

MICROPROCESSOR AND ASSEMBLY LANGUAGE  
CSE 311  
TOPIC: 8086 ADDRESSING MODES

---

Shaila Rahman  
Assistant Professor

# 8086 ADDRESSING MODE

Addressing mode is the way to fetch the operands that are needed for instruction execution. This is defined by the instruction itself.

# CATEGORIES

There are five categories of addressing modes. They are

1. Register and Immediate Addressing Modes
2. Memory Addressing Mode
3. I/O Addressing Mode
4. Relative Addressing Mode
5. Implied Addressing Mode

# REGISTER AND IMMEDIATE ADDRESSING MODES

Two types

- i. Register Addressing Mode
- ii. Immediate Addressing Mode
- iii. Register Addressing Mode

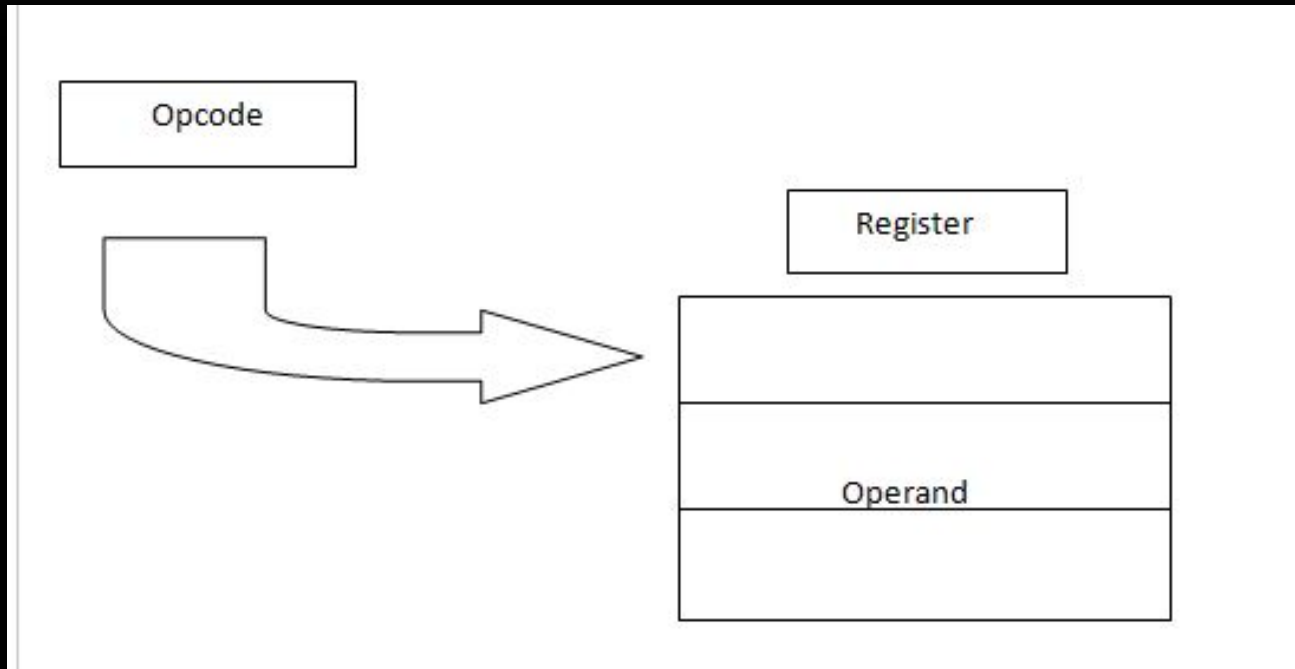
The operand is specified using register.

Syntax: <Opcode> <DestReg> <SourceReg>

e.g. MOV AX,BX

ADD AL, BL

# DATA AVAILABLE IN REGISTER/ FASTEST ACCESS



# REGISTER AND IMMEDIATE ADDRESSING MODES

## ii. Immediate Addressing Mode

The operand is specified as an immediate value by the instruction.

Syntax: <Opcode> <DestReg><Source Immediate value>

<Opcode> <Memory> <Source Immediate value>

e.g. MOV BX,1234H

MOV [BX], 12H

# DATA AVAILABLE IN INSTRUCTION

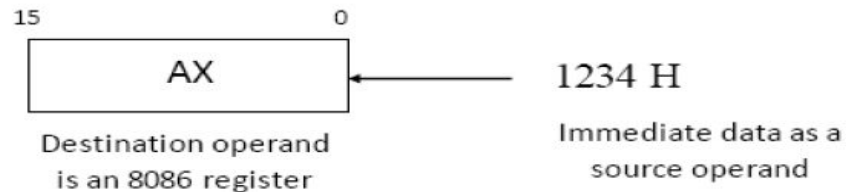
## Immediate Addressing Mode

In an immediate mode, 8 or 16-bit data can be specified as a part of instruction.

**MOV AL, 20 H**



**MOV AX, 1234 H**



Arrow indicates direction of data flow.

# MEMORY ADDRESSING MODE

According to memory addressing mode the offset/ EA (Effective Address) 16-bit is specified instead of the operand using a register or as an immediate value. So it needs time to calculate the memory address. Accessing is slower compared to category one.



# CATEGORIES OF MEMORY ADDRESSING

Six types

- i. Register Indirect Addressing Mode
- ii. Direct Addressing Mode
- iii. Based Addressing Mode
- iv. Indexed Addressing Mode
- v. Based-Indexed Addressing Mode
- vi. String Addressing Mode

# REGISTER INDIRECT ADDRESSING MODE

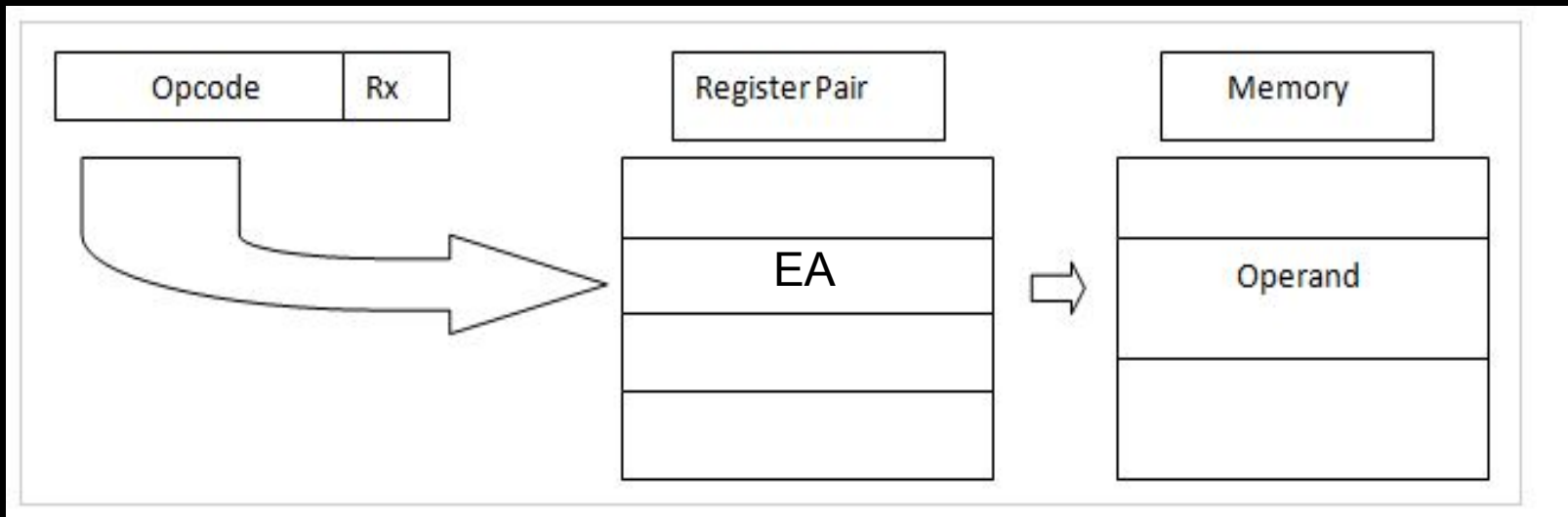
The EA is given by a register (BX/ SP/ BP/ SI/ DI)

Syntax: <Opcode> <DestReg> <EA by Reg BX/ BP/ SP/ SI/ DI>

e.g. MOV AX,[BX]

- e.g. EA= 2340h =BX
- DS= 0123h
- PA = DS\*10h+ EA
- = (0123\*10 +2340)h
- =01230+2340 =03570h
- Data from 03570 ----- AL
- Data from 03571 ---- AH

# REGISTER GIVING THE OFFSET/EA



# DIRECT ADDRESSING MODE

The EA is specified as an immediate value.

Syntax: <Opcode> <DestReg> <Source immediate EA >

e.g. MOV AX, [1234h]

PA= DS<< 4bit + 1234h

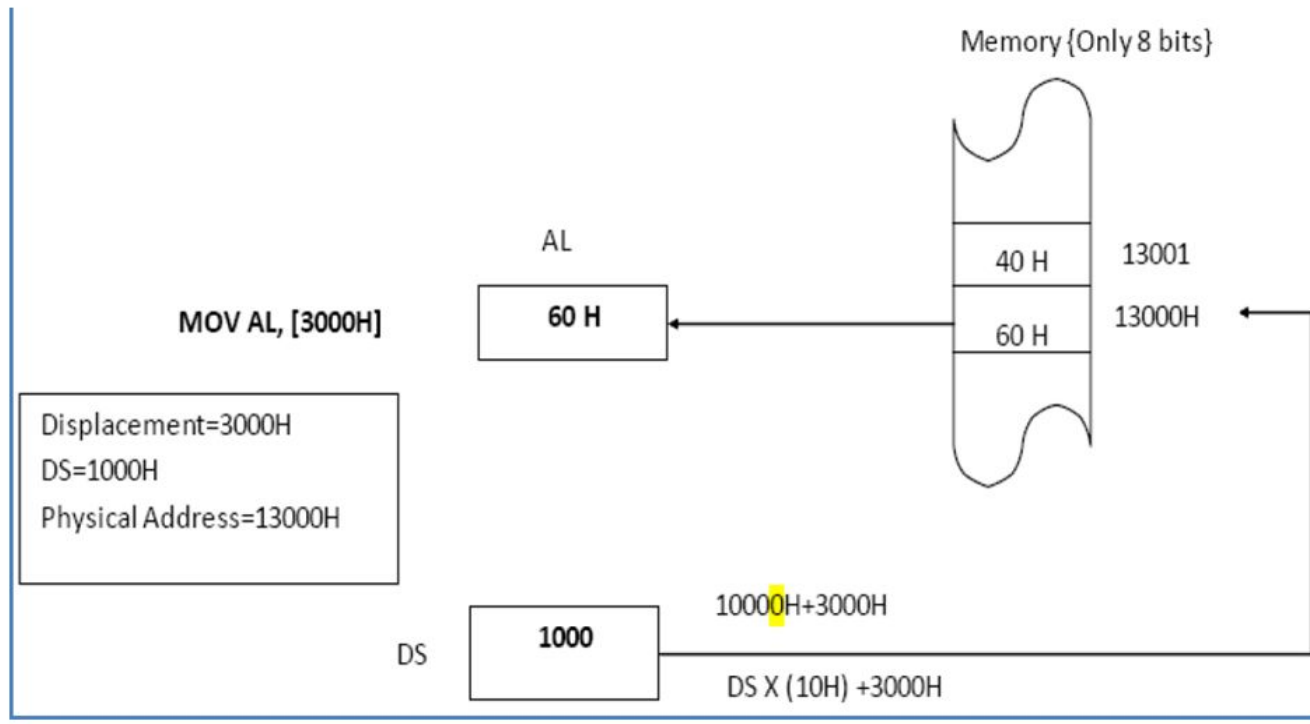
= 01230 + 1234

= 02464 h

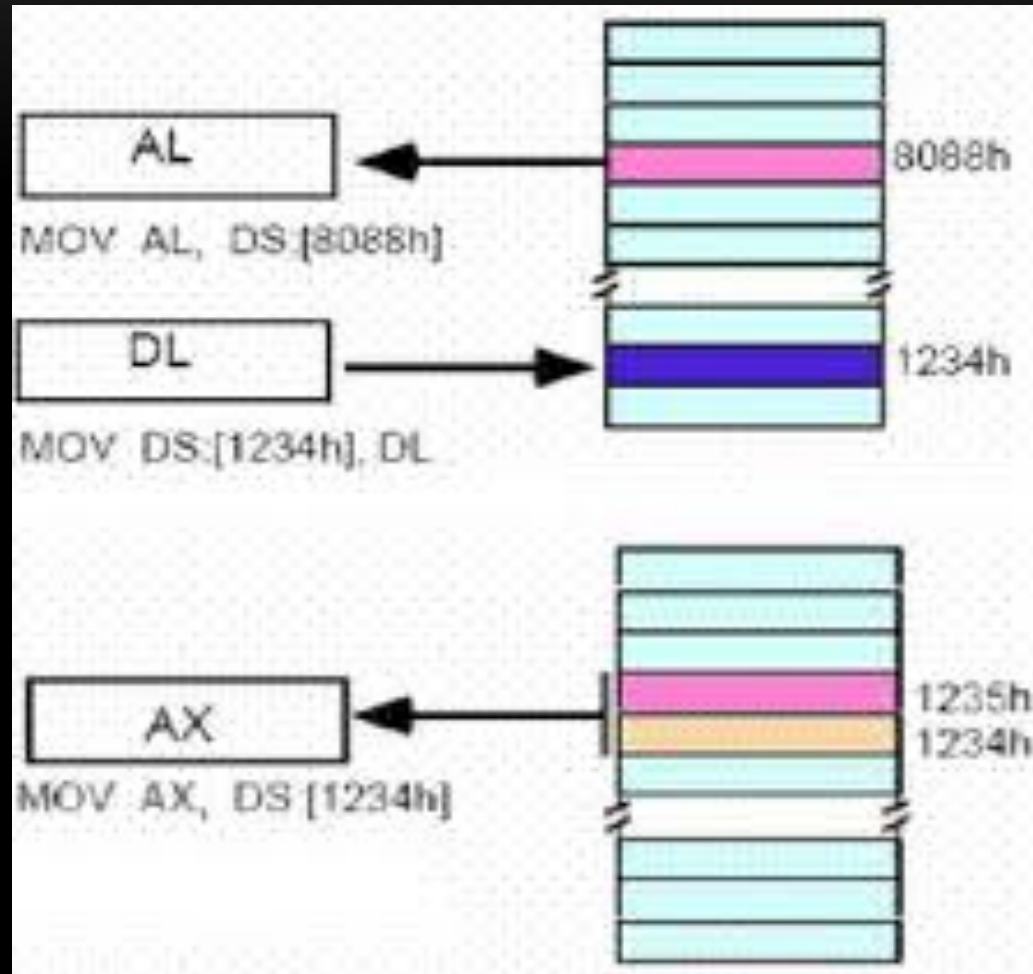
# OFFSET/EA IS GIVEN IN INSTRUCTION

## Direct Addressing Mode:

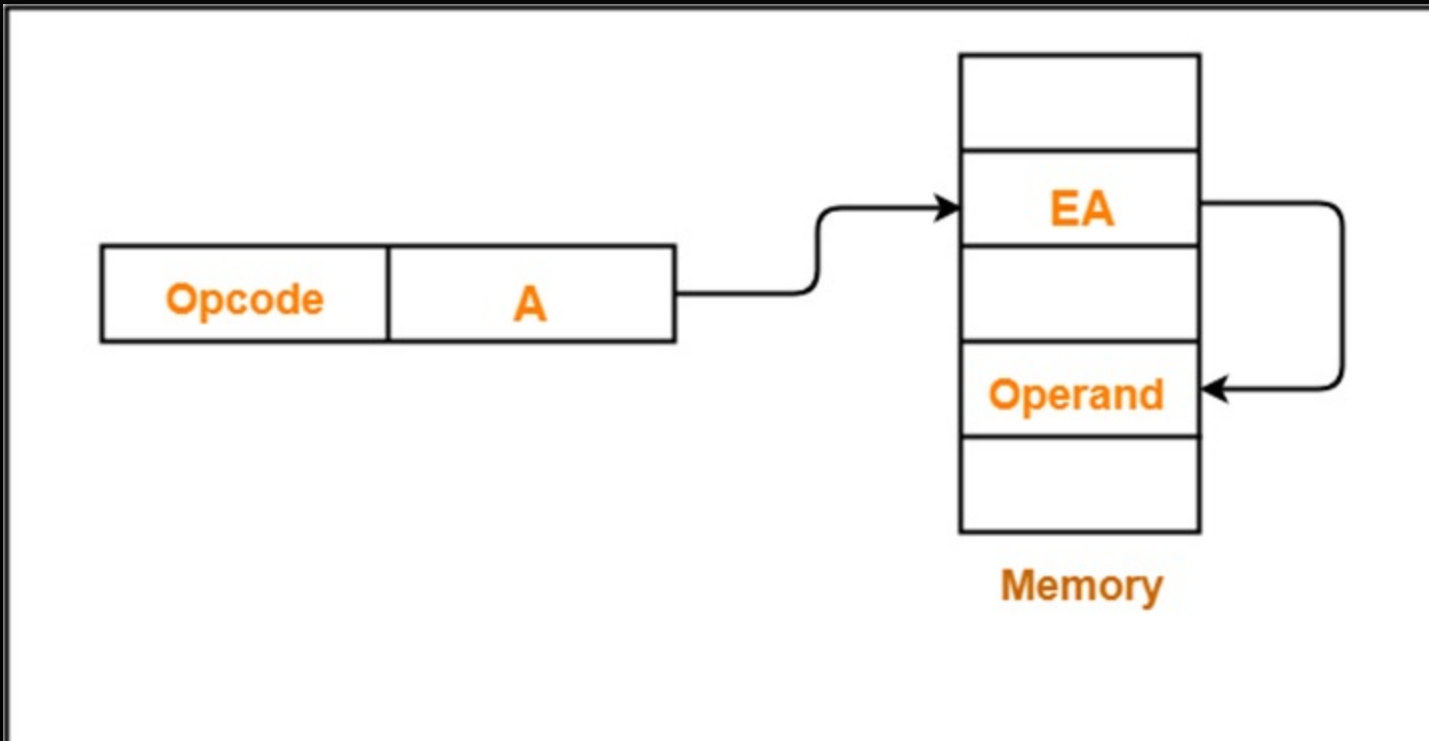
In this mode, the 16-bit effective address (EA) is taken directly from the displacement field of the instruction.



# EXAMPLE



# EXAMPLE



# BASED ADDRESSING MODE

The EA is given by the based register (BX/BP) and an immediate value as displacement. The displacement can be either 8-bit signed or 16-bit unsigned. To access a block of data from a particular base.

Syntax: <Opcode> <DestReg> <Source EA >  
(BX/BP+displacement)

e.g. i. MOV AX,A[BX]

EA= BX+A      BX/BP is fixed but disp can be incremented/decremented

PA= DS: BX+A

ii. MOV AX, [A+BP]

EA= BP+A

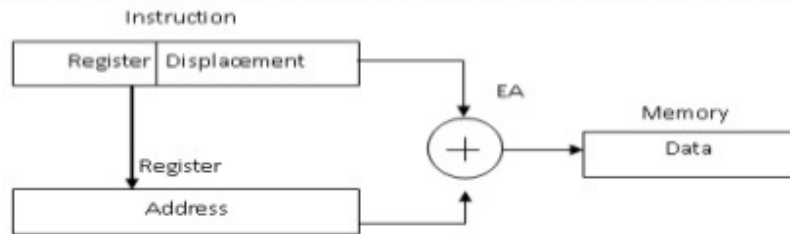
PA= SS: BP+A



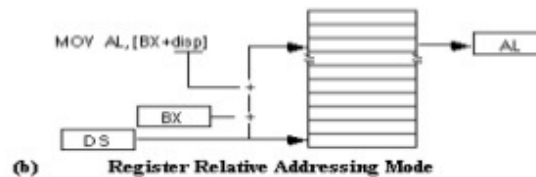
# TWO CONTENTS GIVING EA (BX/BP+ DISP)

## Register Relative

- Effective address = [ Base/pointer register] + 8 or 16 bit displacement
- Base/Pointer register : BX or BP

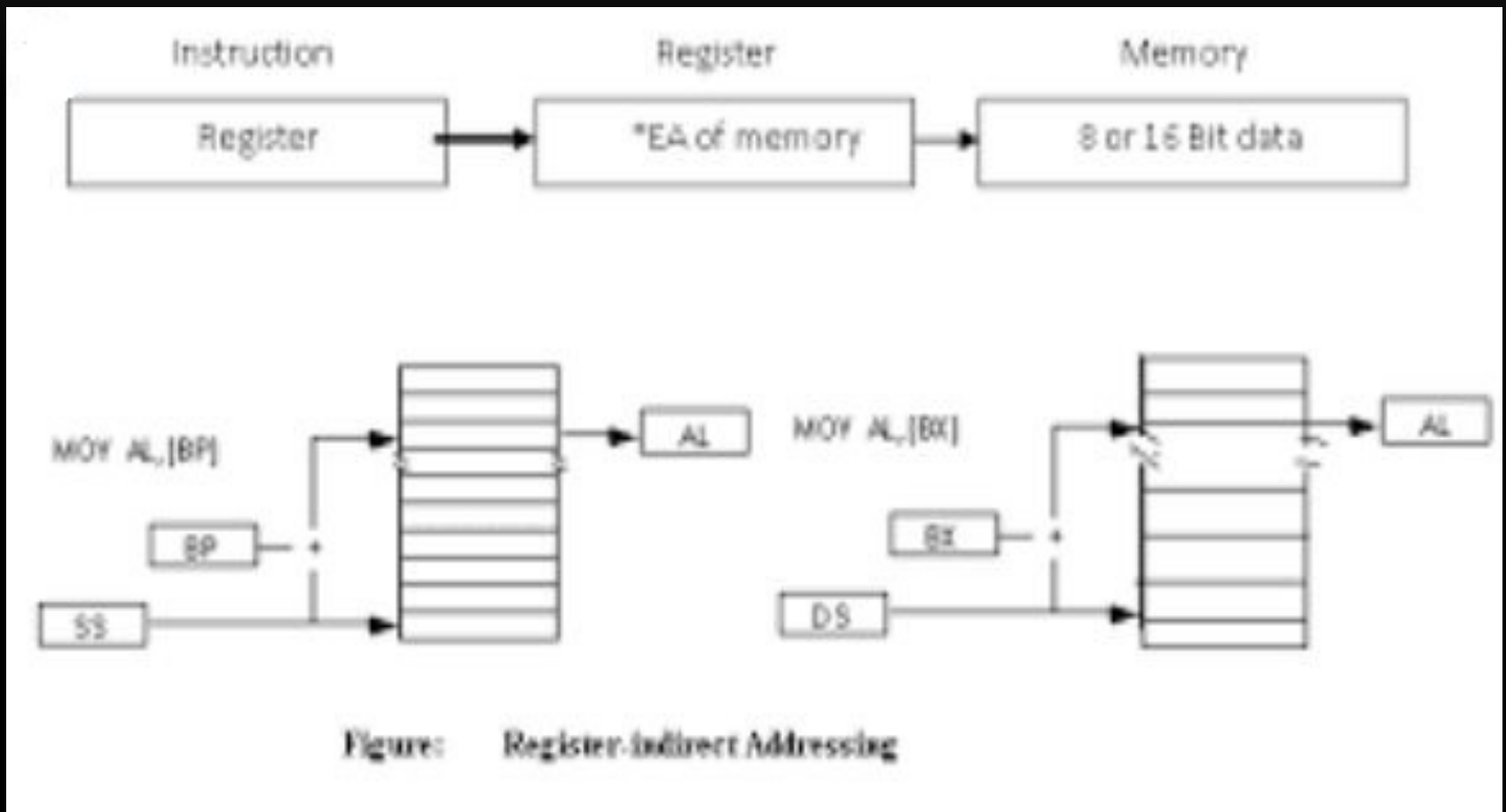


(a) Register Relative Addressing Mode



(b) Register Relative Addressing Mode

# EXAMPLES



# INDEXED ADDRESSING MODE

The EA is given by the index register (SI/DI) and an immediate value as displacement. The displacement can be either 8-bit signed or 16-bit unsigned. To access array type data.

Syntax: <Opcode> <DestReg> <SourceEA>

(SI/DI+displacement)

e.g. i. MOV AX, A[SI] ----- Disp is fixed but SI/DI can be incremented/decremented

EA = SI + A

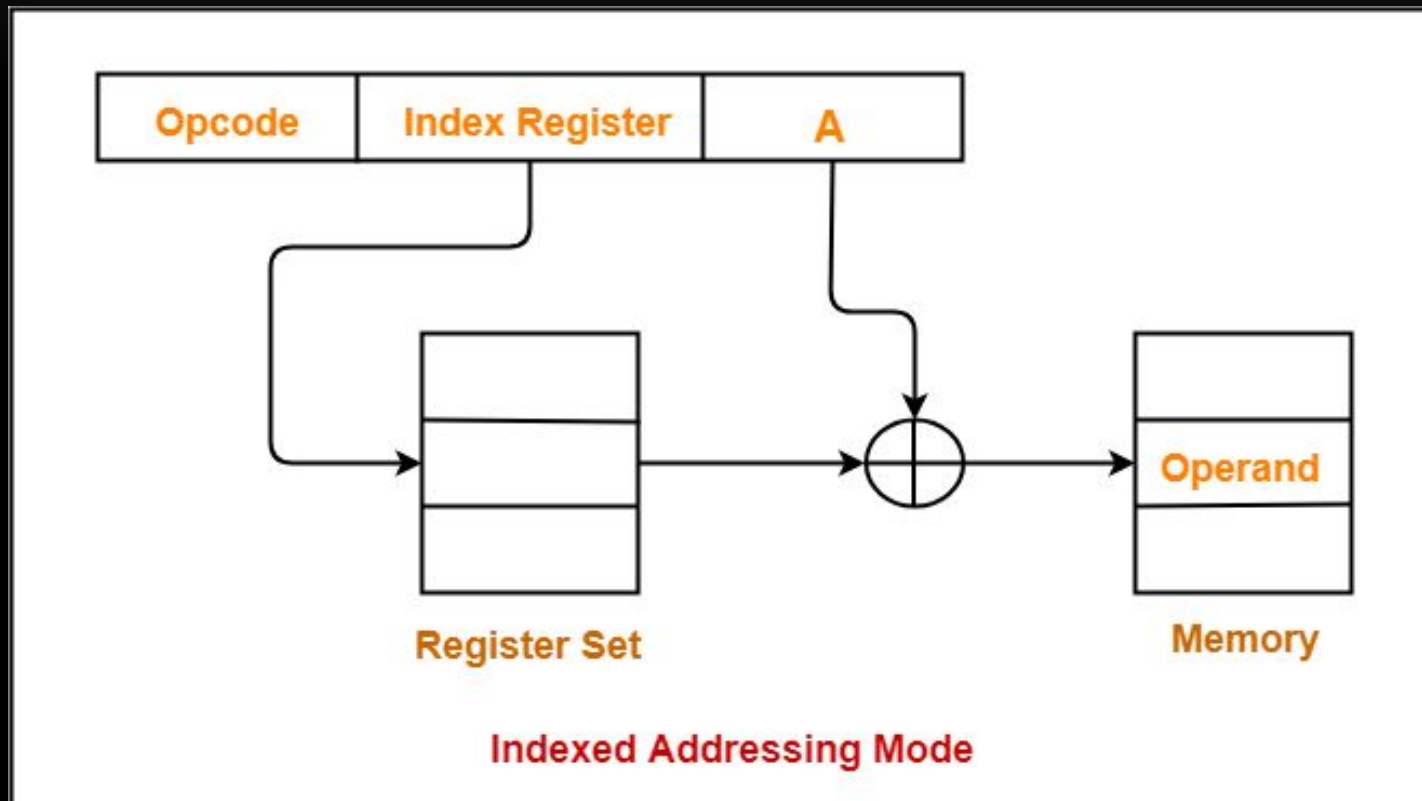
PA = DS \* 10h + EA

ii. MOV AX, [A+DI]

EA = DI + A

PA = DS \* 10h + EA

# TWO CONTENTS GIVING EA (SI/DI+ DISP)



# BASED-INDEXED ADDRESSING MODE

The EA is given by the based register (BX/BP), the index register (SI/DI) and an immediate value as displacement. The displacement can be either 8-bit signed or 16-bit unsigned. They are used to access 2-D array.

Syntax: <Opcode> <DestReg> <Source EA >

$(BX/BP) + (SI/DI) + \text{displacement}$

e.g. i. MOV AX, A[SI][BX]

Offset/ EA = BX+SI+Disp      BX/BP fixed , but SI/DI and disp increment/decrement

PA= DS: EA

ii. MOV AX, A[SI][BP]

Offset/ EA = BP+SI+Disp

PA= SS: EA

# THREE VALUES GIVING OFFSET/EA

## Relative Based Indexed

- Effective address = [Base register] + [Index register] + 8 or 16 bit displacement

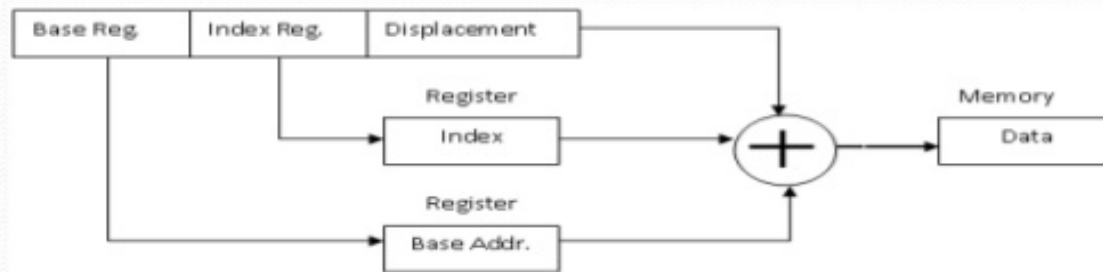


Figure (a) Relative Based Indexed Addressing Mode

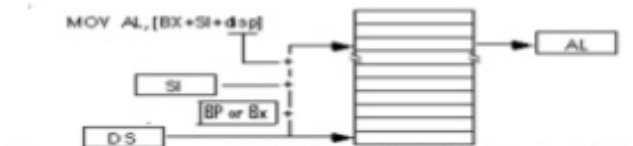


Figure (b) Relative Based Indexed Addressing Mode

# STRING ADDRESSING MODE

String instructions are implicit i.e. operand is hidden from the instruction. They by default use the index registers SI/DI to point the source/destination string by giving offset.

Syntax:<Opcode>

e.g. MOVSB ; move string bytes

MOVSW ;move string words

COMPSB ; Compare Sting bytes

DS=0123h, ES= 0234h , SI= 2345h, DI= 4567h

Source PA= DS: SI =  $01230 + 2345 = 03575h$

Destination PA = ES: DI =  $02340 + 4567 = 064A7h$

Data from 03575h moved to 068A7h

Source offset give by SI and Destination offset given by DI. Source PA given by DS: SI and Destination PA given by ES: DI

SI/DI will be auto-incremented/ decremented by 1/2 for byte/word operation when DF=0/1

# I/O ADDRESSING MODE

According to I/O addressing mode the port address is specified by the instruction as an immediate value or using a particular register DX. Two Types

- I. Direct I/O Addressing Mode
- II. Indirect I/O Addressing Mode



# DIRECT I/O ADDRESSING MODE

The port address is given as an immediate value.

Syntax: <Opcode> <DestReg AX> <Source Port Address>

e.g. IN AX, Port\_A (Word Operation)

IN AL, Port\_A (Byte Operation)

# INDIRECT I/O ADDRESSING MODE

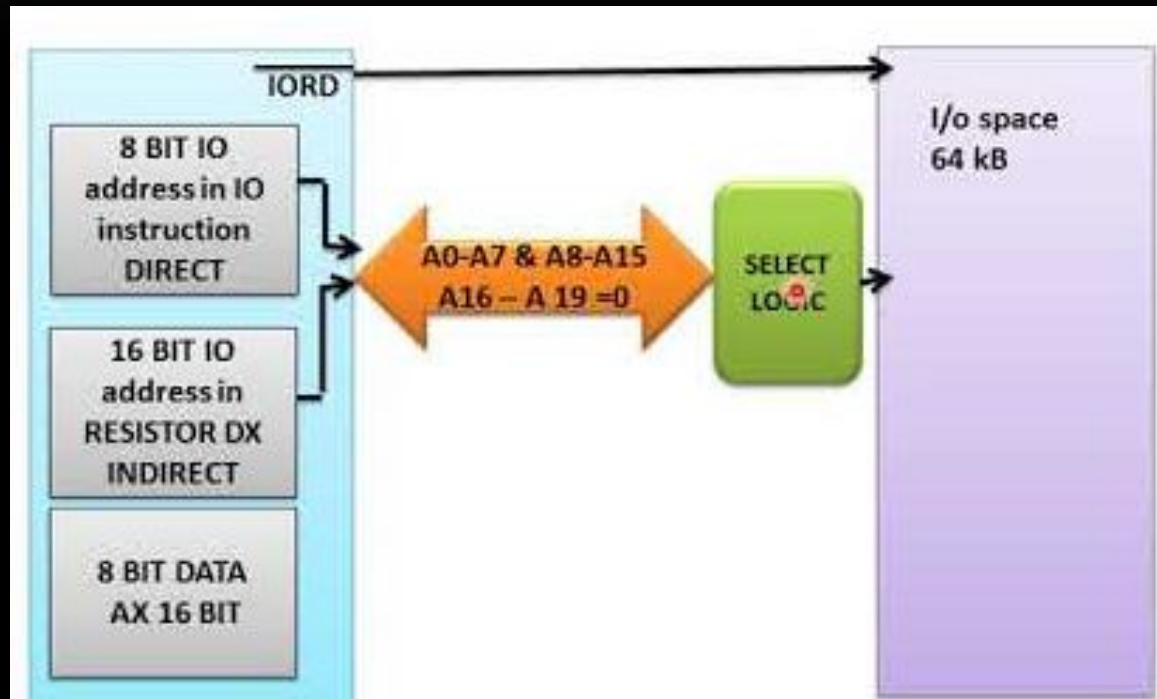
A register DX gives the port address.

Syntax: <Opcode> <DestReg AX> <Source Port Address in DX>

e.g. OUT DX, AL (Byte Operation)

OUT DX, AX (Word Operation)

# DIRECT AND INDIRECT MAPPING



# RELATIVE ADDRESSING MODE

This is basically used by branch instructions. According to this mode a displacement value (8-bit signed or 16-bit unsigned) is given which modifies the IP contents to make a jump on non-sequential branch address.

Syntax: <Opcode> <displacement>

e.g. JUMP L

JNZ L

## Two Types

Intrasegment Jump: This modifies only the IP and CS remains unchanged.

Intersegment Jump: This modifies both the IP and CS contents

# IMPLIED ADDRESSING MODE

According to this mode the instruction carries no operand.

Syntax: <Opcode>

e.g. HLT, NOP, CLC, CLS