

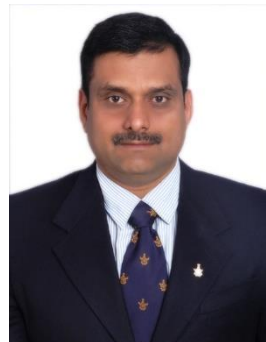


Microprocessors and Interfaces: 2021-22

Lecture 5

8086 Addressing Modes and OP-Code : Part-1

By Dr. Sanjay Vidhyadharan



Types of Instructions

- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Branch and Program control Instructions

Addressing Modes

Instruction = **Opcode**, **Operand**

Opcode/ Operation Field - the type of operation which is to be performed by processor

Operand – the data on which the operation is going to be performed

Addressing Modes

- Register Addressing
- Immediate Addressing
- Direct Addressing
- Register Indirect Addressing
- Base-plus-index Addressing
- Register Relative Addressing
- Base relative -plus-indexed Addressing
- **Scaled Indexed Addressing**

These data-addressing modes are found with all versions of the Intel microprocessor. except for the **scaled-index-addressing** mode, found **only** in 80386 through Core2

MOV Instruction

- MOV destination, source

Data Transfer Instructions

- **MOV DST, SRC**

- Copies the content of source to destination
- No Flags Affected
- Size of source and destination must be the same

Different MOV options

R ← **M**

M ← **R**

R ← **R**

M ← **I**

R ← **I**

Addressing Modes

- **Register Addressing**

- MOV AX, BX

- **Immediate Addressing**

- MOV AX, 1420_H

Addressing Modes

- **Direct Addressing**

- MOV AX, [2340_H]

- **Register Indirect Addressing**

- MOV AX, [BX]

Addressing Modes

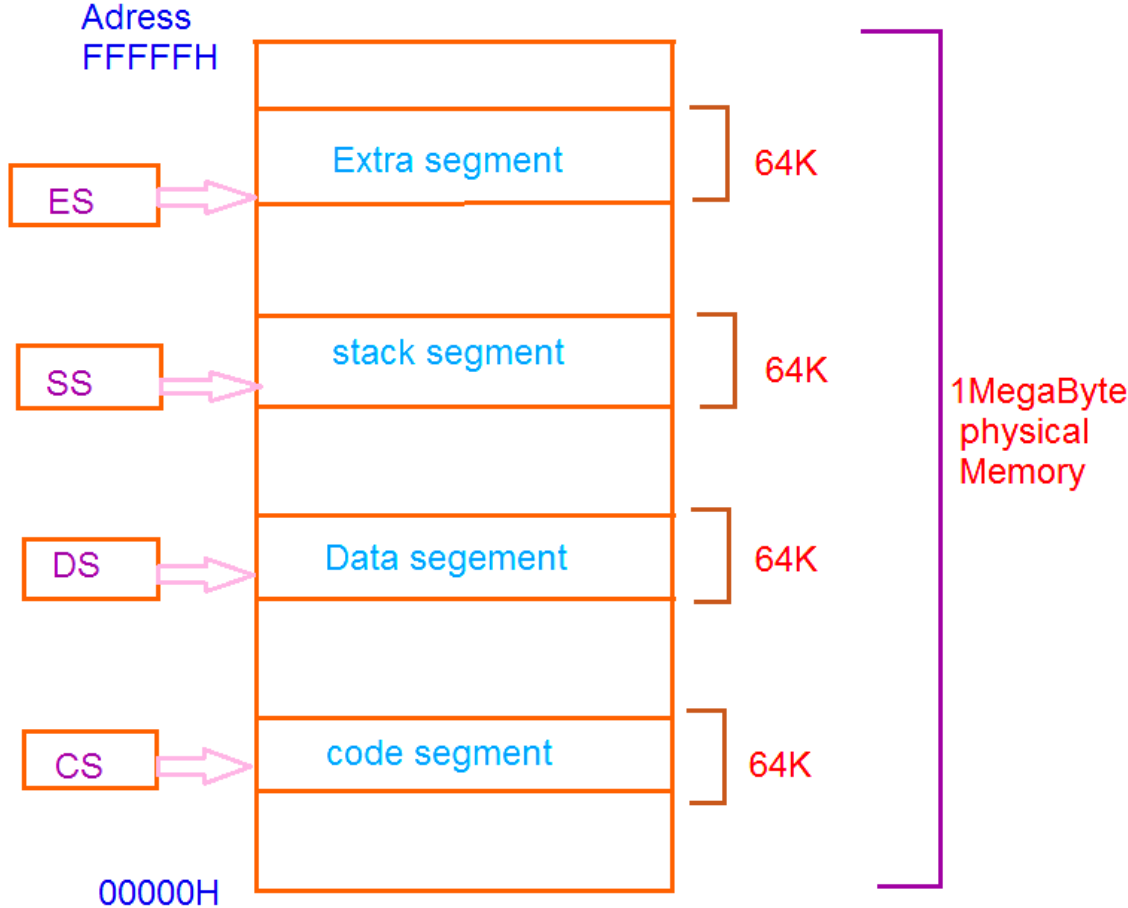
- **Base-plus-index Addressing**

- MOV AX, [BX+SI]

- **Register Relative Addressing**

- MOV AX, [BX+40]

Addressing Modes



Addressing Modes

- **Base relative-plus-indexed Addressing**
 - MOV AX, [BX+SI+10]

- Scaled Indexed Addressing

Addressing Modes

- The microprocessor contains these **8-bit register names** used with **register addressing**: AH, AL, BH, BL, CH, CL, DH, and DL.
- 16-bit register names: AX, BX, CX, DX, SP, BP, SI, and DI.
- In 80386 & above, extended **32-bit register** names are: EAX, EBX, ECX, EDX, ESP, EBP, EDI, and ESI.
- **64-bit mode register** names are: RAX, RBX, RCX, RDX, RSP, RBP, RDI, RSI, and R8 through R15.

- **Important for instructions**

If hexadecimal data begin with a letter, the assembler requires the data start with a **0**.

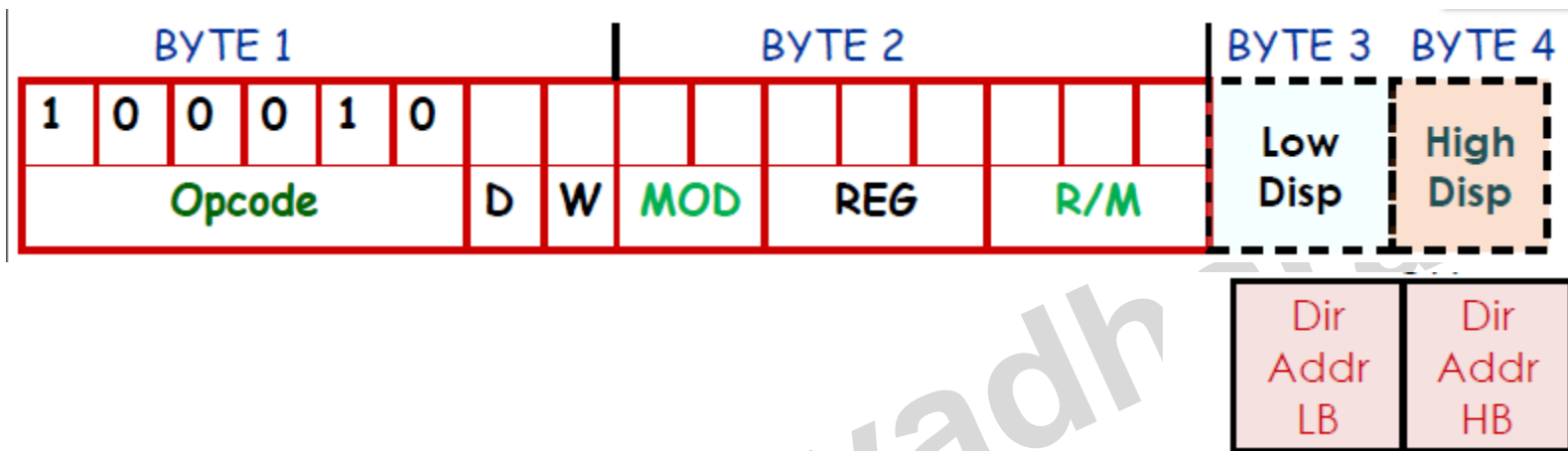
- to represent a hexadecimal F2, **0F2H** is used in assembly language

Sanjayvidhyadharan.in

Important for instructions

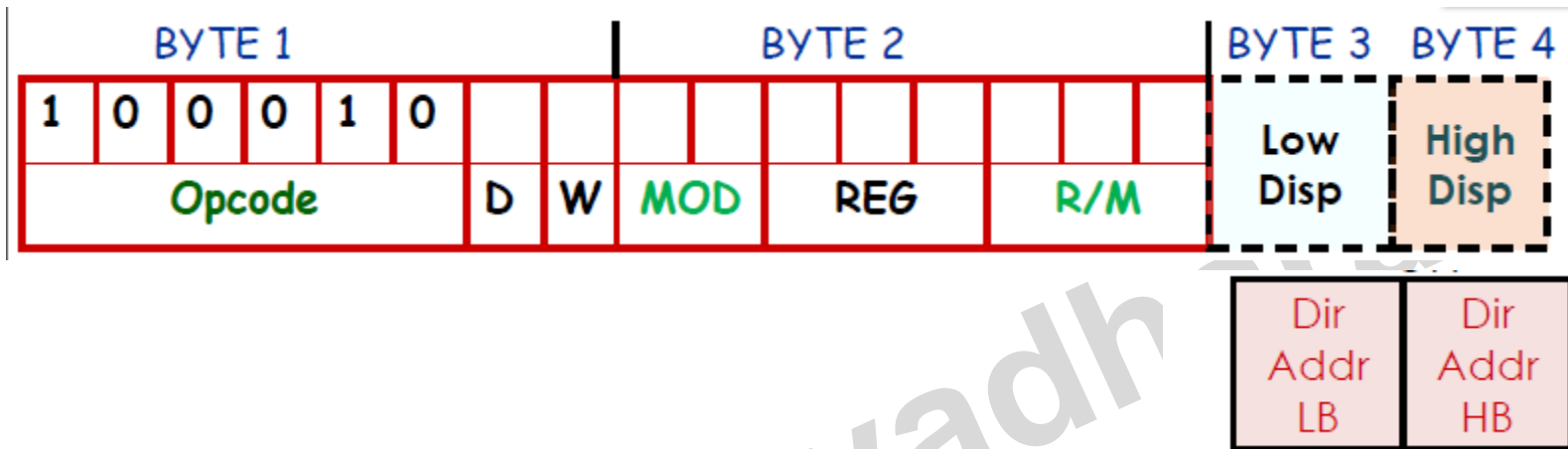
- The source register's contents do not change.
- the **destination register's** contents **do change**
- The contents of the destination register or destination memory location change for all instructions except the **CMP** and **TEST** instructions.
- The MOV BL, CL instruction does **not affect** the **leftmost** 8 bits of register BX.

8086 Assembly Language to Machine language



- **Byte 1 contains three kinds of information:**
 - Opcode field (6 bits) specifies the operation such as add, subtract, or move
 - Register Direction Bit (D bit)
 - Tells the register operand in REG field in byte 2 is source or destination operand
 - 1: Data flow to the REG field from R/M
 - 0: Data flow from the REG field to the R/M
 - Data Size Bit (W bit)
 - Specifies whether the operation will be performed on 8-bit or 16-bit data
 - 0: 8 bits
 - 1: 16 bits

8086 Assembly Language to Machine language



OP Code

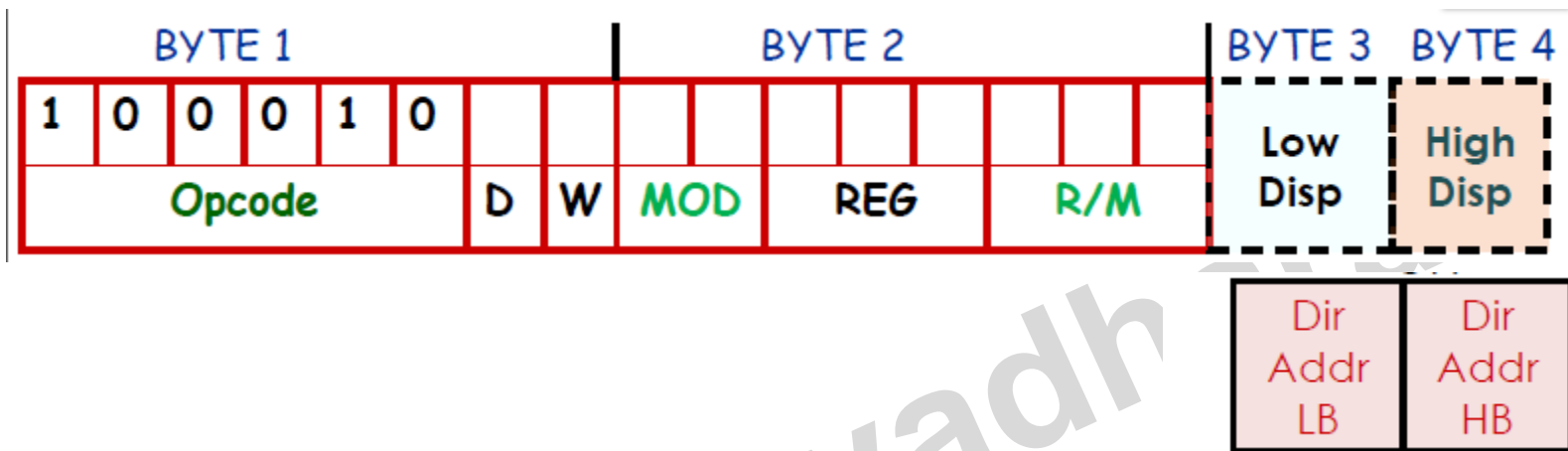
MOV = Move	76543210	76543210
Register/Memory to/from Register	100010 dw	mod reg r/m
Immediate to Register/Memory	1100011 w	mod 000 r/m
ADD = Add:		
Reg/Memory with Register to Either	000000 dw	mod reg r/m
SUB = Subtract		
Reg/Memory and Register to Either	001010 dw	mod reg r/m

8086 Assembly Language to Machine language

- REG field is used to identify the register for the first operand

REG	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

8086 Assembly Language to Machine language



Byte 2 has 3 fields

- Mode field (MOD) – 2 bits
- Register field (REG) - 3 bits
- Register/memory field (R/M field) – 2 bits

8086 Assembly Language to Machine language

The 2-bit MOD field specifies whether the operand is in register or memory as follows:

MOD	Interpretation
00	Memory mode with no displacement follows except for 16-bit Displacement when R/M = 110
01	Memory mode with 8-bit displacement
10	Memory mode with 16-bit displacement
11	Register mode (no displacement)

8086 Assembly Language to Machine language

Operands	Memory Operands			Register Operands	
	No Displacement	Displacement 8-bit	Displacement 16-bit		
MOD	00	01	10	11	
R/M				W = 0	W = 1
000	(BX) + (SI)	(BX) + (SI) + D8	(BX) + (SI) + D16	AL	AX
001	(BX) + (DI)	(BX) + (DI) + D8	(BX) + (DI) + D16	CL	CX
010	(BP) + (SI)	(BP) + (SI) + D8	(BP) + (SI) + D16	DL	DX
011	(BP) + (DI)	(BP) + (DI) + D8		BL	BX
100	(SI)	(SI) + D8	(SI) + D16	AH	SP
101	(DI)	(DI) + D8	(DI) + D16	CH	BP
110	D16	(BP) + D8	(BP) + D16	DH	SI
111	(BX)	(BX) + D8	(BX) + D16	BH	DI

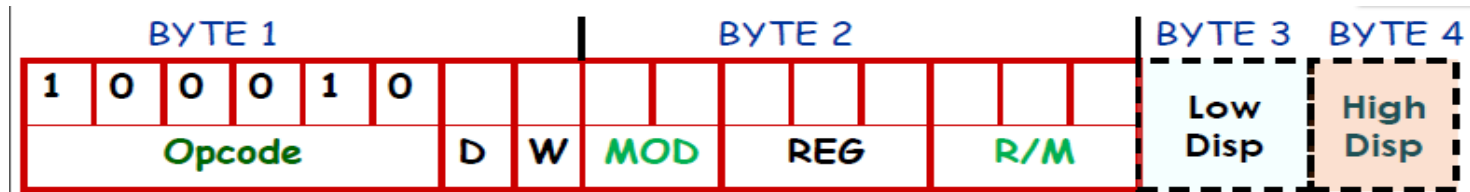
MOD = 11 Register Mode

- MOV BL,AL
- Opcode for MOV = 100010
- We'll encode AL so
 - D = 0 (AL source operand)
- W bit = 0 (8-bits)
- MOD = 11 (register mode)
- REG = 000 (code for AL)
- R/M = 011

OPCODE	D	W	MOD	REG	R/M
100010	0	0	11	000	011

MOV BL,AL => 10001000 11000011 = 88 C3h

MOD = 00 Memory operand with no displacement



MOV [BX],CL

- **w = 0** because we are dealing with a byte
- **d = 0** because REG to R/M
- therefore first byte is **(1000 1000) = 88H**
- since no displacement,
- we can use **MOD=00 REG=001 and R/M=111 = 0000 1111 = 0FH**

result: 88 0F

MOD = 10 Memory operand with 16 bits displacement

MOV BP [SI+ 500H], 7293H

OPCODE	W	MOD	OPCODE	R/M
1 1 0 0 0 1 1	1	1 0	0 0 0	0 1 0
C	7	8		2

LOWER BYTE DISP.	HIGHER BYTE DISP.
0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 1
0	5

Displacement 500H

LOWER BYTE DATA	HIGHER BYTE DATA
1 0 0 1 0 0 1 1	0 1 1 1 0 0 1 0
9	7

Data 7293 H

The complete machine code comes out to be C7 82 00 05 93 72.

MOV = Move

Register/Memory to/from Register

Immediate to Register/Memory

76543210

100010 dw

1100011 w

76543210

mod reg r/m

mod 000 r/m

MOD = 01 Memory operand with 8 bits displacement

- **MOV [BX+10h] ,CL**
 w = 0 because we are dealing with a byte
 d = 0 because we need R/M Table 2 to encode [BX+10h]
- therefore first byte is **10001000 = 88H**
- since 10H can be encoded as an 8-bit displacement, we can use
 MOD=01 REG=001 and R/M=111 = 0100 1111 = 4FH
- and the last byte is 10H
 result: 88 4F 10
 Note: MOV [BX+10H] ,CX = 89 4F 10
- since 10H can also be encoded as a 16-bit displacement, we can use
 MOD=10 REG=001 and R/M=111 = 1000 1111 = 8FH
- and the last bytes are 00 10
 result: 88 8F 00 10

MOV = Move
 Register/Memory to/from Register
 Immediate to Register/Memory

76543210
 100010 dw
 1100011 w

76543210
 mod reg r/m
 mod 000 r/m

Thankyou

