# ADDRESSING MODES

Dr. Manju Khurana
Assistant Professor, CSED
TIET, Patiala
manju.khurana@thapar.edu

# Addressing Modes

- Every instruction of a program has to operate on a data.
- The different ways in which a source operand is denoted in an instruction are known as addressing modes.

1. Register Addressing
2. Immediate Addressing

**Group I : Addressing modes for register and immediate data**

3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Indexed Addressing
8. Based Indexed with Displacement

**Group II : Addressing modes for memory data**

# Addressing Modes

## Group I : Addressing modes for register and immediate data

1. **Register Addressing**
2. **Immediate Addressing**
3. **Direct Addressing**
4. **Register Indirect Addressing**
5. **Based Addressing**
6. **Indexed Addressing**
7. **Based Indexed Addressing**
8. **Based Indexed with Displacement**

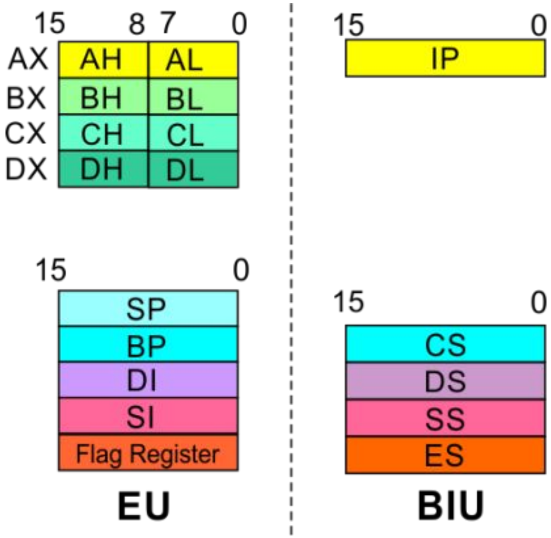
**The instruction will specify the name of the register which holds the data to be operated by the instruction.**

**Example:**

**MOV CL, DH**

**The content of 8-bit register DH is moved to another 8-bit register CL**

**(CL) ← (DH)**



# Addressing Modes

## Group I : Addressing modes for register and immediate data

1. Register Addressing
2. **Immediate Addressing**
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Indexed Addressing
8. Based Indexed with Displacement

In immediate addressing mode, **an 8-bit or 16-bit data** is specified as part of the instruction

**Example:**

```
MOV DL, 08H
```

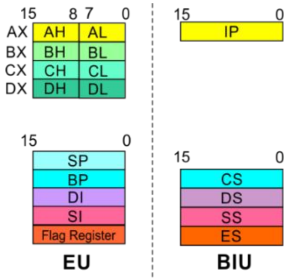
The 8-bit data ($08_H$) given in the instruction is moved to DL

$(DL) \leftarrow 08_H$

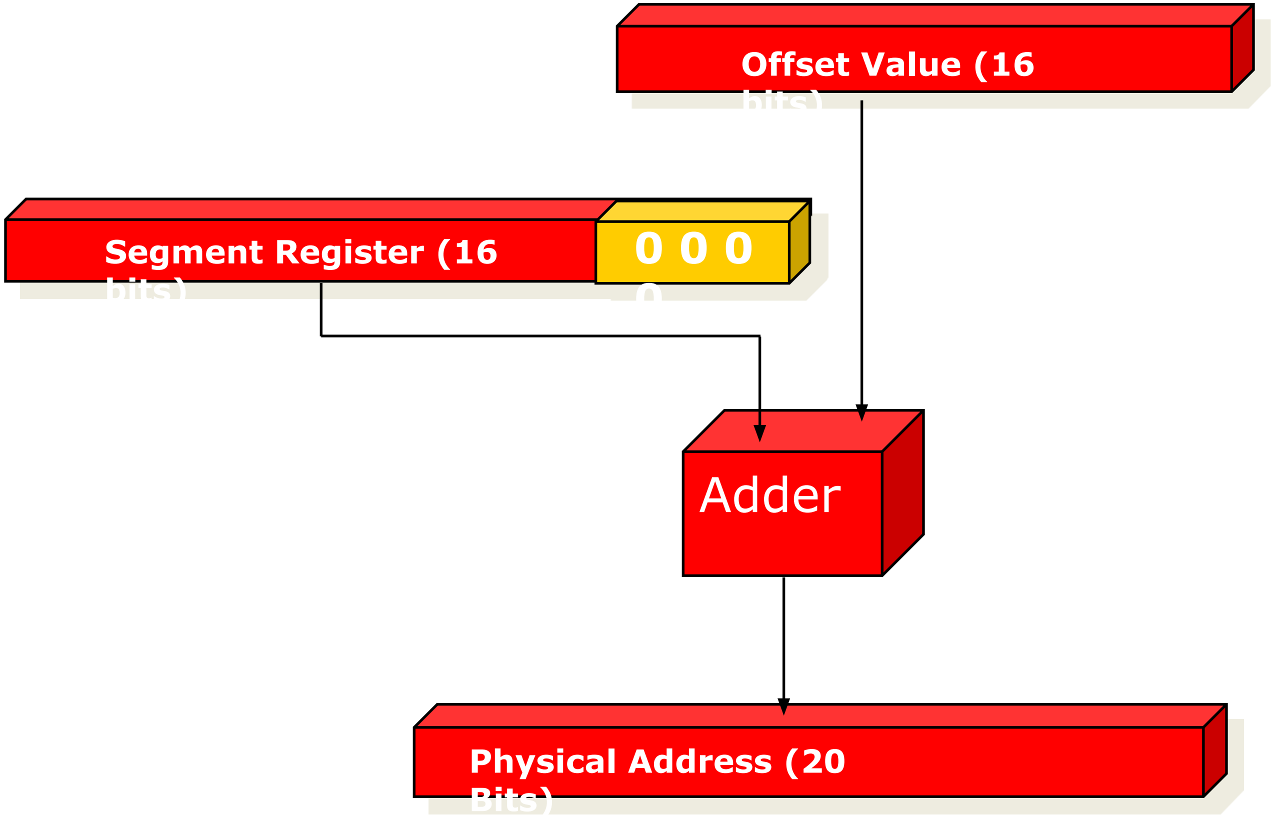
```
MOV AX, 0A9FH
```

The 16-bit data ($0A9F_H$) given in the instruction is moved to AX register

$(AX) \leftarrow 0A9F_H$



# Addressing Modes : Memory Access



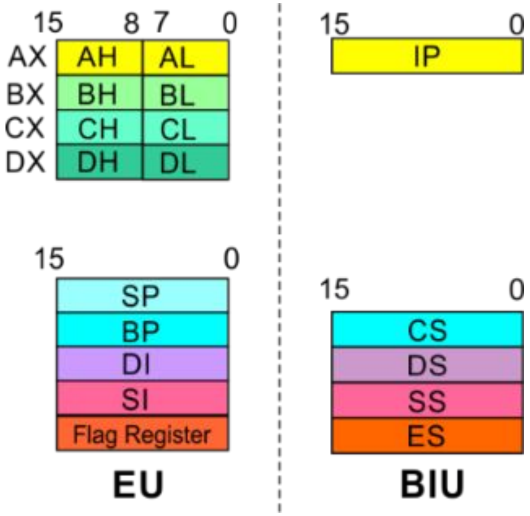
Offset Value (16 bits)

Segment Register (16 bits)

0 0 0 0

Adder

Physical Address (20 Bits)



# Addressing Modes : Memory Access

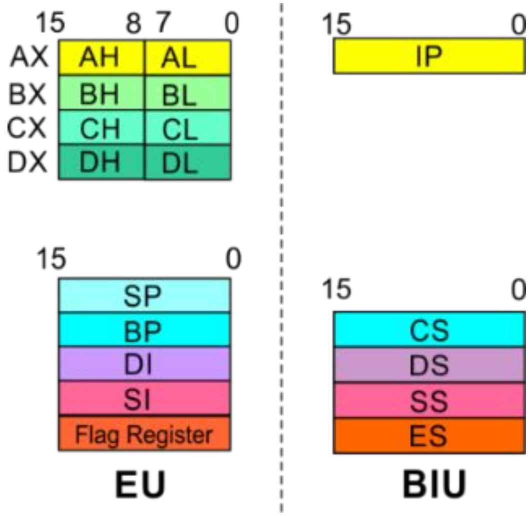
- 20 Address lines ⇒ 8086 can address up to $2^{20}$ = 1M bytes of memory

- However, the largest register is only 16 bits

- Physical Address will have to be calculated
  Physical Address : Actual address of a byte in memory. i.e. the value which goes out onto the address bus.

- Memory Address represented in the form – Seg : Offset (Eg - 89AB:F012)

- Each time the processor wants to access memory, it takes the contents of a segment register, shifts it one hexadecimal place to the left (same as multiplying by $16_{10}$), then add the required offset to form the 20- bit address



16 bytes of contiguous memory

89AB : F012 → 89AB → 89AB0 (Paragraph to byte → 89AB x 10 = 89AB0)
F012 → 0F012 (Offset is already in byte unit)
+ -------
98AC2 (The absolute address)

# Addressing Modes

**Group II : Addressing modes for memory data**

1. Register Addressing
2. Immediate Addressing
3. **Direct Addressing**
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Indexed Addressing
8. Based Indexed with Displacement

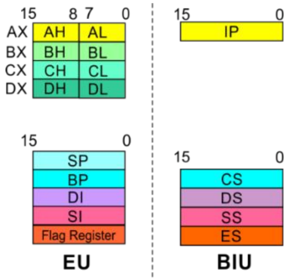
In direct addressing mode the operand’s offset is given in the instruction as an 8-bit or 16-bit displacement element.

**Examples are:**

```
MOV   BX, [1354H]
MOV   BL,  [0400H]
```

The square brackets around the $1354_H$ denotes the contents of the memory location. When executed, this instruction will copy the contents of the memory location into BX register.

This addressing mode is called direct because the displacement of the operand from the segment base is specified directly in the instruction.



# Addressing Modes

Group II : Addressing modes for memory data

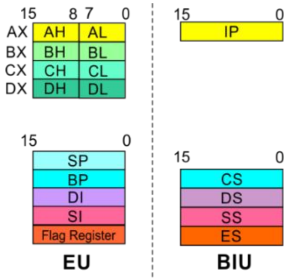
1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Indexed Addressing
8. Based Indexed with Displacement

**In Register indirect addressing, operand's offset is placed in any one of the registers BX, BP, SI or DI as specified in the instruction.**

**Example:**

**MOV CX, [BX]**

**ADD AL, [SI]**



# Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. **Based Addressing**
6. Indexed Addressing
7. Based Indexed Addressing
8. Based Indexed with Displacement

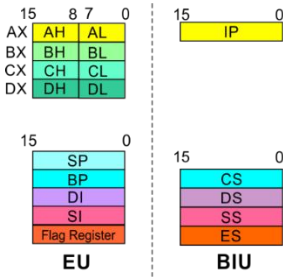
**In Based Addressing, operand's offset is the sum of an 8-bit or 16-bit displacement and the contents of the base register BX or BP. BX is used as a base register for data segment, and BP is used as a base register for stack segment.**

**Offset (Effective address)= [BX + 8-bit or 16-bit displacement]**

**Examples are:**

**MOV AL, [BX + 05H]**

**MOV AL, [BX + 1346H]**



# Addressing Modes

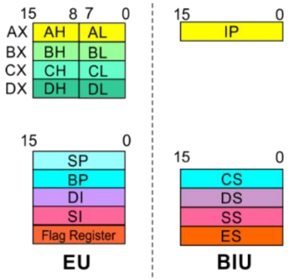
Group II : Addressing modes for memory data

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. **Indexed Addressing**
7. Based Indexed Addressing
8. Based Indexed with Displacement

The operand's offset is the sum of the content of an index register SI or DI and an 8-bit or 16-bit displacement.

Offset (Effective address)= [SI or DI + 8-bit or 16-bit displacement]

MOV AX, [SI + 05]
MOV AX, [SI + 1528H]



# Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. **Based Indexed Addressing**
8. Based Indexed with Displacement

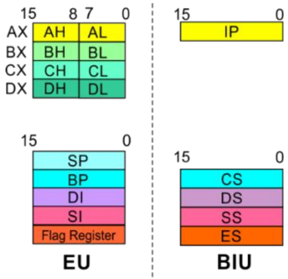
In Based Index Addressing, the operand's offset is the sum of the content of a base register BX or BP and an index register SI or DI.

Offset (Effective address)= [BX or BP]+ [SI or DI]

Examples are:

ADD AX, [BX + SI]

MOV CX, [BX + SI]



# Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Indexed Addressing
8. Based Indexed with Displacement

In this mode of addressing the operand's offset is given by

Offset (Effective address)= [BX or BP]+ [SI or DI] + 8-bit or 16-bit displacement

Examples are:

MOV AX, [BX + SI + 05]

MOV AX, [BX + SI + 1235H]

Thank you!