(2017) Manik sir

1.(a)

The bit size of a microprocessor is determined by the hardware, specifically the width of the data bus. The intel 8086 is a 16-bit processor because it can move 16 bits at a time over the data bus.

-.-

(b)

[2016 - 1(b)]

(c)

Addressing mode: The different ways in which a processor can access data are called addressing modes.

There are several types of addressing modes. Such as:

i) Immediate addressing mode
ii) Register addressing mode
iii) Direct addressing mode
iv) Register based indirect addressing mode.
v) Register relative addressing mode.
vi) Base indexed addressing mode.
vii) Relative base indexed addressing mode.
viii) Implied addressing mode.

i) Immediate AM:

```
mov   CL, 012H
```

This instruction moves 012 immediately into CL register.

CL ← 012H

ii) Register AM:

mov AX, BX

This instruction copies the contents of BX register into AX register. i.e. AX ← BX.

iii) Direct AM:

mov CL, [4321H]

This instruction moves data from location 4321H in the data segment in CL.

Here, the physical address is = 54321H [Let DS = 5000H]

∴ CL ← [54321H]

iv) Register based indirect AM:

mov [CX], [BX]

This instruction moves a word from the address pointed by BX and BX+1 in data segment into CL and CH respectively. CL ← DS: [BX], CH ← DS: [BX+1]

v) Register relative AM:

move CL, [BX+04H]

This instruction moves a byte from the address pointed by BX+4 in the data segment to CL.

CL ← DS:[BX+04H]

vi) Base indexed AM:

mov CL, [BX+SI]

This instruction moves a byte from the address pointed by BX+SI in data segment to CL. CL ← DS:[BX+SI]

vii) Relative base indexed AM:

mov CL, [BX + DI + 20]

This instruction moves a byte from the address pointed by BX+DI+20 in the data segment to CL. CL ← DS: [BX+DI+20]

viii) Implied AM:

In this mode, the operands are implied and are hence not specified in the instruction.

Example: STC

This sets the carry flag.

(d)

2(a)

The advantage of using assembly language instead of writing a program directly in machine language is,

It is easier to correct errors and modify program instructions. Assembly language has the same efficiency of execution as the machine level language. Because this is one-to-one translator between assembly language program and its corresponding machine language program.

— • —

(b)

The difference between unconditional Jump and conditional jump is:

unconditional jump instruction:

Transfers the program sequence to the described memory address. Here no condition needs to be applied. Exp: Jmp label.

conditional jump instruction:

Transfers the program ~~seqen~~ sequence to the described memory address only, If the condition is satisfied.

Example:

JE label

[Jump if equal]

~~(c)~~

```
        mov   cx, 600h
DLY:    DEC   CX
        NOP
        JNZ   DLY
```

i) Here, Dec cx means; this instruction will decrement the value of cx by 1. So JNZ DLY instruction will execute (600h - 1) = 5ffh that means 1535 times.

ii) If we want that JNZ DLY execute 120 times. then the program will be

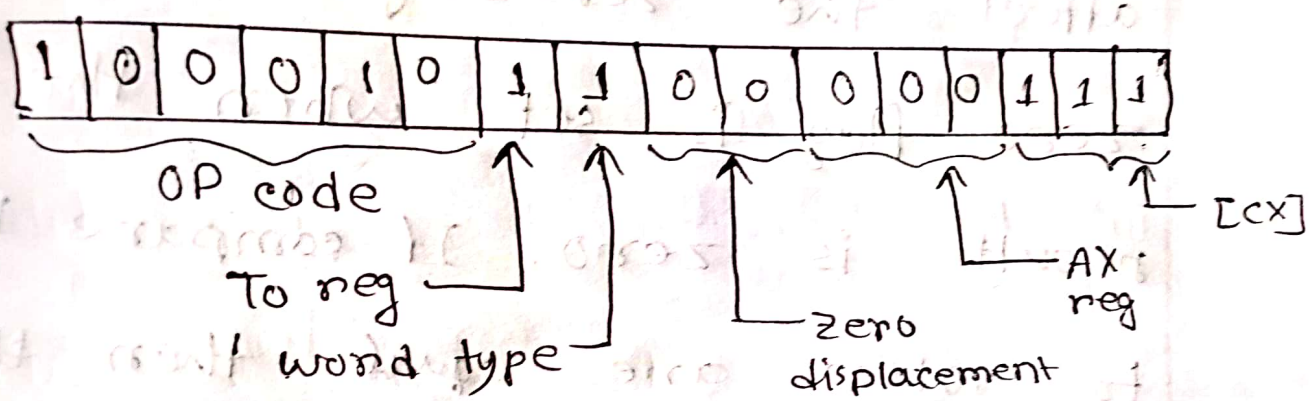
```
        mov   cx, 079h      [121 in decimal]
DLY:    DEC   CX
        NOP
        JNZ   DLY
```

—o~

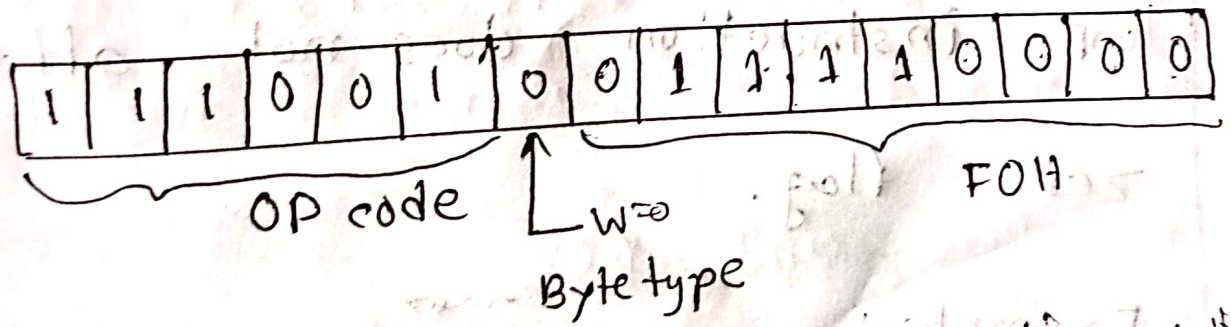
(c)

i) AND BL, 0AH

ii) MOV AX, [CX]



iii) IN AL, F0H



This means, the IN instruction will copy a byte from port F0H to AL.

(d)

The loop instruction decreases the cx register. Because cx register is used for looping by default. But It doesn't affect the zero flag because zero flag is set when the corresponding result is zero. If compare between to values are equal then the differcen will be zero. Then zero flag will be set. otherwise loop instruction does not affect zero flag.

Example:

```
mov cx, 05h
for:
    ax, 01h
    dx, 04h
```

```
cmp ax, dx
jne not equal
to
inc ax
loop for
not equal:
ret
```

Here, For loop instruction, cx will be decreased but ZF will not be set. when ax=dx, then the cmp ax, bx result will be zero, then ZF will set. That means loop instruction can't affect zero flag.

3(a)

i) REP and LOOP

'REP' instruction means, repeat string instruction until specified conditions Exist.

'LOOP' means Jump to specified Label if cx ≠ 0 after auto decrement.

ii) STC and STD

'STC' means set the carry flag to a '1' doesn't effect other flags.

'STD' means set the direction flag to a '1'.

iii) JMP and JB

'JMP' means an unconditional jump to a specified destination.

'JB' means Jump if below. This instruction is used when referring to the magnitude of unsigned numbers.

iv) ROL and RCL

'RCL' means rotate operand around to the Left through ~~carry flag~~.

'ROL' means Rotate all bits of operand left.

Every instruction has its own designation.

(b)

Here, 8086 system frequency is 4MHz.

∴ Time for each clock cycle is $\left(\frac{1}{4MHz}\right)$

$= 0.25$ ns

for 5seconds delay, clock cycle needed,

$$C_T = \frac{5}{0.25 \times 10^{-9}} = 2 \times 10^{10}$$

$C_0 = 4$

$C_r = 4$

AFF0H = 45040D

[For, mov bx, N)

[For, Repeat: mov cx, AFF0H
1st time

Now,

$$C_T = C_0 + C_r + N[45040(2+16) - 12 + 2 + 3 + 16] - 12$$

$$\therefore C_T = C_0 + C_r + 810729N - 12$$

$$\therefore N = \frac{C_T + 12 - C_0 - C_r}{810729} = \frac{(2 \times 10^{10}) + 12 - 4 - 4}{810729}$$

$$\simeq 24670 = 605EH \text{ (Ans.)}$$

(c)

(d) There are 9 flag registers. 3 of them are control flags and 6 of them are status flag or condition flag register

Function of condition flags:

carry flag:

CF → [1 = if unsigned overflow accured
0 = otherwise]

AF → [Auxiliary flag, set to 1 when an unsigned overflow for low nibble]

PF → [1 = even number of '1' bits
0 = odd " of '1' "]

SF → [1 = when result is negative
0 otherwise

ZF → [zero flag is set (1) when the result is zero]

OF → [overflow flag is set (1) when a signed overflow accured]

4(a)

(b,c) Shuvo sir এর part

(d) 2016 → 3(b) (Same ans)

function of control flags:

Direction flag (DF) → 0 = processing is done forward

1 = processing is done backward

Interrupt flag IF → 1 = processor will recognize the interrupts from peripherals

0 = the interrupts will be ignored.

Trap flag, TF → 1 → it'll work in a single step mode. after each instruction, one internal interrupt is generated.

0 → otherwise