Gramin Shikshan Prasarak Mandal's

GRAMIN POLYTECHNIC

(ISO 9001:2008 Certified Institute)

Chapter 3. Instruction Set of 8086 Microprocessor

By

Shinde G. B.

M.TECH. (Electronics Engineering)

Syllabus

3.1 Machine Language Instruction Format.

3.2 Addressing modes.

3.3 Instruction set, group of instructions: Arithmetic instructions, logical instructions, data transfer instructions, bit manipulation instructions, string operation instructions, program control transfer or branching instructions, process control instructions.

Unit Outcomes :

- Determine the length of given instruction.
- Describe the given addressing modes with examples.
- Explain the operation performed by the given instruction during its execution.
- Identify the addressing modes in given instructions.

Instruction Format

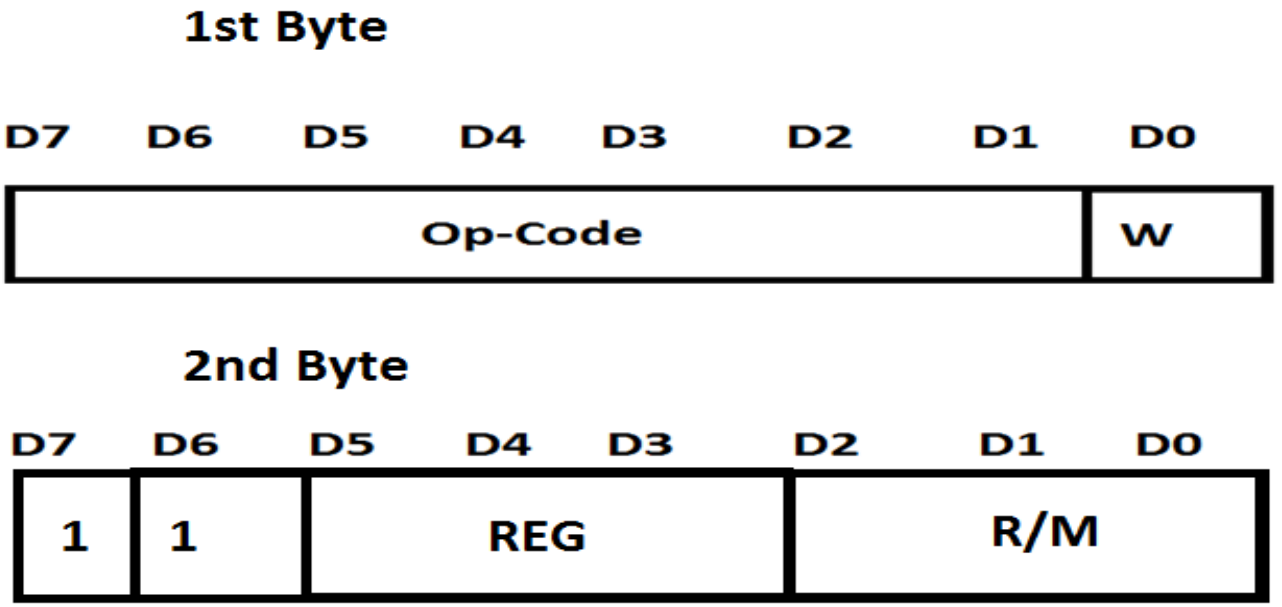
- One byte instruction
- Register to register
- Register to/from memory with no displacement
- Register to/from memory with displacement
- Immediate operand to register
- Immediate operand to memory with 16 bit displacement

One-byte Instruction

- This format is one byte long and may have implicit data or register operands.
- The least significant 3 bits of the op-code are used to specify the register operand.

Register-to-register

- Register to register instruction the format of instruction is of 2 bytes long.



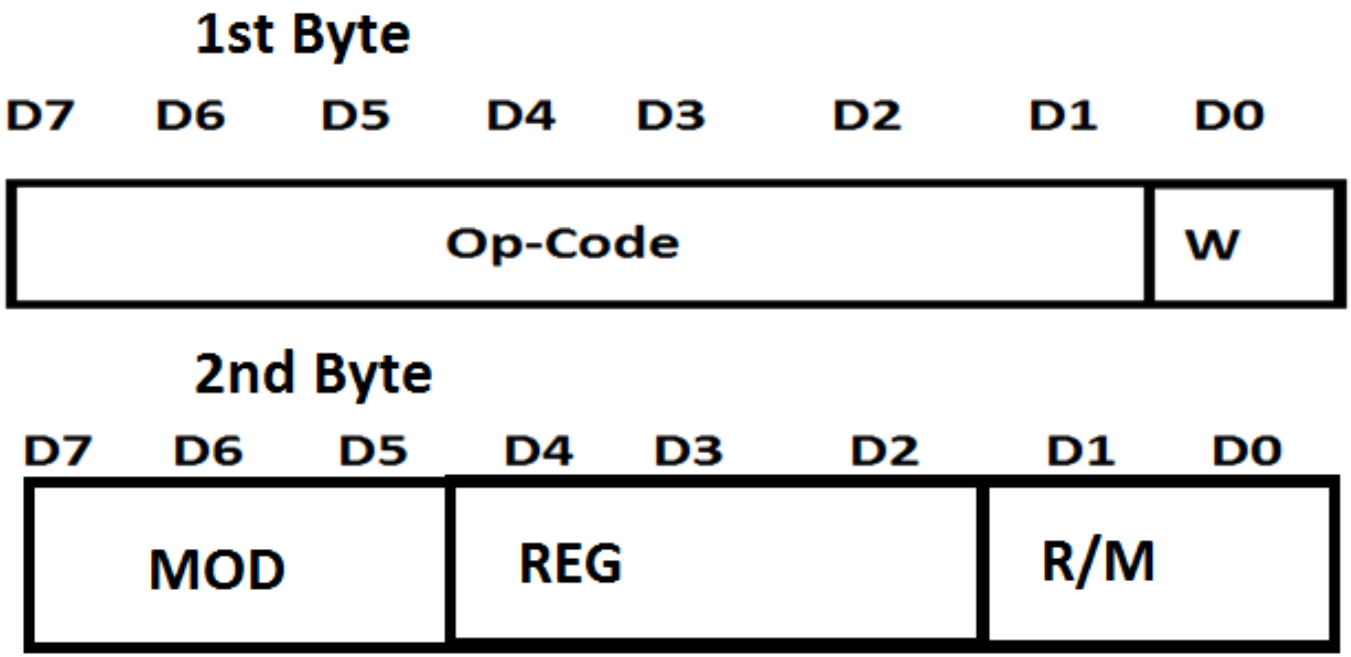
- Register to register instruction the format of instruction is of 2 bytes long.
- The first byte of code indicates the op-code and the width of the operand specified by W bit.
- The second byte of the instruction code indicate the register operand and R/M field.
- REG field is one of the register operand
- R/M indicate another operand which may be register or memory location.

REG	W = 0	W = 1
0 0 0	AL	AX
0 0 1	CL	CX
0 1 0	DL	DX
0 1 1	BL	BX
1 0 0	AH	SP
1 0 1	CH	BP
1 1 0	DH	SI
1 1 1	BH	DI

When W = 0, all 8 bit registers are selected whereas for W = 1 all 16 bit registers are selected.

Register to/from memory with no displacement

- In this type of instruction the format of instruction is of 2 bytes

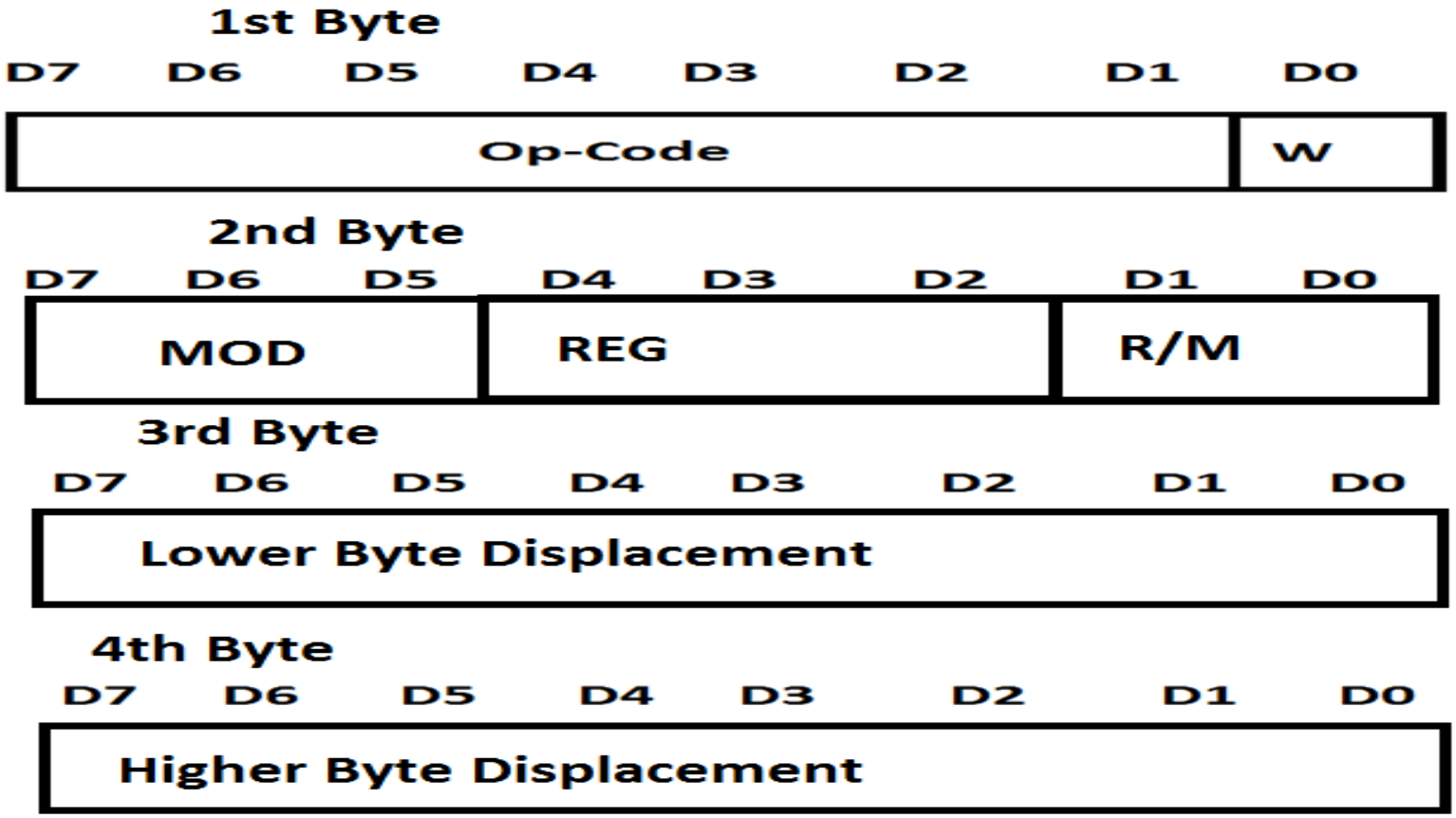


- The first byte is same as that of register to register format but second byte contains the MOD field.
- MOD field indicate whether one of the operand is in memory or whether both operands are registers.

CODE	EXPLANATION
0 0	Memory mode, no displacement follows*
0 1	Memory mode, 8 bit displacement follows
1 0	Memory mode, 16 bit displacement follows
1 1	Register mode (No displacement)

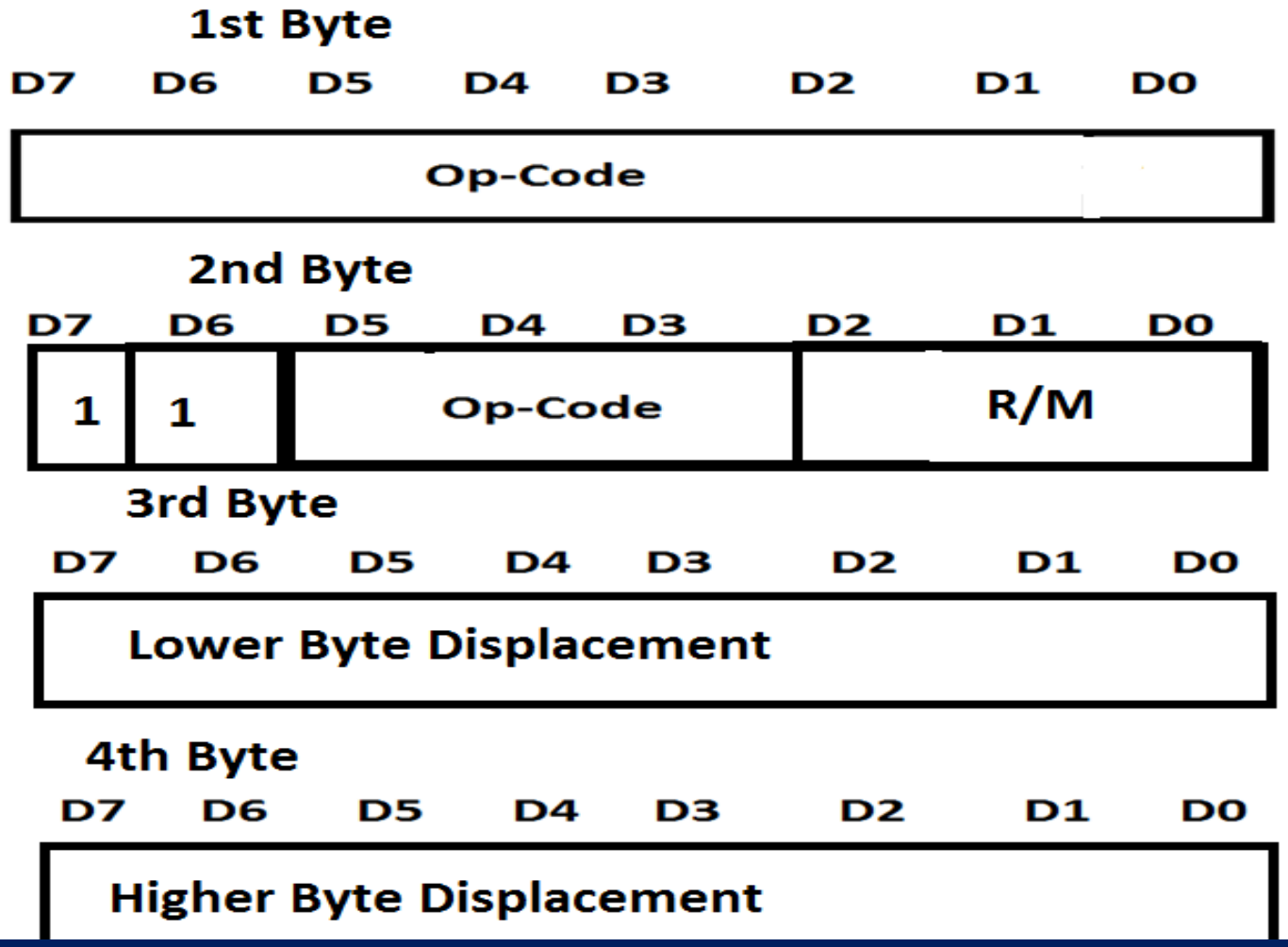
Register to/from memory with displacement

- Its size is 4 byte.



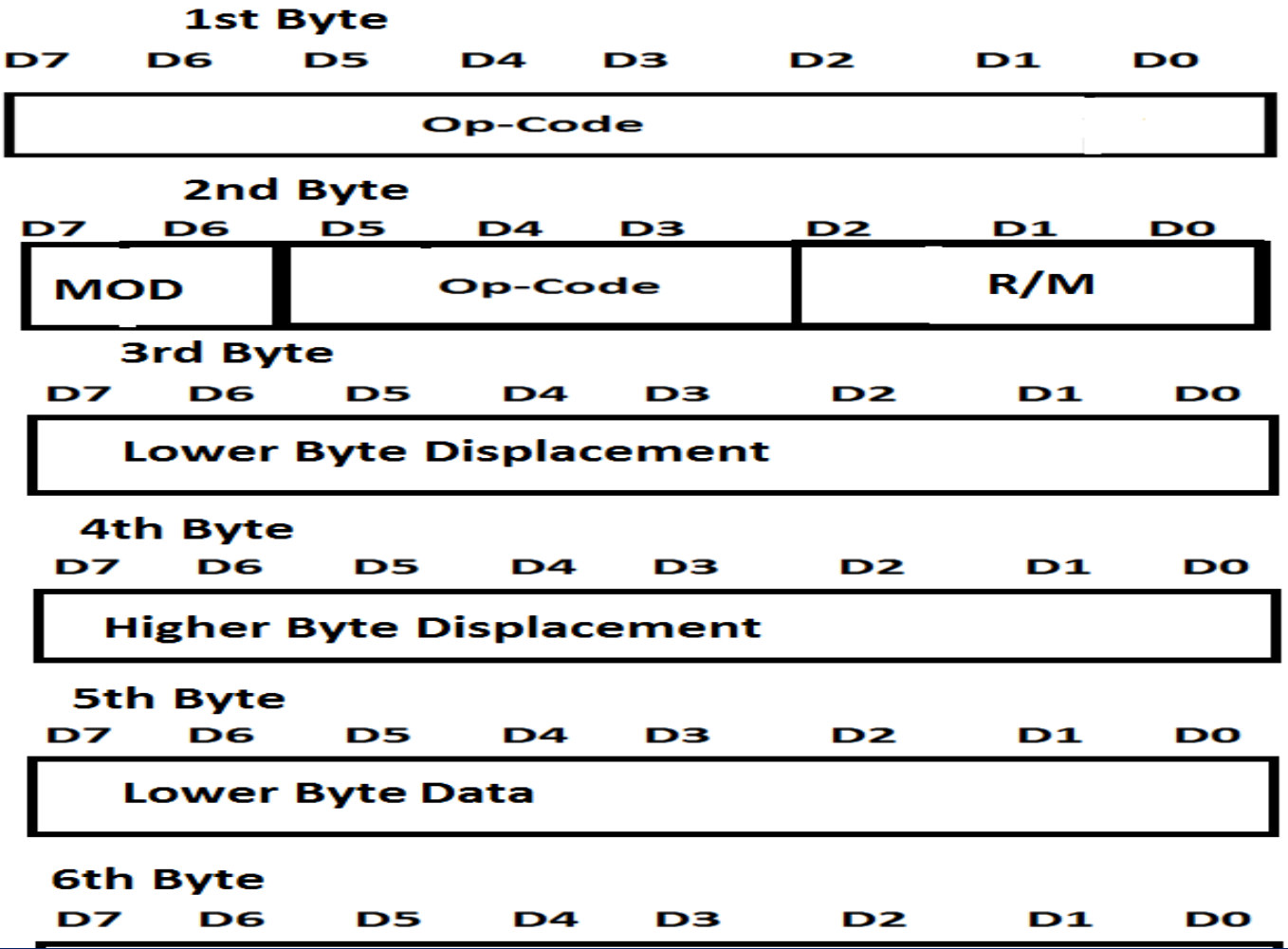
Immediate operand to register

- Its size is 4 byte.



Immediate operand to memory with 16 bit displacement

- Its size is 6 byte.



Addressing Modes of 8086

- Immediate addressing mode
- Direct addressing mode
- Register addressing mode
- Register indirect addressing mode
- Indexed addressing mode
- Register relative addressing mode
- Base indexed addressing mode
- Relative base indexed addressing mode
- Implicit or implied addressing mode

- Addressing mode is used to have the access of the data.
- They gives various way to access data or operand in memory ,registers or I/o.

Immediate addressing mode

- The immediate data is the part of the instruction and appear in the form of successive byte or bytes after the op-code bytes.
- EX:

MOV AL,46H

MOV BX,1234H

Direct addressing mode

- A 16 bit memory address(offset) of operand is directly specified in the instruction as part of it.
- EX:

MOV AL,[3000H]; al will be loaded with the content of memory location whose offset is 3000h from the base address.

AND AX,[8000H]

Register addressing mode

- The data is stored in a register and it is referred using the particular register.
- EX:

MOV AL,BL

AND AL,BL

Register indirect addressing mode

- The address of the memory location which contains data or operand is available in indirect way using offset register such as BX, SI,DI register.
- EX:

MOV AX,[BX] ;copies the content of the memory location whose offset is in BX register.

SUB [SI],AL

Indexed addressing mode

- The offset of the operand is stored in any one of the index registers.i.e.SI and DI
- DS and ES are the default segment register for SI.
- EX:

MOV BL,[SI] ;copies the byte from memory location whose offset is in index register SI to the BL.

ADD AX,[DI+8]

Register relative addressing mode

- The data is available at an effective address formed by adding 8 bit or 16 bit displacement with content(BX,BP,SI,DI).
- EX:

MOV AX,50[BX] ;copies the word from memory location whose offset will be calculated by adding 50 with the content of BX register.

ADD Ax,5000[SI]

Base Indexed addressing mode

- The effective address of data is calculated by adding the content of a base register BX or BP to the content of SI or DI with default segment DS or ES.
- EX:

MOV AX,[BX][SI]

ADD al,[BX][DI]

Relative Base Indexed addressing mode

- The offset address of data is calculated by adding the 8 bit or 16 bit displacement with the sum of base register BX or BP and SI or DI.
- EX:

MOV AX,60[BX][SI] ;copies the word from memory location whose offset is calculated by adding the 60H with the content of BX and SI

Implicit or Implied addressing mode

- Instruction using this mode have no operands. In the instruction itself will specify the data to be operated by the instruction.
- EX:

```
CLC          ; clear the carry flag
DAA          ; decimal adjust after arithmetic
```

- Identify the addressing mode of the following

MUL AL,BL

MOV AX,BX

MOV BX,[SI]

MOV DX,0040H

MOV AX,2050H

STC

INC BX

MOV DS,AX

MOV AX,[4172]

ADD AX,[SI]

MOV AX,2100H

- Identify the addressing mode of the following

ADD AX,[SI][BX][04]

MUL AL,BL

MOV AX,BX

INC[4712]

ADD AX,4712H

DIV BL

MOV AX,[BX+SI]

MOV AX,2034H

MOV AL,[6000H]

ADD AL,CL

MOV AX,50[BX][SI]

Instruction Set of 8086

- Data transfer / Copy instruction
- Arithmetic and logical instruction
- Branch Instruction
- Loop Instruction
- Machine Control Instruction
- Flag Manipulation instruction
- Shift and rotate instructions
- String instructions

Data Copy / Transfer Instructions

MOV

- This instruction is used to transfer the data from source to destination
- Syntax:

MOV destination, source

- Operation:

Destination← Source

EX:

MOV BX,3456H

MOV AL,[3000H]

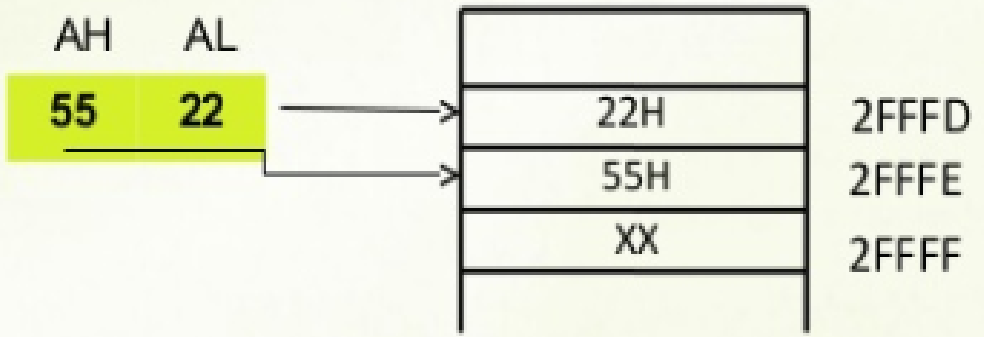
PUSH

- This instruction is used to store word from source on to the stack location.
- Syntax:
- PUSH source
- Operation:
- SP← SP-2
- EX:
- PUSH BX

PUSH: Push to Stack

Ex: PUSH AX
PUSH DS
PUSH [5000H]

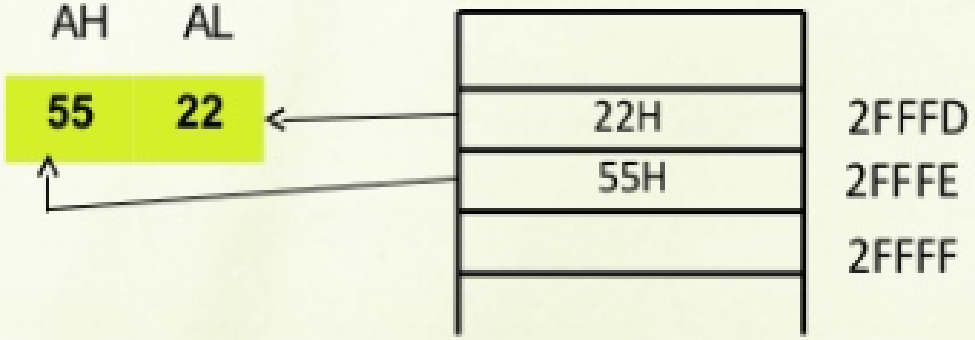
PUSH AX



POP: Pop from Stack

Ex: POP AX
POP DS
POP [5000H]

POP AX



POP

- This instruction is used to store word from stack location pointed by the stack pointer to a destination specified in the instruction.
- Syntax;
- POP destination
- Operation:
- SP← SP+2
- EX:
- POP DX

XCHG

- This instruction exchanges the contents of register with the contents of another register or memory location.
- Syntax;
- XCHG destination, source
- Operation:
- Destination ←> Source
- EX:
- XCHG AX,BX

IN

- This instruction copies data from a port to destination which may be AL or AX.
- Syntax;

  IN accumulator , port
- Operation:

  AL← [port] for bytes

  AL← [port] and AH← [port+1] for word
- EX:

  IN AL,80H ;input a byte from port whose address is 80H

  IN AX,80H

OUT

- This instruction copies data from AL or word from AX to the specified port.
- Syntax;

  OUT port ,accumulator

Operation:

[port] ← AL for bytes

[port] ← AL and [port+1] ← AH for word

- EX:

  OUT 80H,AL

  OUT 80H,AX

LEA

- This instruction determines the offset(effective address) of the variable and loads the offset address of memory location to register.
- Syntax;

  LEA 16bit register, source.

Operation:

16 bit register ← effective address

- EX:

  LEA BX,ARRAY; load BX with the offset of variable ARRAY.

LDS/LES

- This instruction loads content of offset address of memory location to register specified in the instruction.
- Syntax;

  LDS 16bit register, source.
- EX:

LDS BX,[1234H] ;copy the content of memory location 1234H in BL, content of 1235H to BH and the content of 1236H and 1237H to DS register.

LAHF

- This instruction stores lower byte of flag register of 8086 to the AH register.
- EX:
- LAHF

SAHF

- This instruction copies the content of AH register which is used to set or reset the flag in the lower byte of the flag register of 8086.
- EX:
- SAHF

PUSHF

- This instruction used to store flag register on to stack.
- SP is decremented by 2
- EX:
- PUSHF

POPF

- This instruction used to store word from memory location at the top of stack to the flag register.
- SP is incremented by 2
- EX:
- POPF

Arithmetic Instructions

- These instructions perform the arithmetic operations like addition ,subtraction , multiplication and division.

ADD/ADC

- The ADD instruction adds a number from some source to destination.
- Flags Affected:
- OF,CF,PF,AF,SF,ZF
- Operation

Destination←Destination+source

Destination←Destination+source+CF

EX:

ADD AL,74H

ADC AX,1234H

SUB/SBB

- The SUB/SBB instruction subtracts a number from some source to destination.
- Flags Affected:
- OF,CF,PF,AF,SF,ZF
- Operation

Destination – Destination - source

Destination – Destination – source -CF

EX:

SUB AL,74H

SUB AX,BX

INC

- The instruction adds 1 to the indicated destination.
- Flags Affected:
- OF,PF,AF,SF,ZF
- Operation

Destination ← Destination+1

EX:

INC AX

DEC

- The instruction subtracts 1 from the indicated destination.
- Flags Affected:
- OF,PF,AF,SF,ZF
- Operation

Destination ← Destination - 1

EX:

DEC AX

CMP

- The instruction make a comparison between source and destination.
- Source and Destination will not changed but flags are affected.
- Flags Affected:
- OF,CF,PF,AF,SF,ZF.
- Syntax:

cmp destination,array

EX:

CMP AL,0FFH

CMP AX,BX

- If destination> source then CF=0 ZF=0 SF=0
- If destination<source then CF=1 ZF=0 SF=1
- If destination=source then CF=0 ZF=1 SF=0

DAA

It is a decimal adjust after arithmetic

This instruction is used after ADD/ADC

The ADD/ADC instruction adds the two BCD numbers in hexadecimal format and DAA instruction converts this hexadecimal result to BCD result.

Syntax:

DAA

DAS

It is a decimal adjust after substraction

This instruction is used after SUB/SBB

The SUB/SBB instruction subtract the two BCD numbers in hexadecimal format and DAS instruction converts this hexadecimal result to BCD result.

Syntax:

DAS

NEG

- The instruction converts the number byte/word in a destination in the 2's complement.
- Flags Affected:
- OF,CF,PF,AF,SF,ZF
- Operation

Destination ← 2nd complement of destination

EX:

NEG AX

MUL

- This instruction is used to multiply an unsigned byte from source with an unsigned byte in AL register.
- Flags Affected
- OF,CF and PF , AF,SF,ZF are undefined.
- Operation
- 1. AL←AL * unsigned 8 bit source
- 2. DX:AX← AX * unsigned 16 bit source
- EX
- MUL BL

IMUL

- This instruction is used to multiply an signed byte from source with an signed byte in AL register.
- Flags Affected
- OF,CF and PF , AF,SF,ZF are undefined.
- Operation
- 1. AL←AL * signed 8 bit source
- 2. DX:AX← AX * signed 16 bit source
- EX
- IMUL BL

DIV

- This instruction is used to divide an unsigned word by an unsigned byte.
- After the division the AL will contain an quotient and AH will contain an 8 bit remainder.
- After the division the AX will contain the 16 bit an quotient and DX will contain an 16 bit remainder.
- Flags Affected
- OF,CF and PF , AF,SF,ZF are undefined.
- Operation
- If source is byte then
- AL←AL/ unsigned 8 bit source
- AH← AL MOD unsigned 8 bit source
- If source is word then
- AX←DX:AX / unsigned 16 bit source
- DX←DX:AX MOD unsigned 16 bit source

IDIV

- This instruction is used to divide an signed word by an signed byte.
- After the division the AL will contain an quotient and AH will contain an 8 bit remainder.
- After the division the AX will contain the 16 bit an quotient and DX will contain an 16 bit remainder.
- Flags Affected
- OF,CF and PF , AF,SF,ZF are undefined.
- Operation
- If source is byte then
- AL←AL/ signed 8 bit source
- AH← AL MOD signed 8 bit source
- If source is word then
- AX←DX:AX / signed 16 bit source
- DX←DX:AX MOD signed 16 bit source

Logical Instructions

- These instructions perform the logical operations like AND , OR ,NOT and XOR.

AND

- This instruction AND's bit by bit the source operand with destination operand and result is stored in destination specified in the instruction.
- Flags affected
- CF=0,OF=0,PF,SF,ZF
- Syntax:
- AND destination,source
- Operation:
- Destination←Destination AND Source
- EX:
- AND BH,CL
- AND BX,00FFH

OR

- This instruction OR's bit by bit the source operand with destination operand and result is stored in destination specified in the instruction.
- Flags affected
- CF=0,OF=0,PF,SF,ZF
- Syntax:
- OR destination,source
- Operation:
- Destination←Destination OR Source
- EX:
- OR BH,CL
- OR BX,00FFH

XOR

- This instruction Exclusive OR's bit by bit the source operand with destination operand and result is stored in destination specified in the instruction.
- Flags affected
- CF=0,OF=0,PF,SF,ZF
- Syntax:
- XOR destination,source
- Operation:
- Destination←Destination XOR Source
- EX:
- XOR BH,CL
- XOR BX,00FFH

NOT

- This instruction inverts each of the byte or word at the specified destination i.e.1's complement.
- Syntax:
- NOT destination
- Flags Affected:None
- Operation:
- Destination←NOT destination
- EX:
- NOT BX
- NOT [4000H]

TEST

- This instruction ANDs the content of a source byte or word with the contents of specified destination byte or word and flags are updated but neither operands are changed.
- Syntax:
- TEST destination,source
- Flags Affected:CF,OF,PF,SF and ZF
- Operation:
- Flags← set for result of(dest AND source)
- EX:
- TEST BH,CL

SHL/SAL

- This instruction shifts each bit in the specified destination counts times towards left.(0 is inserted)
- Syntax:
- SAL destination,count
- Flags Affected:CF,OF,PF,ZF and AF is undefined
- Operation:
- CF← MSB← ---------- LSB ←0
- EX:
- CF=0 BX=1110 0101 1101 0011
- SAL BX,1
- CF=1 BX=1100 1011 1010 0110

SAR

- This instruction shifts each bit in the specified destination counts times towards right.
- Syntax:
- SAR destination,count
- Flags Affected:CF,OF,PF,ZF and AF is undefined
- Operation:

- MSB ---------- --> LSB → CF
- EX:
- CF=0 BX=1110 0101 1101 0011
- SAR BX,1
- CF=1 BX=1111 0010 1110 1001

SHR

- This instruction shifts each bit in the specified destination counts times towards right.(0 is inserted)
- Syntax:
- SHR destination,count
- Flags Affected:CF,OF,PF,ZF and AF is undefined
- Operation:

- 0→ MSB ---------- --> LSB → CF
- EX:
- CF=0   BX=1110 0101 1101 0011
- SHR BX,1
- CF=1   BX=0111 0010 1110 1001

ROR(Rotate right without carry)

- This instruction rotates all of the bits of the specified byte or word count times towards right.
- Syntax:
- ROR destination,count
- Flags Affected:OF,CF
- Operation:
- MSB ---------- --> LSB → CF

- EX:
- CF=0 BL=0011 1011
- ROR Bl,1
- CF=1 Bl=1001 1101

ROL(Rotate Left without carry)

- This instruction rotates all of the bits of the specified byte or word count times towards left.
- Syntax:
- ROL destination,count
- Flags Affected:OF,CF
- Operation:
- CF← MSB <---------- -- LSB

- EX:
- CF=0 BL=1011 1010
- ROL BL,1
- CF=1 BL =0111 0101

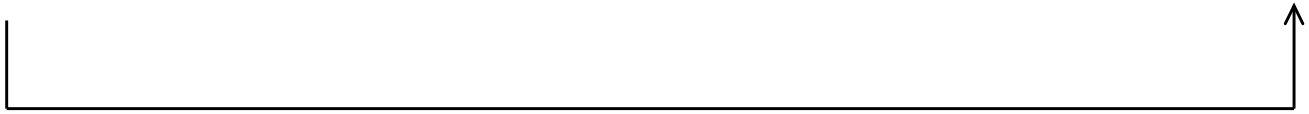
RCR(Rotate Right with carry)

- This instruction rotates all of the bits of the specified byte or word count times towards right.
- Syntax:
- RCR destination,count
- Flags Affected:OF,CF
- Operation:
- MSB LSB CF

- EX:
- CF=0 BL=0011 1011
- RCR BL,1
- CF=1 BL =0001 1101

RCL(Rotate Left with carry)

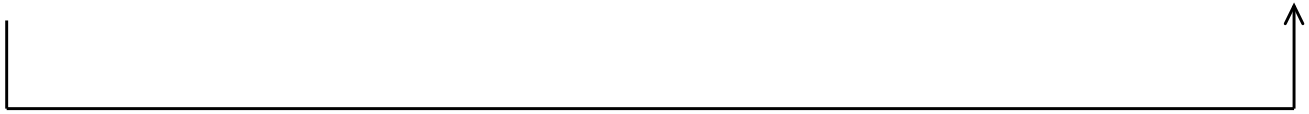
- This instruction rotates all of the bits of the specified byte or word count times towards left.
- Syntax:
- RCL destination,count
- Flags Affected:OF,CF
- Operation:
- CF ← MSB ← LSB



- EX:
- CF=1 BL=0011 1011
- RCL BL,1
- CF=0 BL =0111 0111

RCL(Rotate Left with carry)

- This instruction rotates all of the bits of the specified byte or word count times towards left.
- Syntax:
- RCL destination,count
- Flags Affected:OF,CF
- Operation:
- CF ← MSB ← LSB



- EX:
- CF=1 BL=0011 1011
- RCL BL,1
- CF=0 BL =0111 0111

Machine Control Instructions

HLT: Halt

- The instruction HLT causes the processor to enter the halt state.
- The CPU stops fetching and executing the instructions.
- Syntax:
- HLT

NOP: No Operation

- This instruction is used to add wait state of three clock cycles and during these three clock cycles CPU does not perform any operation
- Syntax:
- NOP

WAIT

- The instruction WAIT causes processor to enter into an ideal state or a wait state and continues to remain in that the processor receives state until ne of the following signal.
- Signal on processor TEST pin
- A valid interrupt on INTR pin
- A valid signal on NMI pin
- Syntax:
- WAIT

LOCK

- This instruction prevents other processors to take the control of shared resources.
- Syntax:
- LOCK
- EX:
- LOCK IN AL,80H

Flag Manipulation Instruction

- This type of instructions are used to changed the status of flags in the flag register such as
- CF,DF,IF

- CLC[Clear Carry]
- CMC [Complement Carry]
- STC [Set Carry]
- CLD [Clear Direction Flag]
- STD [Set Direction Flag]
- CLI [Clear Interrupt Flag]
- STI [Set Interrupt Flag]

String Manipulation Instructions

- A string is contiguous block of byte or word and can used to hold any type of data or information that will fit into bytes or words.

MOVS: Move String

MOVSB : Move String Byte

MOVSW : Move String Word

- The source must be placed in data segment and destination must be in extra segment.
- DS:SI,ES:DI
- Operation:
- ES:[DI]←DS:[SI]
- If byte movement
- For DF=0 SI←SI+1 and DI←DI+1
- For DF=1 SI←SI-1 and DI←DI-1
- If word movement
- For DF=0 SI←SI+2 and DI←DI+2
- For DF=1 SI←SI-2 and DI←DI-2

EX

- MOV AX,@data
- MOV DS,AX
- MOV ES,AX
- CLD
- MOV SI,OFFSET S_STRING
- MOV DI,OFFSET D_STRING
- MOVS S_STRING,D_STRING

LODS: Load String

LODSB : Load String Byte

LODSW : Load String Word

- The instruction LODS transfer a byte or word from the source string pointed by SI in DS to AL for byte or AX for Word.
- Operation:
- If byte movement
- AL←DS:[SI]
- For DF=0 SI←SI+1
- For DF=1 SI←SI-1
- If word movement
- AX←DS:[SI]
- For DF=0 SI←SI+2

EX

- MOV AX,@data
- MOV DS,AX
- MOV ES,AX
- CLD
- MOV SI,OFFSET S_STRING
- LODS S_STRING

STOS: Store String

STOSB : Store String Byte

STOSW : Store String Word

- The instruction STOS transfer a byte or a word from the AL for byte and AX for word to destination string pointed by DI in ES.
- Operation:
- If byte movement
- ES:[DI]←AL
- For DF=0 DI←DI+1
- For DF=1 DI←DI-1
- If word movement
- DS:[DI]←AX
- For DF=0 DI←DI+2

EX

- MOV AX,@data
- MOV DS,AX
- MOV ES,AX
- CLD
- MOV DI,OFFSET D_STRING
- STOS D_STRING

CMPS: Compare String
CMPSB : Compare String Byte
CMPSW : Compare String Word

- The instruction CMPS compares a byte or word in the source string with a byte or word in the destination string.

EX

- MOV AX,@data
- MOV DS,AX
- MOV ES,AX
- CLD
- MOV SI,OFFSET S_STRING
- MOV DI,OFFSET D_STRING
- CMPS S_STRING,D_STRING

SCAS: Scan String
SCASB : Scan String Byte
SCASW : Scan String Word

- The instruction SCAS scans a string byte or word with byte in AL and Word in AX.
- The destination string is pointed by DI in ES.

EX

- MOV AX,@data
- MOV DS,AX
- MOV ES,AX
- CLD
- MOV DI,OFFSET D_STRING
- MOV AL,'V'
- SCAS D_STRING