0305 sec:-H

COAL ASSIGNMENT

Date 29/Oct/2020

QUESTION #01:-

(i) Assembly language is used for direct hardware manipulation, we prefer assembly language over high level languages when we want to work on low level systems such as addressing performance issues.

(ii) Another application of assembly is writing operating systems or drivers where close interaction with hardware is required (working on bits 0/1)

(ii) Assembly language occurs at 3rd level right below high-level language which is 4th level after which comes ISA and Digital logic

(iii) Memory takes more machine cycles because they are not inside CPU unlike registers for which addressing time is reduced or abolished. Also, registers are very fast and expensive than memory, they are made up of materials (gold etc) which very quickly transmits information or data

(iv) We override the declared size of an operand into smaller destination through PTR operator.

Example:

```
.Data
    Var1 Dword 0FFAAGh
.code
    move ax, PTR word Var1      ; ax = AAG
```

(v) a. Indirect addressing

```
MOV EAX, [ESI]
```

b. Based Index

```
MOV AX, Array[13]
```


Teacher's Signature

Date

QUESTION # 02

i) TITLE MyProgram (Test-agm)
INCLUDE IRVINE32.inc
.data
T1 word 1
T2 word 1
nextTerm word ?

.code
main PROC

mov ax, T1 // just to display 1 and 1 in ax
mov ax, T2
mov ecx, 5

L1: mov bx, T1
add bx, T2
mov nextTerm, bx

mov dx, T2 // swapping contents of T1 & T2
mov T1, dx

mov dx, nextTerm
mov T2, dx

mov ax, nextTerm
loop L1

Call DumpRegs
exit
main ENDP
END main

Page No.

Teacher's Signature

Date

```
.data
Mid1 word 10h, 17h, 13h, 15h, 20h, 16h
Mid2 word 12h, 13h, 14h, 16h, 18h, 16h
MidTermTotal Dword 6 Dup(?)
.code
main PROC
    mov ecx, lengthof Mid1
    mov esi, 0
    mov edi, 0
L1:
    mov ax, [Mid1 + esi]
    add ax, [Mid2 + esi]
    movzx ebx, ax
    mov MidTermTotal[edi], ebx
    add esi, 2
    add edi, 4
Loop L1

call DumpRegs
exit
main ENDP
END main.
```

Teacher's Sign

Date

QUESTION #03:-

i) The instruction execution cycle of push instruction works as follows. ① The CPU will fetch the instruction from instruction queue. ② Then the push instruction is decoded by the decoder to check its bit by bit pattern. ③ Next, operands which in this case is EAX, is fetched and their values are obtained. ④ The instruction is finally executed in ALU, CU or mainly CPU. ⑤ Lastly in this case value of EDX will be written in stack (temporary storage)

OFFF 201C	0000 000C
OFFF 2018	00FF C126
OFF 204	0000 0020

ii)

EAX	EDX	ECX
0000 0026	CF1170A	0000 000C

iv)

ZF	CF	SF	PF
1	0	1	1

v)

0000 0007	07
0000 0008	00
0000 0009	00
0000 000A	00
0000 000B	0A
0000 000C	97
0000 000D	11
0000 000E	0F

Teacher's Signature