BSCS . COAL ASSIGNMENT #2 ...-454

i) In real addressing mode, the 16bit Segment registers indicite the base address of pre-assigned memory ares - Segment. In pibtcted mode, Segment registers hold printer to Segments. In descriptor tables. The discripter describes the location, length and access rights of the memory Segment.

ii) memory Specified as RAM outside the Cpu and it responds more slowly to acc revent ... registers are hand-... to Cpu so the Cycle required are lesser in Comparisum to memory (RAM)

iii) Assembly has Complete Control over a System's resources so more efficient Calc Can be performed. Assembly allow has small number of operations, it is less Complex and easier to debug.. Embeded System programming multiple devices. use assembly language and game programer use it as it optimised space and speed.

iv) The program counter holds address of the next address that will get executed, the Cpu fetch the instruction from the instruction queue. The Instruction address hold in instruction pointer, next the Cpu decodes the instruction by looking at its binary pattern. If operand involved. Cpu fetches ...

executes the program then the result saved in register or memory (RAM).

vi) A machine Cycle defines itself as the step that gets performed by the processor getting employed in a device and all the instruction that get implemented An instruction Cycle is a process by ... a Compiles takes an instruction given by a Program, then it and executes it from memory. for steps of memory Cycle is fetch, decode, execute and Store. In instruction Cycle these steps are fetch, decode, execute and sum.

The Java programming is based on the Virtual machine Concept. A program written in Java is translated by Java Compiler into Java byte Code. A low level language quickly executed at run by a program called Java Virtual machine (JVM). The Java Byte Code is always the Same on different OS, which have Java Program independent.

The Pc holds the address of instruction it sent to instruction register using it then decoded, the address by are fetched that are the operands register value in this instance. The ALU performs the operation of ADD

are shared.

vii) mov 2020h, AL : An Immediate value Cannot Save register. Using address, by then its decoded, the operands are fetched that are the register value in that instance. The ALu then performs the operations of ADD on the value. By the Data By the values are shared.

viii) mov 2020h, AL : An Immediate value Cannot Save register value. Register ilve Cannot be moved in immediate value.

B) mov 2x ax, bx : There is nothing entend in this instruction as the sizes are Same for both register.

C) mov al, WORD PTR [ebx] : Most be of Save Size the illegal.

D) add [al], [ch] : the Command must have an index of a base register.

E) INC 1Ah : Nu. register to Share value, the operation Cannot be executed on immediate value.

(ix) $\frac{16 \text{ bit} + 32 \text{bit} + (1656)}{8} = 1326 \text{ byte}$

(x) AL AX : 3F7A
CF = 1 AX = A7A2

Scanned with CamScanner

Sf = 0

Zf = 0

) BX = FA77

FA78

Cf = 0

Sf = 1

Zf = 0

) (i) Segment = 50AEh

Offset = 53D9h

Real Address : 5B4B9.

(ii) Segment = 089Eh

offset = B3f63h

Real Address = BC993h

Segment = E823h

offsent = 50ADh

Real Address : ED32Dh

3) (i) . data

A word 10

B word 20

. code

XCHG A, B

Call Dump Regs

ii) . data

X1 WORD 1,2,3,4,5,6,7,8,9,10,11,12,13,

v. X_2 WORD 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
32, 33, 34, 35, 36, 37 → 38, 39, 40

: Code

```
mov  ax, [x1]
mov  bx, [x2]
xchg [x1], bx
xchg ax, [x1+2]
mov  bx, [x2+2]
mov  [x2+2], ax
xchg [x2+2], bx
xchg [x1+2], [x1+10]
mov  ax, [x1+10]
mov  bx, [x2+10]
xchg [x2+10], ax
xchg [x1+10], bx
```

vi

```
mov ax, [x1+6]
mov bx, [x2+6]
xchg [x2+6], ax
xchg [x1+6], bx
mov ax, [x1+8]
mov bx, [x2+8]
xchg [x2+8], ax
xchg [x1+8], bx
ax, [x1+10]
bx, [x2+10]
[x2+10], ax
[x1+10], bx
```

```
mov ax, [x1+12]
mov bx, [x2+12]
xchg [x2+12], ax
xchg [x1+12], bx
mov ax, [x1+14]
mov bx, [x2+14]
xchg [x2+14], ax
xchg [x2+14], bx
mov ax, [x1+16]
mov bx, [x2+16]
xchg [x2+16], bx
xchg [x1+16], ax
mov ax, [x1+18]
```

```
mov ax, [x1+26]
mov bx, [x2+26]
xch, [x1+26], bx
xchg [x2+26], ax
mov ax, [x1+28]
mov bx, [x2+28]
xchg [x1+28], bx
xchg [x2+28], ax
mov ax, [x1+30]
mov bx, [x2+30]
xchg [x1+30], bx
xchg [x2+30], ax
mov ax, [x1+32]
mov bx, [x2+32]
```


Scanned with CamScan

```
mov bx, [x2+18]
xchg [x2+18], ax
xchg ax, [x1+18], bx
mov ax, [x1+20]
mov bx, [x2+20]
xchg [x1+20], bx
xchg [x2+20], ax
mov ax, [x1+22]
mov bx, [x1+22]
mov bx, [x2+22]
xchg [x1+22], bx
xchg [x2+22], ax
mov ax, [x1+24]
mov bx, [x2+24]
xchg [x1+24], bx
xchg [x2+24], ax
xchg [x1+38], bx
xchg [x2+38], ax
mov ax, [x1+40]
mov bx, [x2+40]
xchg [x2+40], ax
xchg [x1+40], bx
```

```
mov bx, [x2+32] bx
xchg [x1+32], ax
xchg [x1+32], bx
mov bx, [x2+34] bx
xchg [x2+34], ax
mov ax, [x1+36]
mov bx, [x2+34]
xchg [x1+34], bx
xchg [x2+34], ax
mov ax, [x1+36]
mov bx, [x2+36]
xchg [x1+36], bx
xchg [x2+38], ax
mov ax, [x1+38]
mov bx, [x2+38]
xchg [x1+38], bx
```


Scanned with CamScan