University of Diyala
College of engineering
Department of computer Engineering
Second class

microprocessor Programming

Lecture 4

8086 Addressing Modes

Presented by
Lecturer :Abdullah Thair Abdalsatir
Department of computer Engineering
University of Diyala

Lecture 4
8086 microprocessor

❖Addressing Modes :-

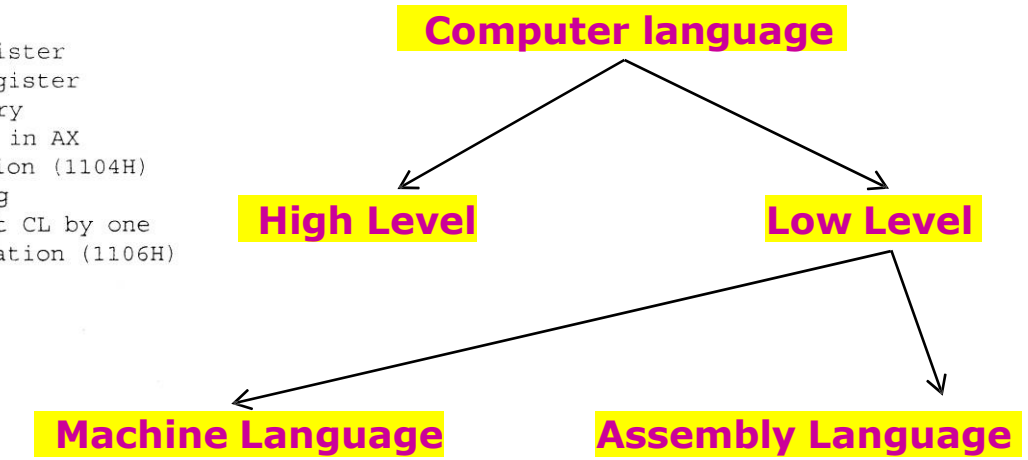
- Register Addressing
- Immediate Addressing
- Direct Addressing
- Register Indirect Addressing
- Based Addressing
- Indexed Addressing
- Based Index Addressing
- String Addressing
- Direct I/O port Addressing
- Indirect I/O port Addressing
- Relative Addressing
- Implied Addressing

Introduction

```
;PROGRAM TO ADD TWO 16-BIT DATA (METHOD-1)

DATA SEGMENT              ;Assembler directive

        ORG 1104H         ;Assembler directive
        SUM DW 0          ;Assembler directive
        CARRY DB 0        ;Assembler directive

DATA ENDS                 ;Assembler directive

CODE SEGMENT              ;Assembler directive

        ASSUME CS:CODE    ;Assembler directive
        ASSUME DS:DATA    ;Assembler directive
        ORG 1000H         ;Assembler directive

        MOV AX,205AH      ;Load the first data in AX register
        MOV BX,40EDH      ;Load the second data in BX register
        MOV CL,00H        ;Clear the CL register for carry
        ADD AX,BX         ;Add the two data, sum will be in AX
        MOV SUM,AX        ;Store the sum in memory location (1104H)
        JNC AHEAD         ;Check the status of carry flag
        INC CL            ;If carry flag is set,increment CL by one
AHEAD:  MOV CARRY,CL      ;Store the carry in memory location (1106H)
        HLT

CODE ENDS                 ;Assembler directive
END                       ;Assembler directive
```

Program
A set of instructions written to solve a problem.

Instruction
Directions which a microprocessor follows to execute a task or part of a task.



- **Binary bits**

- **English Alphabets**
- **'Mnemonics'**
- **Assembler**
  Mnemonics → Machine Language

Introduction

Programis a set of instructions written to solve a problem. Instructions are the directions which a microprocessor follows to execute a task or part of a task. Broadly, computer language can be divided into two parts as high-level language and low level language. Low level language are machine specific. Low level language can be further divided into machine language and assembly language.

Machine language is the only language which a machine can understand. Instructions in this language are written in binary bits as a specific bit pattern. The computer interprets this bit pattern as an instruction to perform a particular task. The entire program is a sequence of binary numbers. This is a machine-friendly language but not user friendly. Debugging is another problem associated with machine language.

To overcome these problems, programmers develop another way in which instructions are written in English alphabets. This new language is known as Assembly language. The instructions in this language are termed mnemonics. As microprocessor can only understand the machine language so mnemonics are translated into machine language either manually or by a program known a3s7 assembler.

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 Index Addressing 8. String Addressing	Group II: Addressing modes for memory data
9. Direct I/O port Addressing 10. Indirect I/O port Addressing	Group III: Addressing modes for I/O ports
11. Relative Addressing	Group IV: Relative Addressing mode
12. Implied Addressing	Group V: Implied Addressing mode

Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing

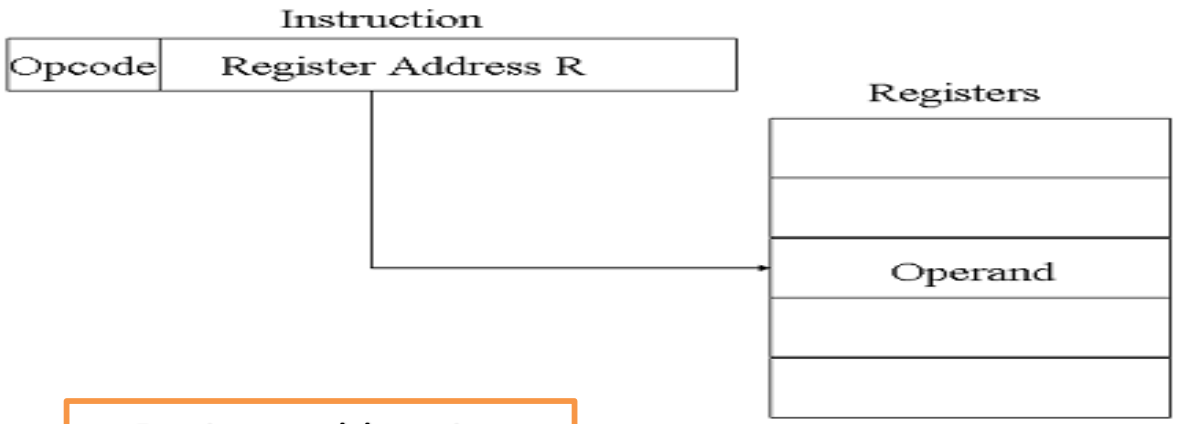
The instruction will specify the name of the register which holds the data to be operated by the instruction.

Example:

MOV CL, DH

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

(CL) ← (DH)



Register addressing

Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing

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

Example:

MOV DL, 08H

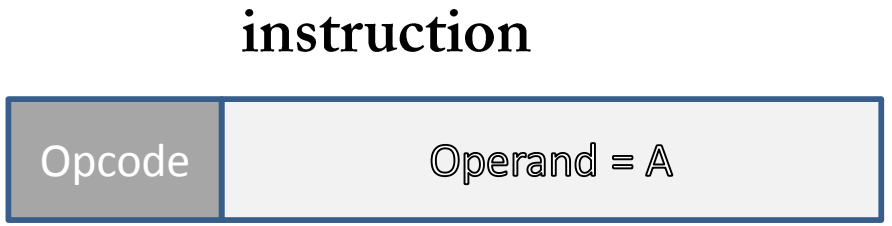
The 8-bit data (08_H) given in the instruction is moved to DL

(DL) $\leftarrow 08_H$

MOV AX, 0A9FH

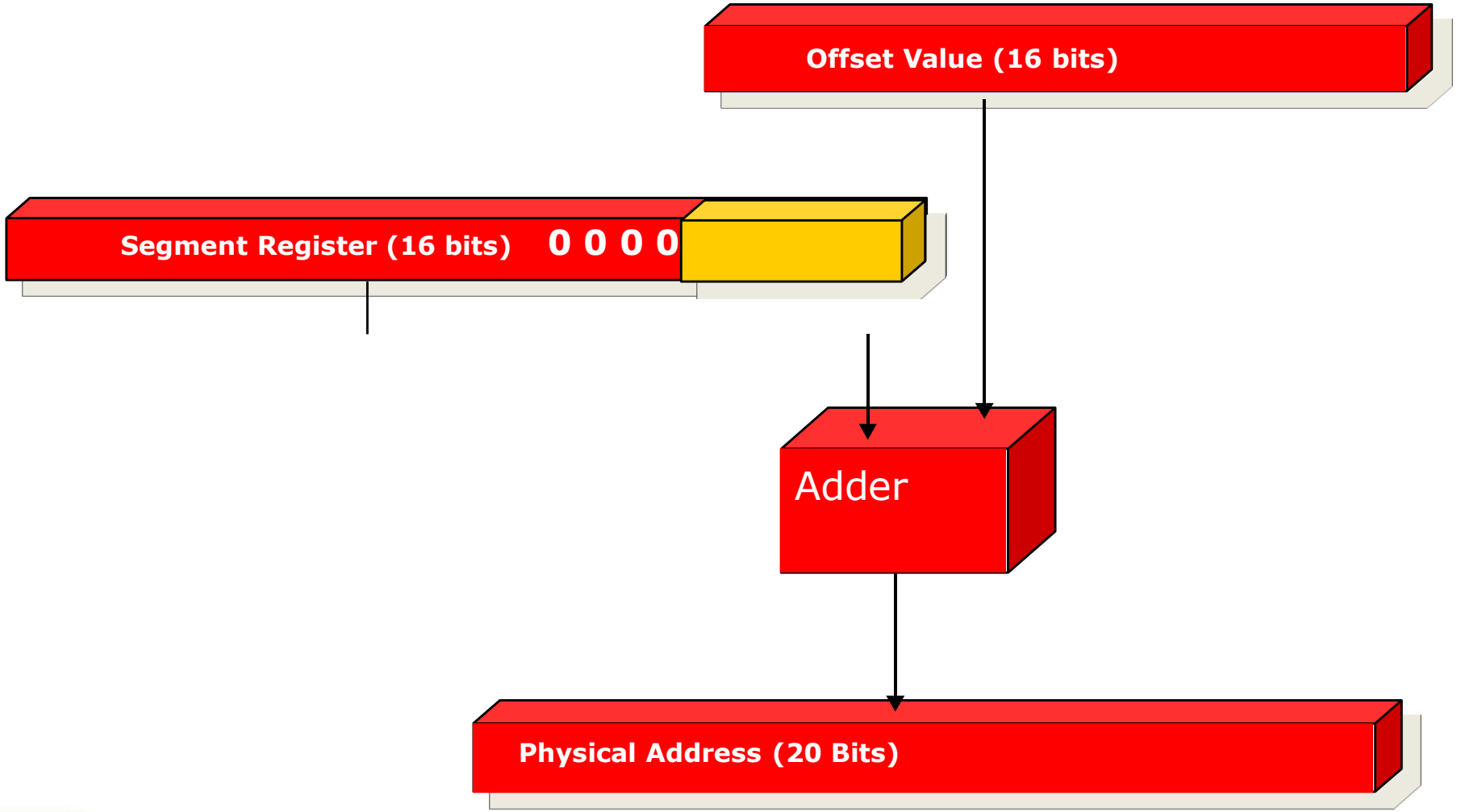
The 16-bit data ($0A9F_H$) given in the instruction is moved to AX register

(AX) $\leftarrow 0A9F_H$



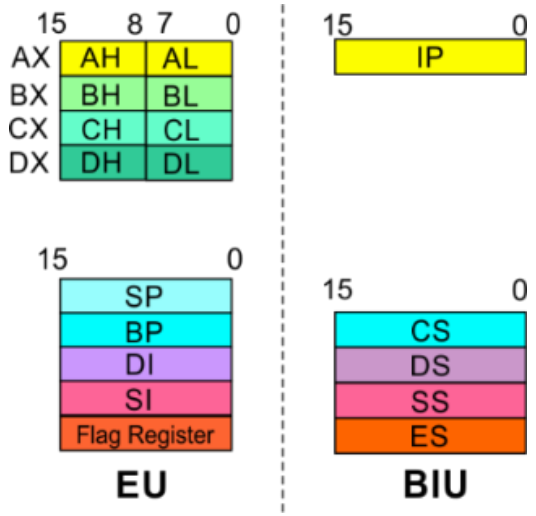
Immediate Addressing

Addressing Modes: Memory Access



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

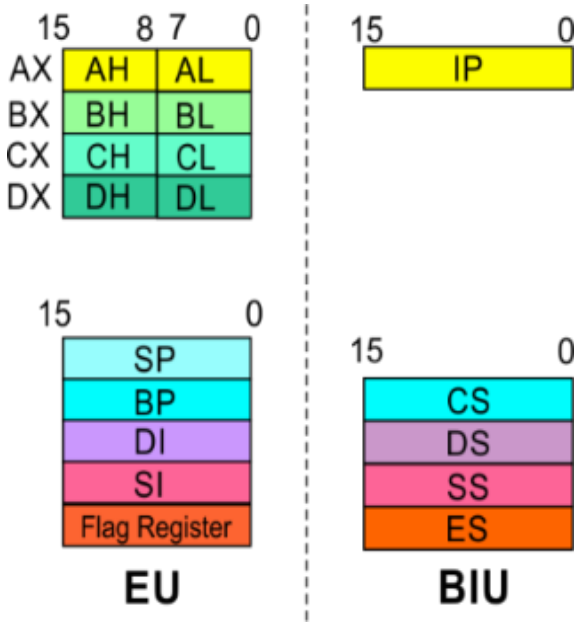
89AB : F012 → 89AB → 89AB0 (Paragraph to byte → 89AB x 10 = 89AB0) F012 →
0F012 (Offset is already in byte unit)
+ -------
98AC2 (The absolute address)

Addressing Modes : Memory Access

- **To access memory we use these four registers: BX, SI, DI, BP**
- **Combining these registers inside [ ] symbols, we can get different memory locations (Effective Address, EA)**
- **Supported combinations:**

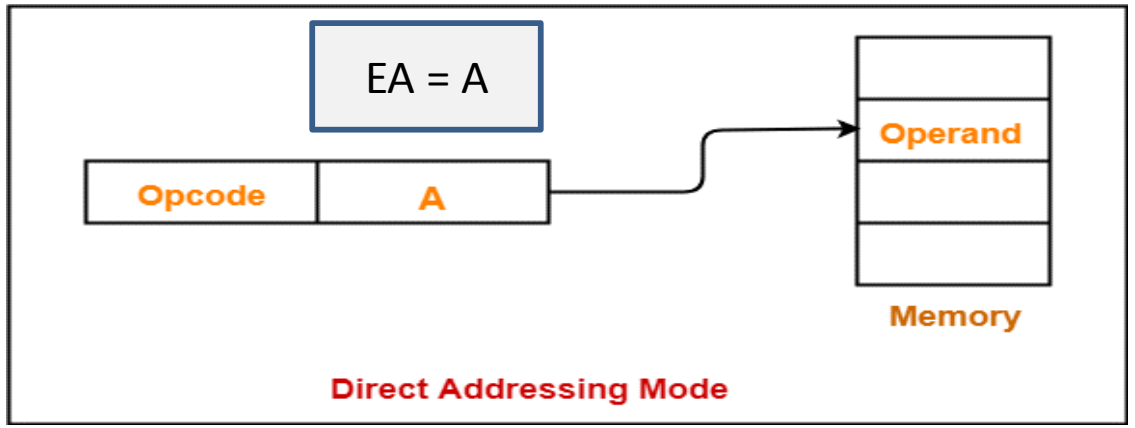
[BX + SI] [BX + DI] [BP + SI] [BP + DI]	[SI] [DI] d16 (variable offset only) [BX]	[BX + SI + d8] [BX + DI + d8] [BP + SI + d8] [BP + DI + d8]
[SI + d8] [DI + d8] [BP + d8] [BX + d8]	[BX + SI + d16] [BX + DI + d16] [BP + SI + d16] [BP + DI + d16]	[SI + d16] [DI + d16] [BP + d16] [BX + d16]

BX SI + disp
BP DI



Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing



Direct Addressing Mode

In this mode, address of the operand is directly specified in the instruction. Here only the offset address is specified, the segment being indicated by the instruction.

Example:

MOV CL, [4321H]

This instruction moves data from location 4321H in the data segment into CL.

The physical address is calculated as

DS * 10H + 4321

Assume DS = 5000H

∴PA = 50000 + 4321 = 54321H

∴CL ← [54321H]

Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing

In Register indirect addressing, name of the register which holds the effective address (EA) will

be specified in the instruction.

Registers used to hold EA are any of the following registers: BX, BP, DI and SI.

Content of the DS address calculation . is used for base address calculations

Note : Register/ memory enclosed in brackets refer to content of register/ memory

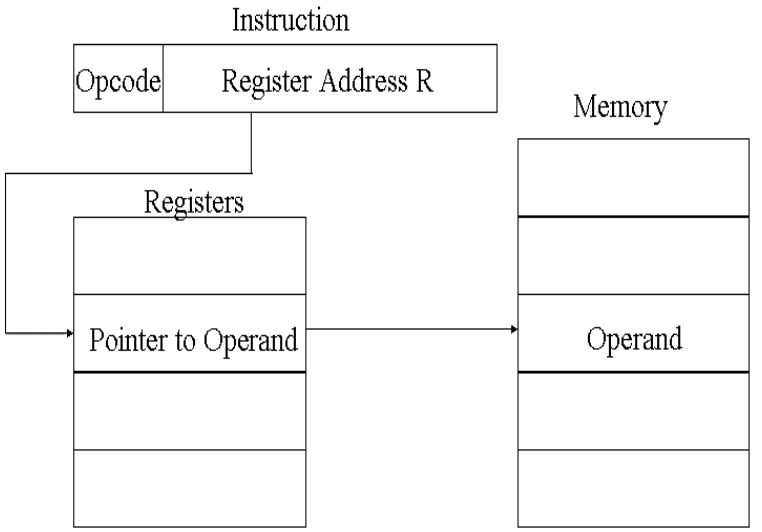
Example:

MOV CX, [BX]

Operations:

EA = (BX) 0r EA=(A)
BA = (DS) x 10H MA = BA + EA

(CX) ← (MA) or,
(CL) ← (MA)
(CH) ← (MA +1)



Physical address can be calculated as DS * 10H + BX.

Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing

In Based Addressing, BX or BP is used to hold the base value for effective address and a signed 8-bit or unsigned 16-bit displacement will be specified in the instruction.

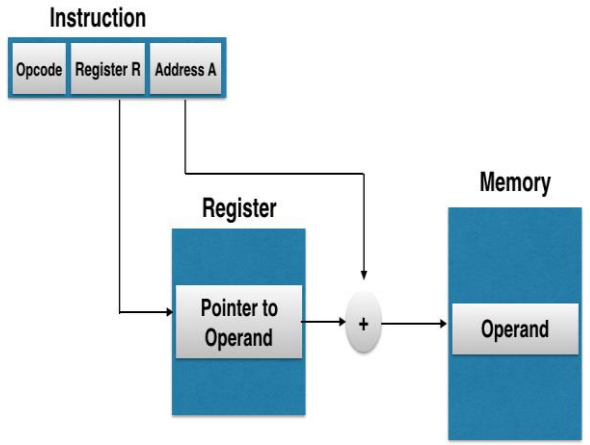
In case of 8-bit displacement, it is sign extended to 16-bit before adding to the base value.

When BX holds the base value of EA, 20-bit physical address is calculated from BX and DS.

When BP holds the base value of EA, BP and SS is used.

Example:

MOV AX, [BX + 08H]



Operations:

$0008_H \leftarrow 08_H$ (Sign extended) EA = (BX) + 0008_H

BA = (DS) x 16_{10}

MA = BA + EA

(AX) ← (MA) or,

(AL) ← (MA)

(AH) ← (MA + 1)

Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing

SI or DI register is used to hold an index value for memory data and a signed 8-bit or unsigned 16- bit displacement will be specified in the instruction.

Displacement is added to the index value in SI or DI register to obtain the EA.

In case of 8-bit displacement, it is sign extended to 16-bit before adding to the base value.

Example:

MOV CX, [SI + 0A2H]

Operations:

$FFA2_H \leftarrow A2_H$ (Sign extended)

$EA = (SI) + FFA2_H$

$BA = (DS) \times 16_{10}$

$MA = BA + EA$

$(CX) \leftarrow (MA)$ or,

$(CL) \leftarrow (MA)$

$(CH) \leftarrow (MA + 1)$

Addressing Modes

1. Register Addressing
2. Direct Addressing
3. Register Indirect Addressing
4. Based Addressing
5. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing

Immediate Addressing In Based Index Addressing, the effective address is computed from the sum of a base register (BX or BP), and index register (SI or DI) and a displacement .

Example: MOV DX, [BX + SI + 0AH]

Operations:

$000A_H \leftarrow 0A_H$ (Sign extended)

$EA = (BX) + (SI) + 000A_H$

$BA = (DS) \times 10H$

$MA = BA + EA$

(DX) (MA) or ,

(DL) $\leftarrow$ (MA)

(DH) $\leftarrow$ (MA + 1)

Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing

Employed in string operations to operate on string data.

The effective address (EA) of source data is stored in SI register and the EA of destination is stored in DI register.

Segment register for calculating base address of source data is DS and that of the destination data is ES

Example: MOVS BYTE

Operations:

Calculation of source memory location:
EA = (SI) BA = (DS) x 10H MA = BA + EA

Calculation of destination memory location:
EA_E = (DI) BA_E = (ES) x 10H $MA_E = BA_E + EA_E$

Note : Effective address of the Extra segment register

(MAE) ← (MA)

If DF = 1, then (SI) ← (SI) – 1 and (DI) = (DI) - 1
If DF = 0, then (SI) ← (SI) +1 and (DI) = (DI) + 1

Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing

These addressing modes are used to access data from standard I/O mapped devices or ports.

In direct port addressing mode, an 8-bit port address is directly specified in the instruction.

Example: **IN AL, [09H]**

Operations: $PORT_{addr} = 09_H$
(AL) ← (PORT)

Content of port with address 09_H is moved to AL register

In indirect port addressing mode, the instruction will specify the name of the register which holds the port address. In 8086, the 16-bit port address is stored in the DX register.

Example: **OUT [DX], AX**

Operations: $PORT_{addr}$ = (DX)
(PORT) ← (AX)

Content of AX is moved to port whose address is specified by DX register.

Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Implied Addressing

IN this addressing mode, the effective address of a program instruction is specified relative to Instruction Pointer (IP)by an 8-bit signed displacement.

Example:JZ0AH

Operations:

$000A_H \leftarrow 0A_H$ (sign extend)

If ZF =1then

EA = (IP) + $000A_H$
BA = (CS) x 16_{10}
MA = BA + EA

If ZF = 1, then the program control jumps to new address calculated above.
If ZF = 0, then next instruction of the program is executed.

Addressing Modes

1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Addressing
6. Indexed Addressing
7. Based Index Addressing
8. String Addressing
9. Direct I/O port Addressing
10. Indirect I/O port Addressing
11. Relative Addressing
12. Relative Addressing

Instructions using this mode have no operands. The instruction itself will specify the data to be operated by the instruction.

Example: `CLC`

This clears the carry flag to zero.

Thank you so much

Any questions ?