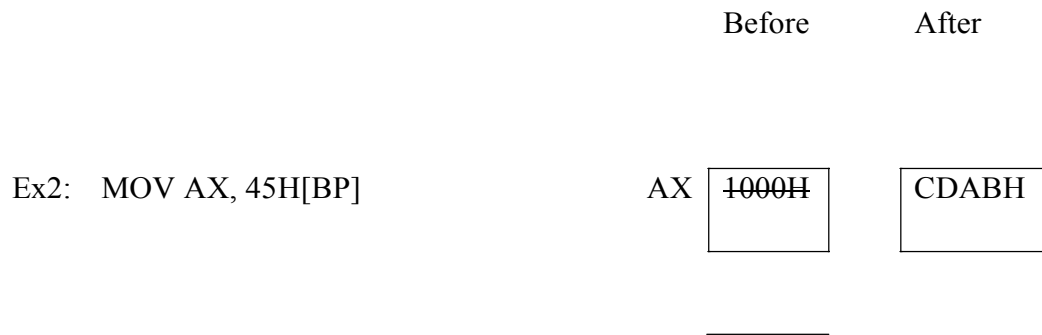


# 8086 ADDRESSING MODES - II

## 2.4.5 Based Addressing with displacement



45H is 8-bit displacement

BP 

3000H
-------

$3000 + 45 = 3045H$

SS:3045H 

ABH
-----

 LS byte

It is SS when BP is used

SS:3346H 

CDH
-----

 MS byte

Base register can only be BX or BP. This scheme provides 4 ways of addressing an operand in memory.

### 2.4.6 Indexed Addressing with displacement

Before      After

Ex1: MOV CL, 2345H[SI]

CL 

60H
-----

85H
-----

2345H is 16-bit displacement

SI 

6000H
-------

$6000 + 2345 = 8345H$

DS:8345H 

85H
-----

Before      After

Ex2: MOV DX, 37H[DI]

DX 

7000H
-------

B2A2H
-------

37H is 8-bit displacement

DI 

5000H
-------

$5000H + 37H = 5037H$

DS:5037H	A2H	LS byte
DS:5038H	B2H	MS byte

Index register can only be SI or DI. This scheme provides 4 ways of addressing an operand in memory.

### 2.4.7 Based Indexed Addressing

Before      After

Ex1: MOV CL, [SI][BX]

CL 

<del>40H</del>
----------------

67H
-----

SI 

2000H
-------

BX 

0300H
-------

$2000H + 0300H = 2300H$

DS:2300H 

67H
-----

Before      After

Ex2: MOV CX, [BP][DI]

CX 

<del>6000H</del>
------------------

6385H
-------

BP 3000H

DI 0020H

$$2000H + 0300H = 2300H$$

It is SS when BP is used

SS:3020H	85H	LS byte
SS:3021H	63H	MS byte

This scheme provides 4 ways of addressing an operand in memory. One register must be a Base register and the other must be an Index register.

For ex. MOV CX, [BX][BP] is an invalid instruction.

#### 2.4.6 Based Indexed Addressing with Displacement

	Before	After		
Ex1: MOV DL, 37H[BX+DI]	DL <table border="1"><tr><td>40H</td></tr></table>	40H	<table border="1"><tr><td>12H</td></tr></table>	12H
40H				
12H				
37H is 8-bit displacement	BX <table border="1"><tr><td>2000H</td></tr></table>	2000H		
2000H				
	DI <table border="1"><tr><td>0050H</td></tr></table>	0050H		
0050H				

$$2000H + 0050H + 37H = 2300H$$

DS:2087H 

12H
-----

Before      After

Ex2: MOV BX, 1234H[SI+BP]

BX 

3000H
-------

3665H
-------

SI 

4000H
-------

BP 

0020H
-------

$$4000H + 0020H + 1234 = 5254H$$

SS:5254H 

65H
-----

      LS byte

It is SS when BP is used

SS:5255H 

36H
-----

      MS byte

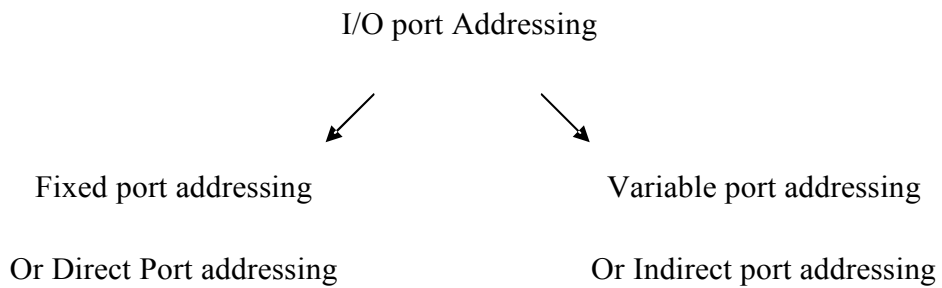
This scheme provides 8 ways of addressing an operand in memory.

### 2.4.7 Memory modes as derivatives of Based Indexed Addressing with Displacement

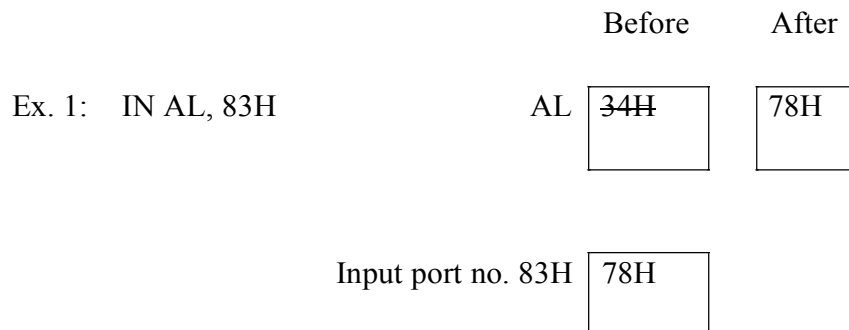
Instruction	Base Register	Index Register	Displacement	Addressing mode
MOV BX, DS:5634H	No	No	Yes	Direct Addressing
MOV CL, [SI]	No	Yes	No	Register Indirect
MOV DX, [BX]	Yes	No	No	
MOV DH, 2345H[BX]	Yes	No	Yes	Based Addressing with

				Displacement
MOV DX, 35H[DI]	No	Yes	Yes	Indexed Addressing with displacement
MOV CL, 37H[SI+BX]	Yes	Yes	No	Based Indexed Addressing
MOV DL, 37H[BX+DI]	Yes	Yes	Yes	Based Indexed Addressing with displacement

### 2.4.8 I/O port Addressing



### Fixed Port Addressing



	Before	After
Ex. 2: IN AX, 83H	AX 5634H	F278H

Input port no. 83H	78H
Input port no. 84H	F2H

	Before	After
Ex. 3: OUT 83H, AL	AL 50H	

Output port no. 83H	65H	50H
---------------------	-----	-----

	Before	After
Ex. 4: OUT 83H, AX	AX 6050H	

Output port no. 83H	65H	50H
Output port no. 84H	40H	60H

IN and OUT instructions are allowed to use only AL or AX registers. Port address in the range 00 to FFH is provided in the instruction directly.

### 2.4.9. Variable Port Addressing

I/O port address is provided in DX register. Port address ranges from 0000 to FFFFH. Data transfer with AL or AX only.

Ex. 1: IN AL, DX

	Before	After
AL	<input type="text" value="30H"/>	<input type="text" value="60H"/>

DX

Input port no. 1234H

Ex. 2: IN AX, DX

	Before	After
AX	<input type="text" value="3040H"/>	<input type="text" value="7060H"/>

DX

Input port no. 4000H

Input port no. 4001H

Ex. 3: OUT DX, AL

	Before	After
AL	<input type="text" value="65H"/>	

DX

Output port no. 5000H



	Before	After
Ex. 4: OUT DX, AX	AX 4567H	
	DX 5000H	
Output port no. 5000H	<del>25H</del>	67H
Output port no. 5001H	<del>36H</del>	45H

Source : <http://elearningatria.files.wordpress.com/2013/10/cse-iv-microprocessors-10cs45-notes.pdf>