Unit III : Instruction Set of 8086 Microprocessor.
(Weightage - 16 marks)

3.1 Machine Language Instruction format.

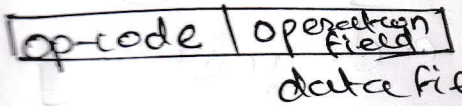
Questions.....

Q) Draw machine language instruction format for Register to Register.....(2M)

Instruction Format

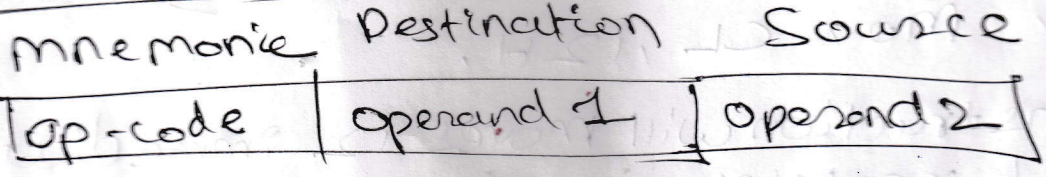
A machine language instruction format has more than one number of field.

- 1st field is called as operation code field or op-code → indicates the type of operation to be performed by CPU.
- 2nd field is called as operand field → Indicates data field on which the operation by instruction op-code.
- It has 6 general format of instruction in 8086 instruction set.
- Length of any instruction may vary from 1 byte.



Op-code byte.
- This is byte always present in instruction.
- This indicates the operation to be carried out by 8086.

Instruction Format



3.2 Addressing modes

Questions -----

Q1 Describe any six addressing modes of 8086 with suitable diagram. (6M). ☆☆☆

Q2 Define immediate addressing mode with suitable example. (2M).

Addressing modes of 8086

R⁴ID²B

1. Immediate Addressing mode
2. Register Addressing mode
3. Implied Addressing mode
4. Direct ~~memory~~ addressing mode.
5. Register ~~based~~ indirect addressing mode.
6. Register relative addressing mode
7. Base indexed addressing mode
8. Relative based indexed addressing mode.

Immediate Addressing Mode

An instruction in which 8-bit or 16 bit operand (data) is specified in the instruction, then addressing mode of such instruction is known as Immediate Addressing mode.

Example, Mov CL, 12H.

This instruction will moves 12 immediately into CL register

CL ← 12H.

Register addressing mode

An instruction in which an operand (data) is specified in general purpose register, the known as register addressing mode

Register may be used as source operands, destination operands or both.

Example

```
mov AX,BX
```

The instruction copies the contents of BX register into AX register.

AX ← BX

Direct ~~memory~~ addressing mode

In this mode address of operand is directly specified in instructions.

An instruction in which 16bit effective address of an operand is specified in instruction, then the addressing mode of such instruction is called as Direct addressing mode.

Example,

```
mov CL,[2000H]
```

Register indirect addressing mode

An instruction in which address of an operand is specified in pointer register or in indexed register or BX, then the addressing mode is known as register indirect addressing mode.

Example: `mov AX,[BX]`

Register relative addressing mode

In this mode, the operand address is calculated using one base register and an 8bit or 16bit displacement.

Example `mov AX,50H[BX]` or

`mov AX,[BX+50H]`

so DS* 10H + BX + 50H

Base indexed addressing mode

Here, operand address is calculated as base register plus an index register.

Example

```
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]

So, DS*10H + BX+SI

Relative based indexed addressing mode.

In this mode, the address of operand is calculated as sum of base register, index register and 8bit or 16bit displacement.

```
mov CL, [BX+DI+20]
        Base  Index  displacement.
```

CL ← DS:[BX+DI+20]

So, physical address

DS*10H + BX+DI+20

Implied addressing mode

It is also known as implicit addressing mode.

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

Example,

```
STC
CMC
```

- It sets the carry flag.

Question 1

Identify addressing mode. (4 marks S-23)

i) Mov CL, 34H

⇒ Immediate Addressing mode

ii) Mov BX, [4100H]

⇒ Direct addressing mode

iii) Mov DS, AX

⇒ Register Addressing mode

iv) Mov AX, [SI + BX + 04]

⇒ Relative Base Index addressing mode

Question 2

6 marks S22

1) Mov AX, 3456H

⇒ Immediate Addressing mode

2) ADD BX, [2000H]

⇒ Direct addressing mode

3) DAA

⇒ Implied / Implicit addressing mode

4) Mov AX, [SI]

⇒ Register indirect addressing mode

5) Mov AX, BX

⇒ Register addressing mode

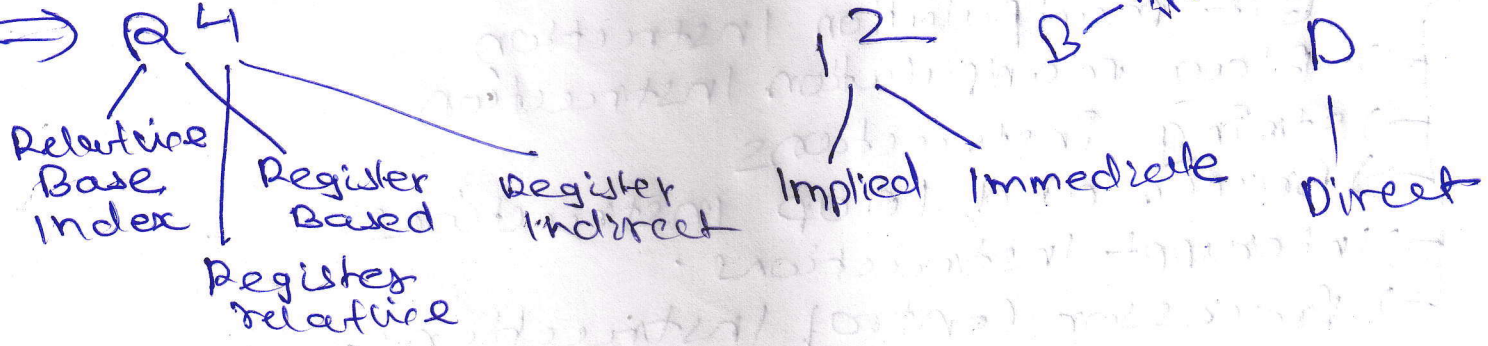
6) SUB AX, [BX + SI + 80H]

⇒ Relative Base Index addressing mode.

Question 3

State addressing modes.

⇒



3.3 Instruction set. Group of instructions: Arithematic instructions, logical instructions. Data transfer instructions, Bit manipulation instruction. string operation instructions. Program control transfer or branching instructions, Process control instructions

Questions.....

Q// write any four bit manipulation instructions of 8086. (2M)

Q// Use of AAM instruction. (2M)

Q// Explain the suitable example.
i) DAA ii) AAM

Q// Explain logical instructions of 8086 ..... 6M

Q// state use of STC & CMC instructions (2M)

Q// Describe any four string instructions of 8086.

Q// Differentiate ROL & RCL. (2M) (6M)

Q// Explain
i) DAA ii) ADC iii) XCHG ... 6M

Q// Describe different branching instructions used in 8086 up in brief. --- 6M

Q// Classification of instruction set of 8086. Explain any one type out of them. 4M

Classification of instruction set.

- Arithematic and Logical Instruction
- Data Transfer Instructions.
- Bit manipulation Instruction
- Flag manipulation Instructions
- string instructions
- Branch and loop instructions
- Interuppt Instructions.
- Processor Control Instructions.

Arithmetic Instructions

These instructions are used to perform arithmetic operations like addition, subtraction, multiplication, division etc....

1) ADD

It adds instruction adds the contents of source operand to destination operand.

Syntax: Add source, destination, source.

Example : ADD AX, BX
ADD AX, 20h
ADD AX, [SI]

2) ADC (Add with Carry)

This instruction performs the same operation as ADD instruction. but adds the carry flag to result.

Example. ADC AX, BX
ADC AX, [SI]

3) SUB

The subtract instruction subtracts the source operand from destination operand.
Result in destination operand.

Syntax Sub destination, source

Example. Sub AX, 0100 H
sub AX, BX
Sub AX, [5000H]

4) SBB (Subtract with Borrow)

The subtract with borrow instruction subtract the source operand and borrow flag (CF) may reflect previous calculation. from destination operand

Syntax : SBB Dest, Src

Exp SBB AX, 0110 H
SBB AX, BX

MUL

It is an unsigned number multiplication instruction

Example MUL BH ; ALXBH

Syntax MUL source

IMUL

It is signed number multiplication

Eg. IMUL BH Syntax IMUL SRC

IMUL [SI]

DIV: unsigned division

IDIV: signed division

CBW (Convert byte to word)

This instruction converts byte in AL to word in AX.

Eg: AX=0000 0000 1001 1000

Result AX=1111 1111 1001 1000

CMP

- It compares two specified bytes or word.
The comparision is done simply by internally
- subtracting source from destination

Syntax: CMP SRC, desc

Example: CMP BX, 0100H

CMP AX, CX

DAA (Decimal Adjust after BCD Addition) (OM)

when two BCD numbers are added, the DAA is used after ADD or ADC instruction to get correct answer in BCD.

This instruction is used to make sure the result of adding two packed BCD number is adjusted to be corrected BCD number.

The result of addition must be in AL for DAA instruction to work correctly.

If lower nibble in AL after addition is >9 or AF is set, then add 6 to lower nibble of AL.

If upper nibble in AL is >9H or CF=1, and then add 6 to upper nibble of AL.

Example

AL=99 BCD BL=99BCD

then, ADD AL, BL

1001 1001 =AL = 99 BCD +

1001 1001 =BL=99 BCD

0011 0010 = AL =32H.

So, CF=1, AF=1

After DAA, result is CF=1

0011 0010 = AL=32 Hr

AL=1+0110 & 0110

∴ 1001 1000 =AL =98 in BCD.

AAM (Adjust result of BCD multiplication)

This instruction is used after MUL of two unpacked BCD.

It stands for ASCII Adjust for multiplication or BCD Adjust after MUL.

Example

multiply 9 & 8

MOV AL, 00000101

MOV BH, 00001001

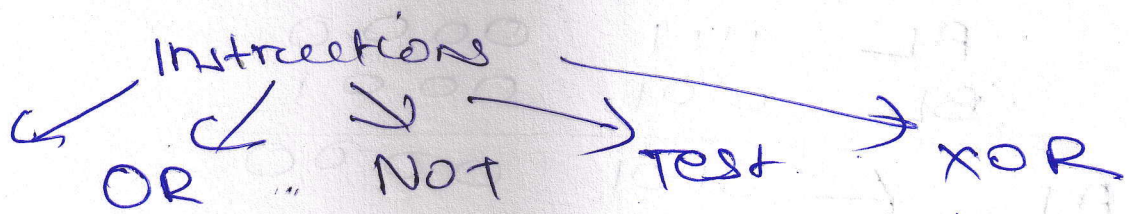
MUL BH.

AAM

Logical Instructions & Bit manipulation Instructions

Logical Instructions are also known as Bit manipulation instructions.

They are used for performing operations such as AND, OR, NOT, XOR etc.---



NOT

Syntax: Not ~~destSource~~ destination

Operation: Destination ← Not destination.

It complements each bit of destination to product 1's complement of specified operand.

Example, not AX

not BL

BL = 0000 0011

Not BL = 1111 1100

No flags are Affected here

AND

Syntax: AND Destination, Source

Operation: Destination ← destination AND source

- It performs AND operation of Destination & Source.
- Source can be immediate number, memory location or register.
- Destination can be register or memory location.
- Both operands cannot be memory locations at same time.
- CF & OF = 0 after operation
- PF, ZF & SF are updated.
- Example And AL, BL

AL	1111	0000
BL	0101	0001
AL ←	0101	0000

OR

It performs OR operation in Des & src

Syntax- OR destination, source

Example- OR

Operation - destination ← desc OR source

example OR AL, BL

AL	0000	0010
BL	1101	0010
AL ←	1101	0010

XOR

It performs XOR operation of Des and Src.

Syntax : XOR destination, source

Operation : Destination ← Desc XOR src

Example: XOR AL, BL

AL	1111	1100
BL	0000	0011
AL ←	1111	1111

TEST

It performs add operation only but result is stored in flag.

Syntax : Test Desc, Src

Flag ← Des AND Src

Eg mov AL, 00000101

Test AL, 1 ; ZF=0

Test AL, 10b ; ZF=1

Data Transfer Instructions

① Mov Des, Src

② Push operand

③ Pop operand

④ IN Accumulator, Port Address

⑤ Out ~~Accum~~ Portaddress, Accumulator

⑥ LEA
Load effective address

⑦ LDS

⑧ XCHG Des, Src (2m)

↳ This instruction exchanges Src with Des

↳ It cannot exchange two memory locations directly

↳ Eg XCHG DX, AX

~~So AX DX~~ Content of AX → DX & DX → AX

Flag manipulation Instruction

STC: It sets carry flag to 1

CLC: Clears carry flag to 0

CMC: Compliments Carry flag.

STD: Set Direction flag.

CLD: Clear Direction flag

STI: Set the interrupt flag

CTI: clears the interrupt flag

String manipulation Instructions.

- String in assembly language is just a sequentially stored bytes or words.
- There are very strong set of string instructions in 8086.
- By using these string instructions, these size of program is considerably reduce.

1) MOVS / MOVSB / MOVSW

- It causes moving of byte or word from one string to another
- In this instruction, the source string is in Data Segment reffered by DS:SI and Destination string is in Extra Segment ES:DI.
- For example,
```
movs str1, st2.
movsb str1, st2
movsw str1, str2.
```

2) LODS / LODSB / LODSW

- It causes Transfer of byte or word from one string to another.
- In this instruction, the source string is in Data Segment reffered by DS:SI transferred to Accumulator.

For example,
```
lods string.
lodsb string
lodsw string
```

3) STOS / STOSB / STOSW

- It causes Transfer of byte or word from one string to another.
- In this string is in Extra Segment reffered by ES:DI transfered to Accumulator.
- For example
```
stos, string
stosw string
stosb string
```

4) CMPS

Syntax CMPS Des, Src

It compares the string bytes or words

Example, CMPS C1, C2

5) SCAS string

- It scans a string.
- It compares string with byte in AL or with word in AX.

6) REP

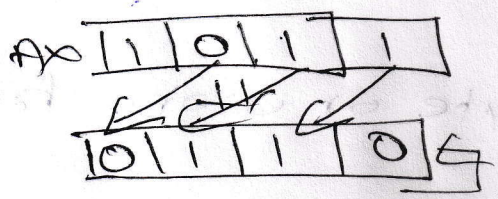
- This is an instruction prefix.
- It causes the repetition of instruction until CX becomes zero.

Example REP MOVSB.

Shift and Rotate instructions

① SHL/SAL Des, Count

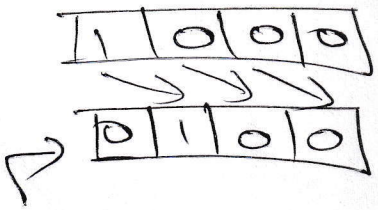
- puts zero in LSB



SAL AX, 01

② SHR/SAR Des, Count

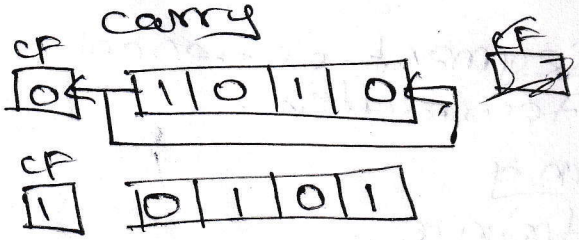
- puts zero in MSB



Shift

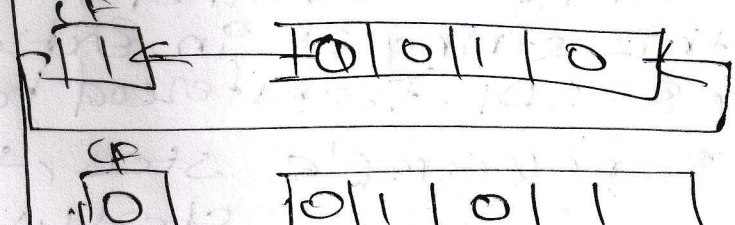
ROL

Rotate left without carry



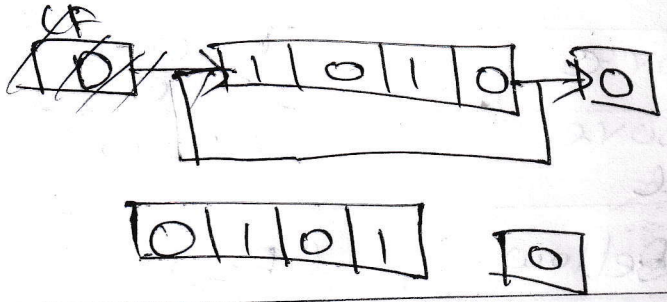
RCL

Rotate left with Carry



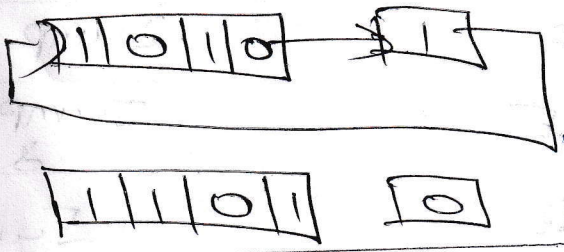
ROR

Rotate right without carry



RCR

Rotate right with carry



Branch / Program Execution Transfer Instruction.

- These instructions cause change in the sequence of the execution of instruction.
- This change can be sometime conditional or unconditional.
- The conditions are represented by flags.

CALL

This instruction is used to call a subroutine function or procedure.

The address of next instruction after call is saved into steack.

Syntax CALL Des

RET

It returns the control from procedure to calling program.

Every call instruction should have a RET.

JMP

This instruction is used for unconditional jump from one place to another.

Syntax. JMP Des.

Jxx Des (Conditional Jump)

mnemonic	Meaning	Jump condition
JA	Jump if Above	CF=0 & ZF=0
JAE	Jump if Above & Equal	CF=0
JB	Jump if Bellow	CF=1
JBE	Jump if Below & Equal	CF=1 or ZF=1
JC	Jump if Carry	CF=1
JE	Jump if Equal	ZF=1
JNC	Jump if not Carry	CF=0
JNE	Jump if not Equal	ZF=0
JNZ	Jump if not zero	ZF=0
JPE	Jump if parity even	PF=1
JPO	odd	PF=0
JZ	Jump if zero	ZF=1

Loop Des

- This is looping instruction.
- The number of times looping is required is placed in CX register.
- With each iteration, the contents of CX are decremented.
- ZF is checked whether to loop again or not.

Question-Answers

Question 1 : select the instructions for each for following. ---- (SU-23.... 6M)

⇒ i) Multiply AL by 05H

⇒
```
mov AL, 05H
MUL AL
```
|
```
mov BL, 05H
MUL BL
```

ii) Move 1234H in DS register

⇒
```
mov AX, 1234H
mov DS, AX
```

iii) Add AX with BX

⇒ ADD AX, BX

iv) Signed Division of Ax by BL

⇒ idiv BL

v) Rotate the contents of Ax towards left by 4 bits through carry

⇒
```
mov CL, 04
RCL AX, CL
```

vi) Load SP register with FF00H

⇒ mov SP, FF00H

Question2

Write appropriate 8086 instructions to perform following operation ---(S-23 ... 6m)

i) Rotate the contents of BX Register towards right by 4 bits.

⇒ MOV CL, 04H
ROR BX, CL

ii) Rotate the content of AX towards left by 2 bits.

⇒ MOV CL, 02H
ROL AX, CL

iii) Add 100H to the content of AX register.

⇒ Add AX, 0100H.

iv) Transfer 1234H to DX register

⇒ MOV DX, 1234H

v) multiply AL by 08H

⇒ MOV BL, 08H
MOV BL

vi) Signed division of BL & AL

⇒ idiv BL

Question3 : Select assembly language for each of following (1*9 = 6M)

i) Rotate register BL right 4 times

⇒ MOV CL, 04H
RCR BL, CL

ii) multiply AL by 04H

⇒ MOV BL, 04H
MUL BL

iii) Signed division of AX by BL

⇒ IDIV BL

iv) Move 2000H in BX register

⇒ MOV BX, 2000H

v) Increment the content of AX by 1.

⇒ INC AX

vi) Compare AX with BX

⇒ CMP AX, BX

Question 4: 6m ---- 5-22

i) Multiply BL by 88H

⇒ mov AL, BL
mov BL, 88H
mov BL

ii) Signed division of AL by BL

⇒ idiv BL

iii) Move 4000H to DS register

⇒ mov AX, 4000H
mov DS, AX

iv) Rotate content of AX register to left 4 times mov CX, 04h

⇒ ROL AX, CX

or

RCL AX, CX

v) Shift content of BX register to right 3 times

⇒ mov CX, 03H

SHR BX, CX

vi) Load SS with FF00H

→ mov SS, FF00H

Questions ... S-19 ... 6M

i) Rotate the contents of DX to write 2 times without carry

⇒ mov CL, 02H
ROR DX, CL

ii) multiply contents of AX by 06H

⇒ mov BX, 06h
MUL BX

iii) Load 4000H in SP register.

⇒ mov SP, 4000H

iv) Copy contents of BX register to CL

⇒ wrong question....

v) Signed division of BL and AL.

⇒ IDIV BL

vi) Rotate AX register to right through carry 3 times.

⇒ mov CL, 03H
RCR AX, CL