Unit - 8
PC - II

Q) Discuss with an appropriate example for jump, loop and compare instructions of 8086.

i) Jump - JMP

Syntax: JMP destination

destination is a memory location. Jump instn breaks the normal sequence of program execution, and takes the control to a different location in the code segment.

```
        MOV ...
        ADD ...
        JMP AGAIN
        - - - - -
        - - - - -
        .
        .
        .
        - - - - - -
AGAIN:  - - - ...
```

ii) LOOP label

~~This is a very useful and frequently used~~ loop combined jump with a counter. The register Cx is assigned to decrement every time loop executes. When Cx = 0, the loop is exited.

```
        MOV CX, N
MORE:   - - - - -
        - - - - -
        LOOP MORE
        - - - - -
```

iii) CMP destination, source

The instruction compares the two operands, causes the conditional flags to be affected, but source & destination doesn't change

IF	CF	ZF
destination > source	0	0
destination < source	1	0
des = source	0	1

Q.) What is meant by masking? With an example explain how it is used with Logical AND operation.

→ Masking is used to select the part of a word or a byte needed, while making the unwanted bits to be zero. It is associated with AND operation.

Ex: MOV AL, 78H
AND AL, 07H

This gives AL = 07H in AL register

Q) Search a character string for a particular character.

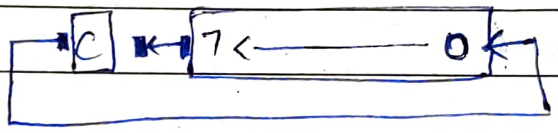
```
.MODEL SMALL
.DATA
STRIN DB "HOWAREYOUMY BOY"
LEN   DW 0EH
MSG1  DB 0AH,0DH," FOUND$"
MSG2  DB 0AH,0DH," NOT FOUND$"
.CODE
LEA BX,STRIN
MOV CX,LEN

MOV AH,01
INT 21
REPEA: CMP [BX],AL
JE FOUND
INC BX
LOOP REPEA
LEA DX, MSG2
JMP DISP

LEA DX, MSG1
MOV AH,09
INT 21H
.EXIT
END
```

Q.) Illustrate with example, rotate through carry left & rotate through carry right.

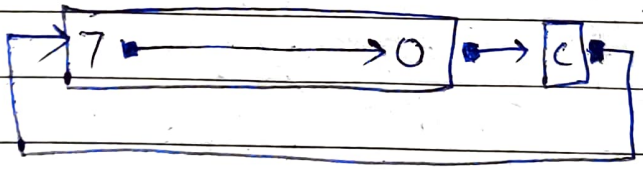
→ Rotate left through carry :

This instruction causes the left most bit In this, MSB enters the CF, and CF gets int LSB position.

Ex: RCL BL, CL



→ Rotate carry through right:



The instruction causes the right most bit to enter the CF and CF enters through the left end.

RCR BYTE PTR [SI], 1

PC - III

Q) Prerequisite for string operation

→ i) Data segment & Extra segment should be initialized & used. Data segment is the source segment & ES is the destination segment

ii) The DI register and SI registers should act as pointers. SI should contain the offset of 1st location of DS and DI should contain offset of first location of ES.

iii) DF should be set or reset so after one operation, it's gets auto incremented or decremented.

iv) Cx should be loaded with the count of number of opⁿ required.

(Q.) STOS & LODS

STOS

The STO instruction is the mnemonic for 'storing' in memory. For this, ES is used and is pointed by DI at it is destination segment. The data to be stored is placed in AL/AX.

LODS : It's an instruction for 'loading' Source memory is the DS & destination segment is AL/AX.

Ex. for stods

```
        .MODEL SMALL
        .DATA
Area DW 50 DUP(?)
        .CODE
        .STARTUP
    MOV AX, DS
    MOV ES, AX
    LEA DI, Area
    MOV CX, 50
    CLD

    MOV AX, 0001
    REP STOSW
        .EXIT
        .END
```

Q.) Distinguish b/w near & far call.

→ Near Call :

i) Direct CALL.

Syntax: CALL Label

OPCODE	OFFSET LOW	OFFSET HIGH

NEW IP = Current IP + OFFSET (16 bit)

Format of the direct near call

Ex:

```
CALL MULTI
```

ii) Indirect Call

Syntax; CALL reg16

In this destination is specified in 16 bit register or in a memory location pointed by register.

Ex - CALL BX

→ FAR CALL :

i) Direct far call

A far call is an intersegment call, which means that destination is in a different code segment.

OPCODE	IP LOW	IP HIGH	CS LOW	CS HIGH

Format of the far CALL

ii) Indirect FAR CALL

For an indirect call, the destination address is not to be in the instruction. It is in register.

EX: CALL WORD PTR [SI]

Q) Discuss with an appropriate example for passing a parameter through stack and through memory.

→ Passing parameter through memory:

The data which is used by procedure is accessed from memory through the pointer register BX, & data is put is put back in memory in same way.

EX:

```
ONE_SET PROC NEAR
        MOV AL, [BX]
```

Passing parameter through stack

The procedure takes data from stack for it's computation

EX:

```
COMPUTE PROC NEAR
        MOV BP, SP
```