



# 8086 Addressing Modes

Dept. of Computer Science and Engineering  
BRAC University

**CSE 341 Team**

# Lecture References:

---

## ? **Book:**

- ? *Microprocessors and Interfacing: Programming and Hardware,*  
Chapter # 2, **Author:** Douglas V. Hall
- ? *The 8086/8088 Family: Design, Programming, And Interfacing,*  
Chapter # 2, **Author:** John Uffenbeck.

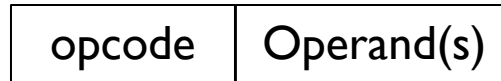
# Addressing Mode and Categories

---

- ? The different ways in which a microprocessor can access data are referred to as its addressing modes.
- ? Addressing modes of 8086 Microprocessor are categorized as:
  - ? *Addressing Data*
  - ? *Addressing Program codes in memory*
  - ? *Addressing Stack in memory*
  - ? *Addressing I/O*
  - ? *Implied addressing*

# Things to know...

? Instruction format



? Instructions can have 1, 2 or no operands

? **INC AX** ; 1 operand

? **ADD CX, DX** ; 2 operands  $\longrightarrow$   $CX = CX + DX$

? **HLT** ; no operand

Destination  
source

? Instruction cannot have:

? **SUB [DI], [1234h]** ; memory locations as both operands

? **MOV 1234, AX** ; immediate data as destination operand

# 1. Addressing Data

---

- I. Immediate addressing
- II. Direct addressing
- III. Register [direct] addressing
- IV. Register indirect addressing
- V. Base-plus-index addressing
- VI. Register relative addressing
- VII. Base-relative-plus-index addressing

# 1. Addressing Data

---

## I. Immediate addressing

? Data is immediately given in the instruction

**MOV BL, 11h**

## II. Direct addressing

? Data address is directly given in the instruction

**MOV BX, [437AH]**

# 1. Addressing Data

---

## III. Register [direct] addressing

- ? Data is in a register (here BX register contains the data)

**MOV AX, BX**

**MOV AL, BX**

## IV. Register [indirect] addressing

- ? Register supplies the address of the required data

**MOV CX, [BX]**



# 1. Addressing Data

---

## v. **Base-plus-index addressing**

- ? Base register is either BX or BP
- ? Index register is either DI or SI

**MOV DX, [BX+DI]**

## v. **Register relative addressing**

- ? Register can be a base (BX, BP) or an index register (DI, SI)
- ? Mainly suitable to address array data

**MOV AX, [BX+1000]**





# 1. Addressing Data

---

## VII. **Base-relative-plus-index addressing**

? Suitable for array addressing

**MOV AX, [BX+DI+10]**

## 2. Addressing Program Codes in Memory

---

? Used with JMP and CALL instructions

? 3 distinct forms:

- I. Direct
- II. Indirect
- III. Relative



## 2. Addressing Program Codes in Memory

---

? Address is directly given in the instruction

**JMP** <sup>IP</sup>1000: <sup>CS</sup>0000

**JMP** doagain ; doagain is a **label** in code

**CALL** 1000:0000

**CALL** doagain ; doagain is a **procedure** in code

? Often known as *far* jump or *far* call

## 2. Addressing Program Codes in Memory

---

? Address can be obtained from

? **a)** any GP registers (AX,BX,CX,DX,SP,BP,DI,SI)

JMP AX  
*IP = AX ; then CS :  
IP*

? **b)** any relative registers ([BP],[BX],[DI],[SI])

JMP [BX]  
*IP = what is inside the physical address of DS : BX ; then CS  
: IP*

? **c)** any relative register with displacement

JMP [BX + 100h]  
*IP = what is inside the physical address of DS : BX + 100h ; then CS  
: IP*

### 3. Addressing Stack in Memory

---

- **PUSH** and **POP** instructions are used to move data to and from stack (in particular from stack segment).

PUSH AX

POP CX

- **CALL** also uses the stack to hold the return address for procedure.

CALL SUM ; SUM is a procedure name

## 4. Addressing Input and Output Port

---

? IN and OUT instructions are used to address I/O ports

? Could be *direct addressing*

**IN AL, 05h** ; Here 05h is a input port number

? or *indirect addressing*

**OUT DX, AL** ; DX contains the address of I/O port

? Only DX register can be used to point a I/O port

## 5. Implied Addressing

---

- ? No explicit address is given with the instruction
- ? implied within the instruction itself
- ? Examples:

**CLC** ; clear carry flag

**HLT** ; halts the program

**RET** ; return to DOS



# 8086 Machine Codes

Dept. of Computer Science and Engineering  
BRAC University

**CSE 341 Team**

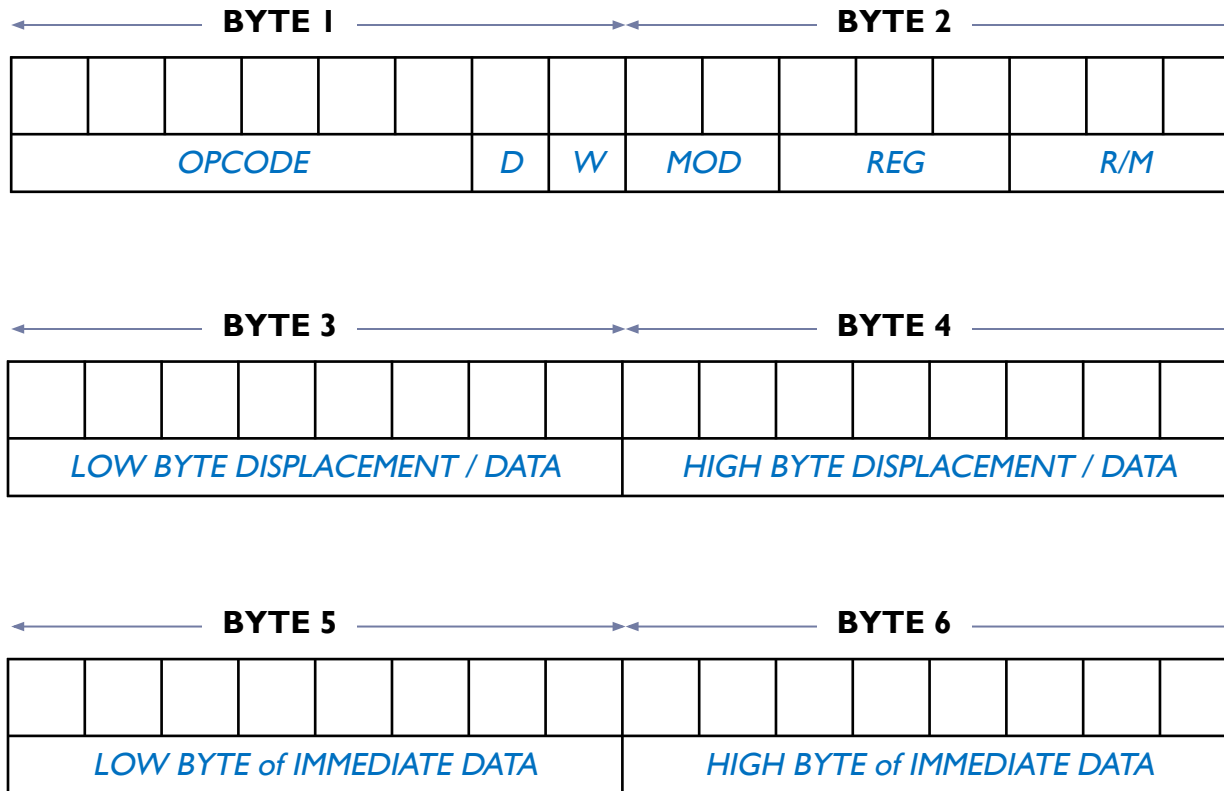


# Instruction template

- ? For 8085: Just look up the hex code for each instruction.
- ? For 8086 it is not simple.
- ? E.g 32 ways to specify the source in **MOV CX, source**.
- ? **MOV CX, source**
  - a 16-bit register (8 in number)*
  - a memory location (24 possible memory addressing modes)*
- ? Each of these 32 instructions require different binary code.
- ? Impractical to list them all in a table.
- ? Instruction templates help code the instruction properly.



# Instruction template (6 bytes)



*An instruction after conversion can have 1 to 6 bytes long of machine code*

# Constructing Machine Codes for 8086

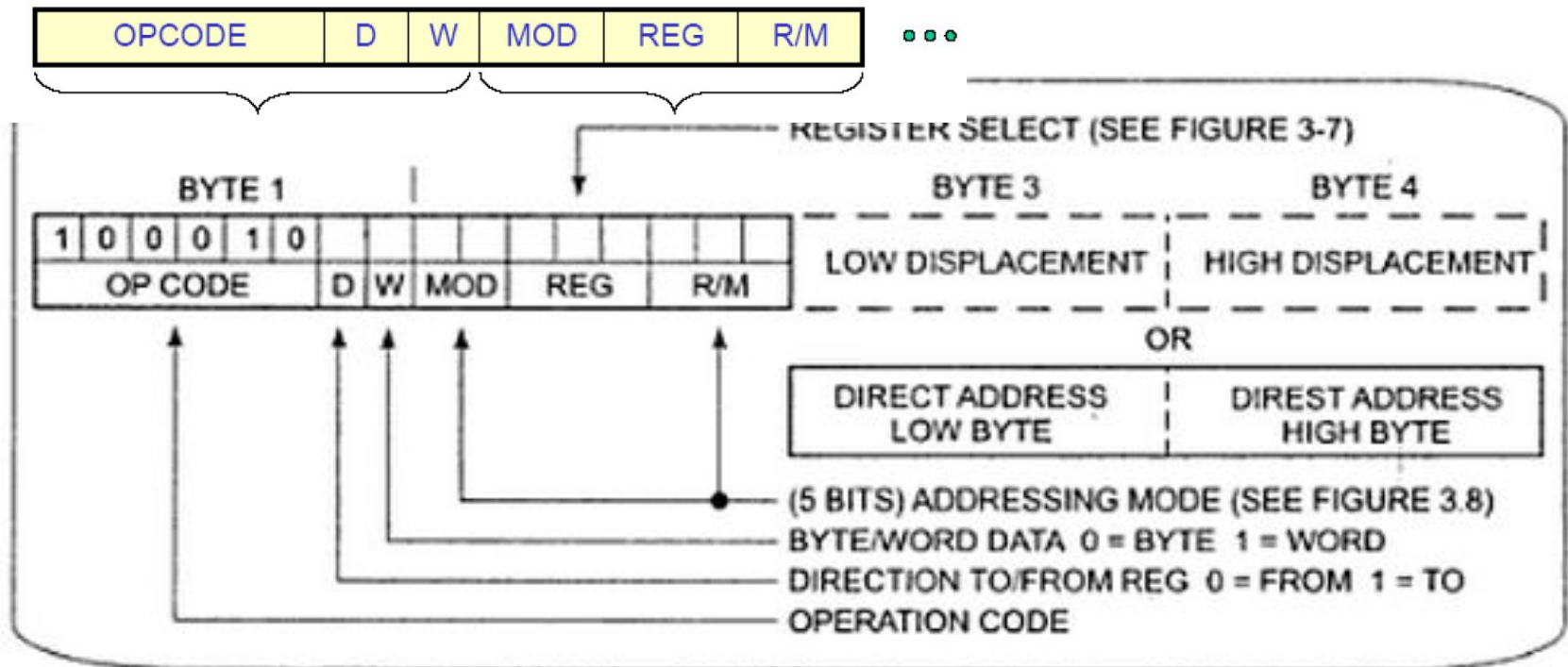
---

- ? Each instruction in 8086 is associated with the binary code.
- ? You need to locate the codes appropriately.
- ? Most of the time this work will be done by assembler
- ? The things needed to keep in mind is:
  - ? Instruction templates and coding formats
  - ? MOD and R/M Bit patterns for particular instruction

# MOV Instruction Coding

? MOV data from a register to a register/to a memory location  
or from a memory location to a register.

**(Operation Code of MOV: 100010)**



# MOD and R/M Field

- ? 2-bit Mode (MOD) and 3-bit Register/Memory (R/M) fields specify the other operand.
- ? Also specify the addressing mode.

RM \ MOD	MOD			
	00	01	10	11
				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	[BP] + [DI] + d16	BL    BX
100	[SI]	[SI] + d8	[SI] + d16	AH    SP
101	[DI]	[DI] + d8	[DI] + d16	CH    BP
110	d16 (direct address)	[BP] + d8	[BP] + d16	DH    SI
111	[BX]	[BX] + d8	[BX] + d16	BH    DI

# MOD and R/M Field

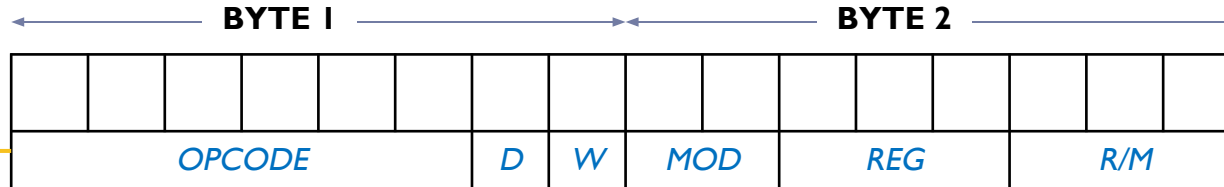
- ? If the other operand in the instruction is also one of the eight register then put in 11 for MOD bits in the instruction code.
- ? If the other operand is memory location, there are 24 ways of specifying how the execution unit should compute the effective address of the operand in the main memory.
- ? If the effective address specified in the instruction contains displacement less than 256 along with the reference to the contents of the register then put in 01 as the MOD bits.
- ? If the expression for the effective address contains a displacement which is too large to fit in 8 bits then out in 10 in MOD bits.

# REG Field

? REG field is used to identify the register of the one 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

# Instruction template



6 bits of  
MOV, ADD etc

**D** - direction

If **D=0**, then direction is from a register (source)  
If **D=1**, then direction is to a register (destination)

**W** - word

If **W=0**, then only a byte is being transferred (8 bits)  
If **W=1**, then a whole word is being transferred (16 bits)

- 34h here is an 8-bit displacement
- [BX+34h] is a memory/offset address

**MOV [BX + 34h], AL**

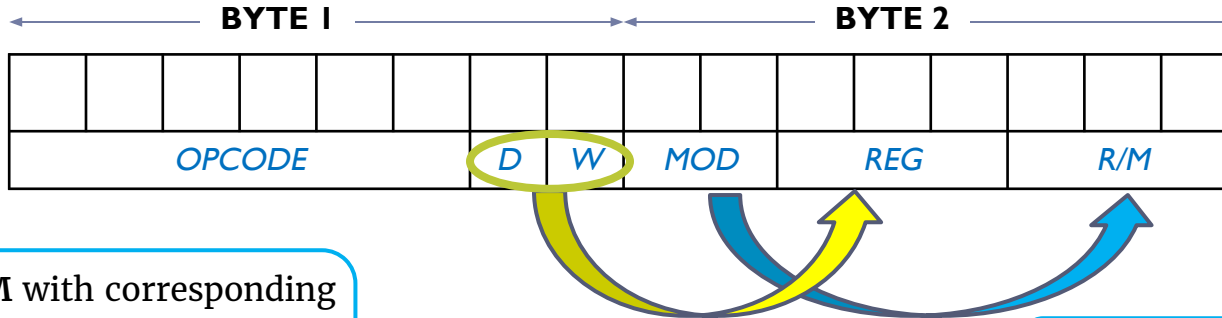
**MOV AX, 1234h**

- 1234h here is a 16-bit immediate data value

MODE	OPERAND NATURE
00	Memory with no displacement → MOV AX, [BX]
01	Memory with 8-bit displacement → MOV AX, [BX + 12h]
10	Memory with 16-bit displacement → MOV AX, [BX + 1234h]
11	Both are registers → MOV AX, BX



# Instruction template



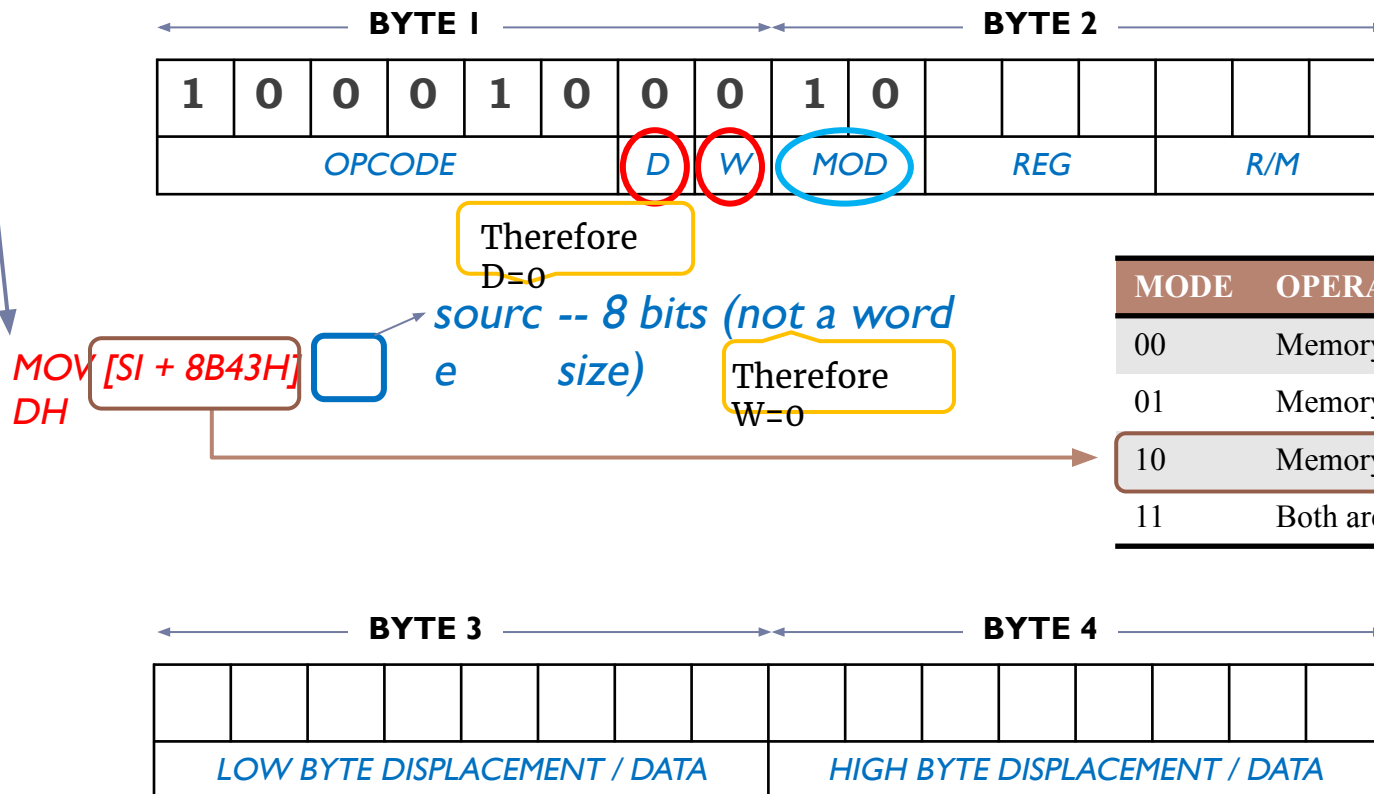
- Value for R/M with corresponding MOD value
- Value for REG with corresponding W value and the register considered in D

Check column that matches with MOD value

RM	MOD				
		00	01	10	11
					W = 0    W = 1
000		[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL    AX
001		[BXI] + [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	[BP] + [DI] + d16	BL    BX
100		[SI]	[SI] + d8	[SI] + d16	AH    SP
101		[DI]	[DI] + d8	[DI] + d16	CH    BP
110		d16 (direct address)	[BP] + d8	[BP] + d16	DH    SI
111		[BX]	[BX] + d8	[BX] + d16	BH    DI

# Example 1

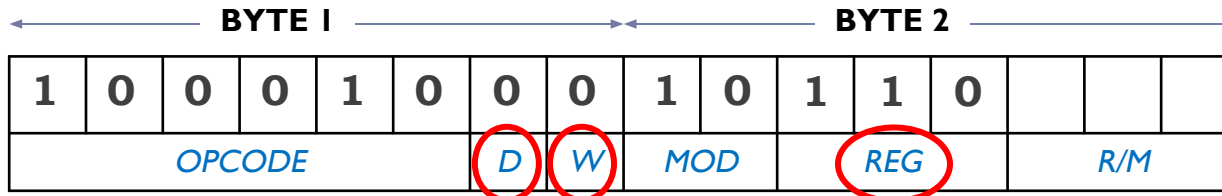
? **MOV 8B43H [SI], DH:** Copy a byte from DH to memory with 16 bit displacement given the opcode for MOV=100010



MODE	OPERAND NATURE
00	Memory with no displacement
01	Memory with 8-bit displacement
10	Memory with 16-bit displacement
11	Both are registers

# Example 1

? **MOV 8B43H [SI], DH:** Copy a byte from DH to memory with 16 bit displacement given the opcode for MOV=100010

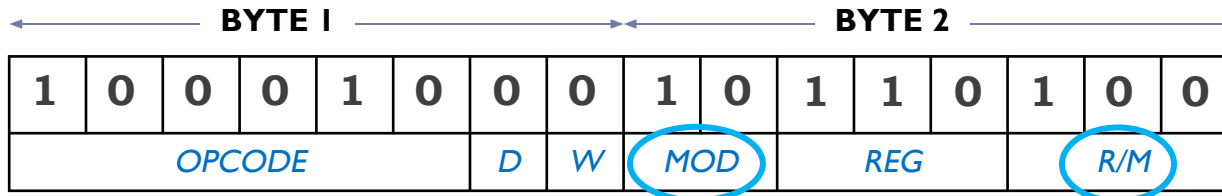


MOV [SI + 8B43H],  
DH

RM \ MOD	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BXI] + [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	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] + d16	AH	SP
101	[DI]	[DI] + d8	[DI] + d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] + d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

# Example 1

? **MOV 8B43H [SI], DH:** Copy a byte from DH to memory with 16 bit displacement given the opcode for MOV=100010

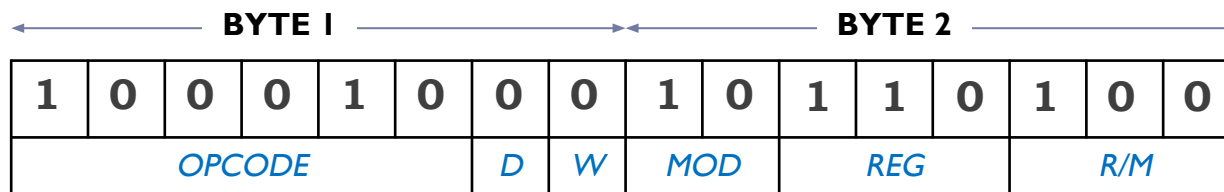


MOV [SI + 8B43H],  
DH

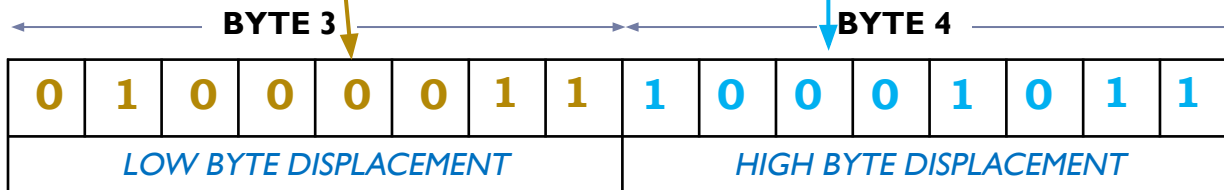
RM \ MOD	MOD					
	00	01	10	11	W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16		AL	AX
001	[BX] + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16		CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16		DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16		BL	BX
100	[SI]	[SI] + d8	[SI] + d16		AH	SP
101	[DI]	[DI] + d8	[DI] + d16		CH	BP
110	d16 (direct address)	[BP] + d8	[BP] + d16		DH	SI
111	[BX]	[BX] + d8	[BX] + d16		BH	DI

# Example 1

? **MOV 8B43H [SI], DH:** Copy a byte from DH to memory with 16 bit displacement given the opcode for MOV=100010



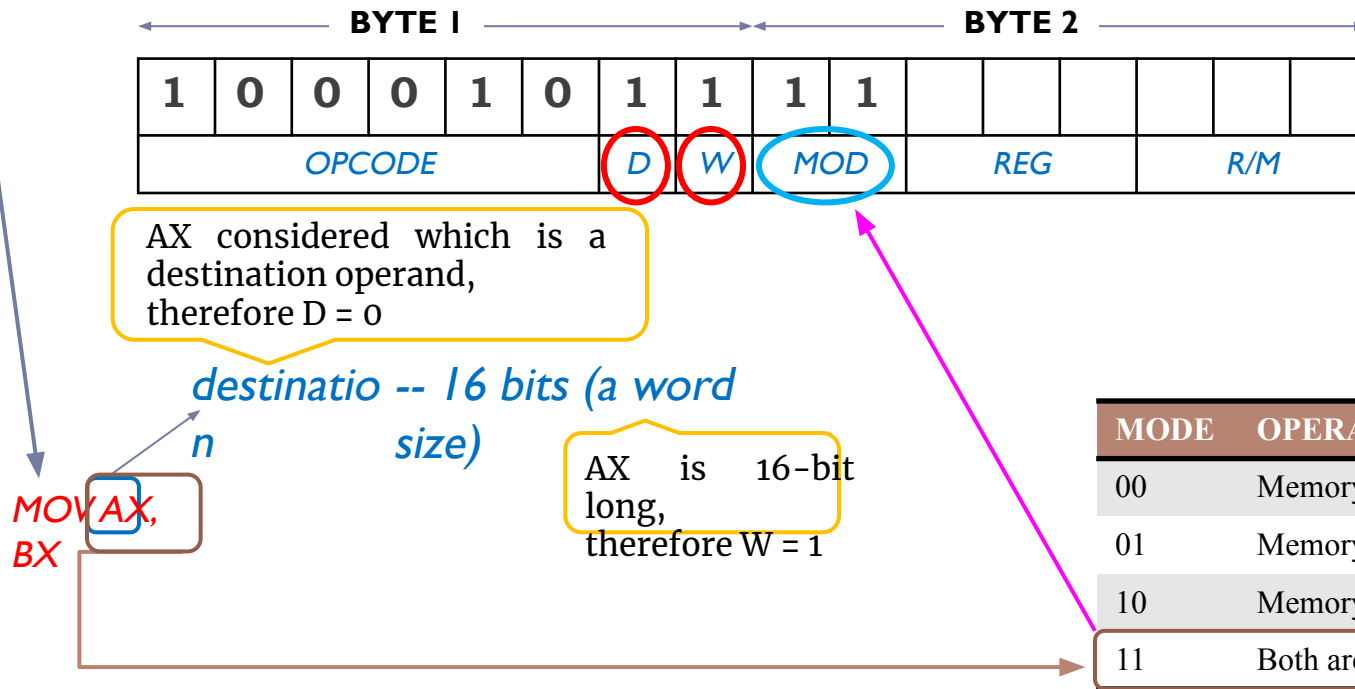
MOV [SI + 8B43H],  
DH



**Machine Code:** 1000 1000 1011 0100 0100 0011 1000 1011<sub>2</sub> or 88 B4 43 8B<sub>16</sub>

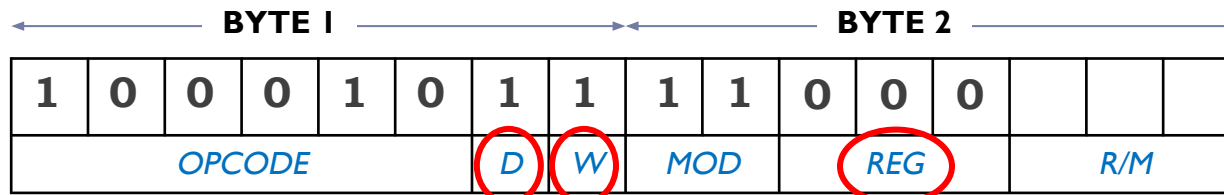
# Example 2

? **MOV AX, BX:** given the opcode for MOV=100010



## Example 2

? MOV AX, BX: given the opcode for MOV=100010

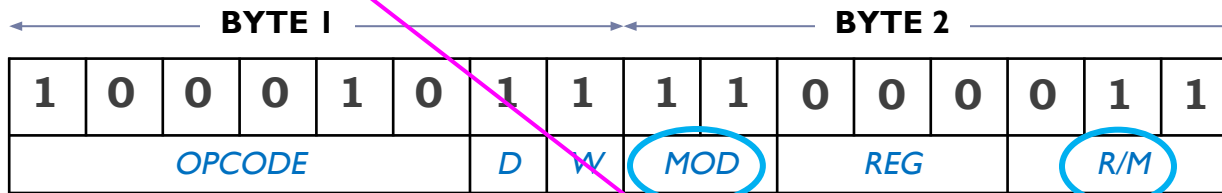


RM \ MOD	00	01	10	11	
				W = 0	W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL	AX
001	[BX] + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL	CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL	DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL	BX
100	[SI]	[SI] + d8	[SI] + d16	AH	SP
101	[DI]	[DI] + d8	[DI] + d16	CH	BP
110	d16 (direct address)	[BP] + d8	[BP] + d16	DH	SI
111	[BX]	[BX] + d8	[BX] + d16	BH	DI

## Example 2

Machine Code: *1000 1011 1100 0011*<sub>2</sub> or *8B C3*<sub>16</sub>

? MOV AX, BX: given the opcode for MOV=100010



RM \ MOD	MOD				11	
		00	01	10	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	[BP] + [DI] + d16	BL	BX
100		[SI]	[SI] + d8	[SI] + d16	AH	SP
101		[DI]	[DI] + d8	[DI] + d16	CH	BP
110		d16 (direct address)	[BP] + d8	[BP] + d16	DH	SI
111		[BX]	[BX] + d8	[BX] + d16	BH	DI



# QUIZ

Compute the machine code for the following using the table below and the opcode for MOV as 100010

a) MOV AX, 5E9Ch

b) MOV DH, [BP+SI+7Dh]

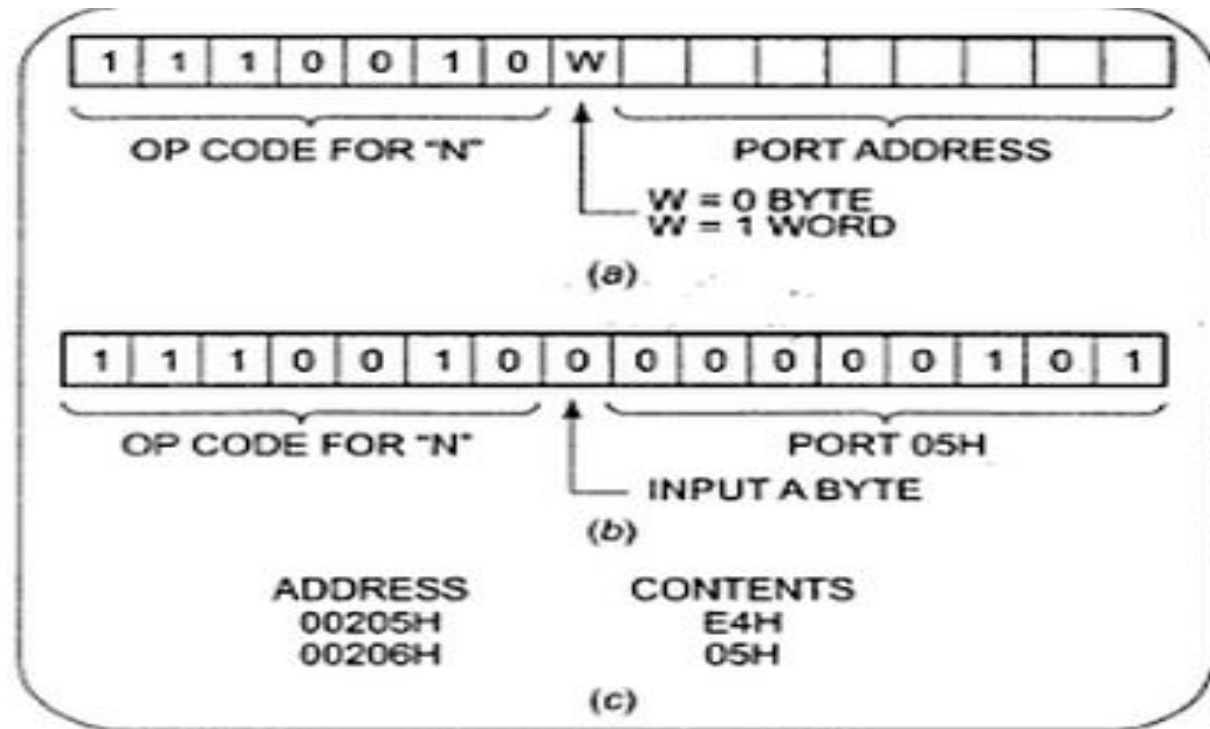
RM \ MOD	MOD			
	00	01	10	11
				W = 0    W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL    AX
001	[BXI] + [DI]	[BX] + [DI] + d8	[BX] + [DI] + d16	CL    CX
010	[BP] + [SI]	[BP] + [SI] + d8	[BP] + [SI] + d16	DL    DX
011	[BP] + [DI]	[BP] + [DI] + d8	[BP] + [DI] + d16	BL    BX
100	[SI]	[SI] + d8	[SI] + d16	AH    SP
101	[DI]	[DI] + d8	[DI] + d16	CH    BP
110	d16 (direct address)	[BP] + d8	[BP] + d16	DH    SI
111	[BX]	[BX] + d8	[BX] + d16	BH    DI

# Instruction Template

- ? The Intel literature shows two different formats for coding 8086 instructions.
- ? Instruction templates help you to code the instruction properly.

? **Example:**

IN AL, 05H



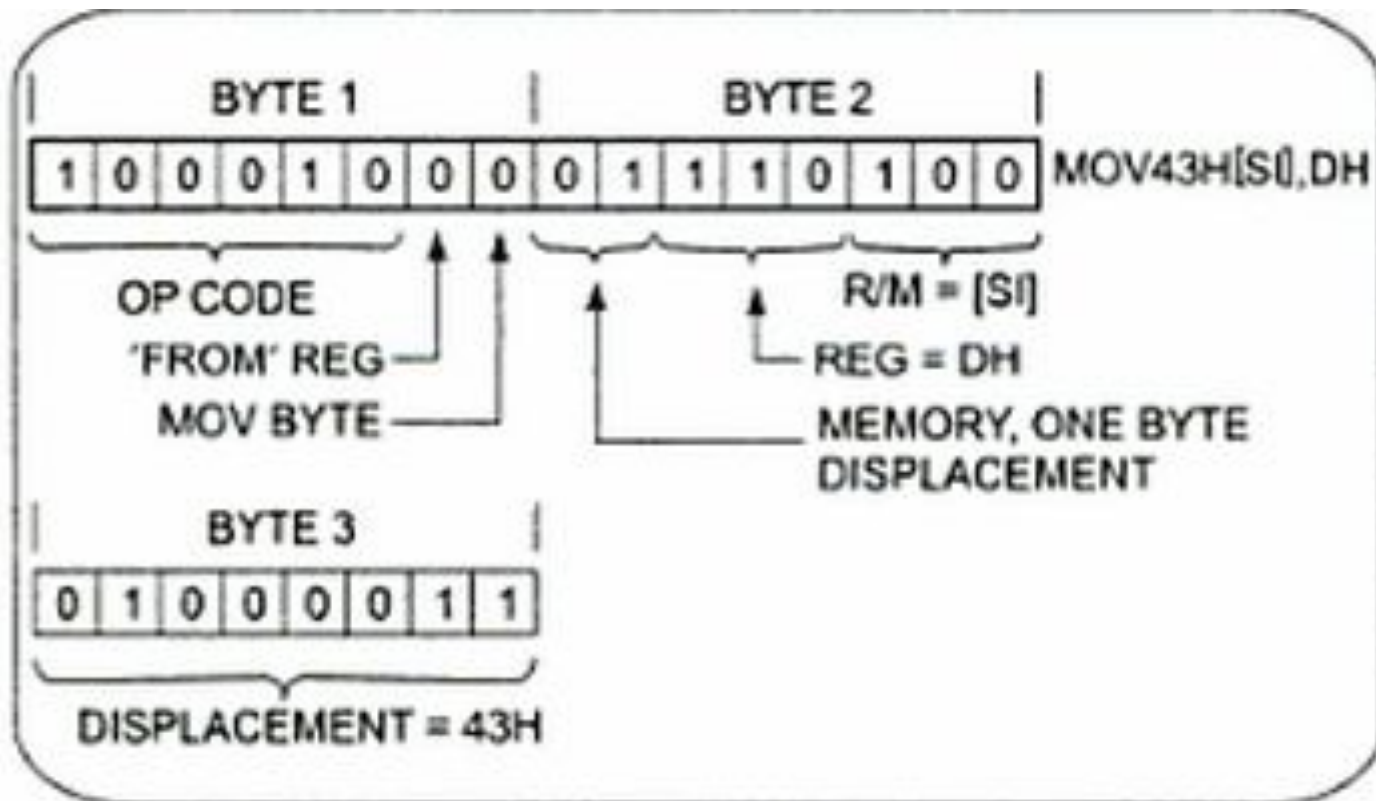
# Example

- 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

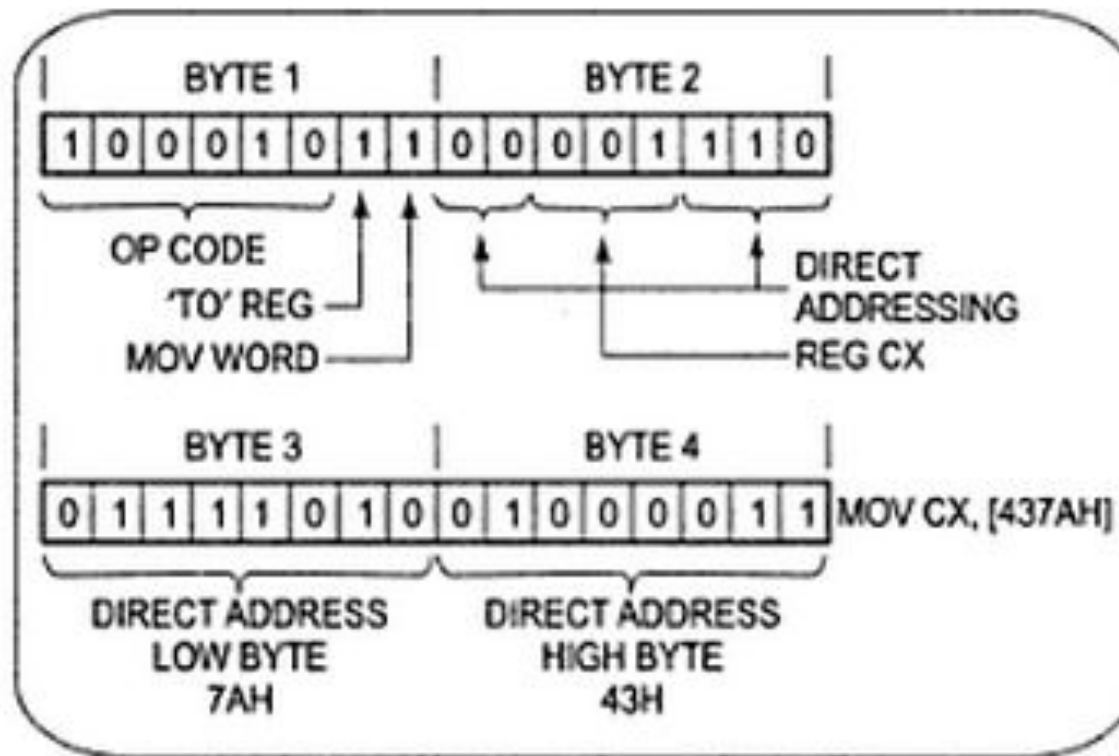
# Example

? **MOV 43H [SI], DH:** Copy a byte from DH register to memory location.



## Example 3

? **MOV CX, [437AH]:** Copy the contents of the two memory locations to the register CX.



# QUIZ

Compute the machine code for the following using the table below and the opcode for MOV as 100010

a) MOV AX, 5E9Ch

b) MOV DH, [BP+SI+7Dh]

RM \ MOD	MOD			
	00	01	10	11
				W = 0    W = 1
000	[BX] + [SI]	[BX] + [SI] + d8	[BX] + [SI] + d16	AL    AX
001	[BXI] + [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	[BP] + [DI] + d16	BL    BX
100	[SI]	[SI] + d8	[SI] + d16	AH    SP
101	[DI]	[DI] + d8	[DI] + d16	CH    BP
110	d16 (direct address)	[BP] + d8	[BP] + d16	DH    SI
111	[BX]	[BX] + d8	[BX] + d16	BH    DI

# Thank You !!!