MICROPROCESSOR AND ASSEMBLY LANGUAGE CSE 311 TOPIC: 8086 ADDRESSING MODES

Shaila Rahman

Assistant Professor

8086 ADDRESSING MODE

DS ES SS CS

Data Code

Data

→ Register

→ Memory →

→ Immd

→ Port

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 — cs
5. Implied Addressing Mode

REGISTER AND IMMEDIATE ADDRESSING MODES

Two types

i. Register Addressing Mode

ii. Immediate Addressing Mode

i. Register Addressing Mode

The operand is specified using register.

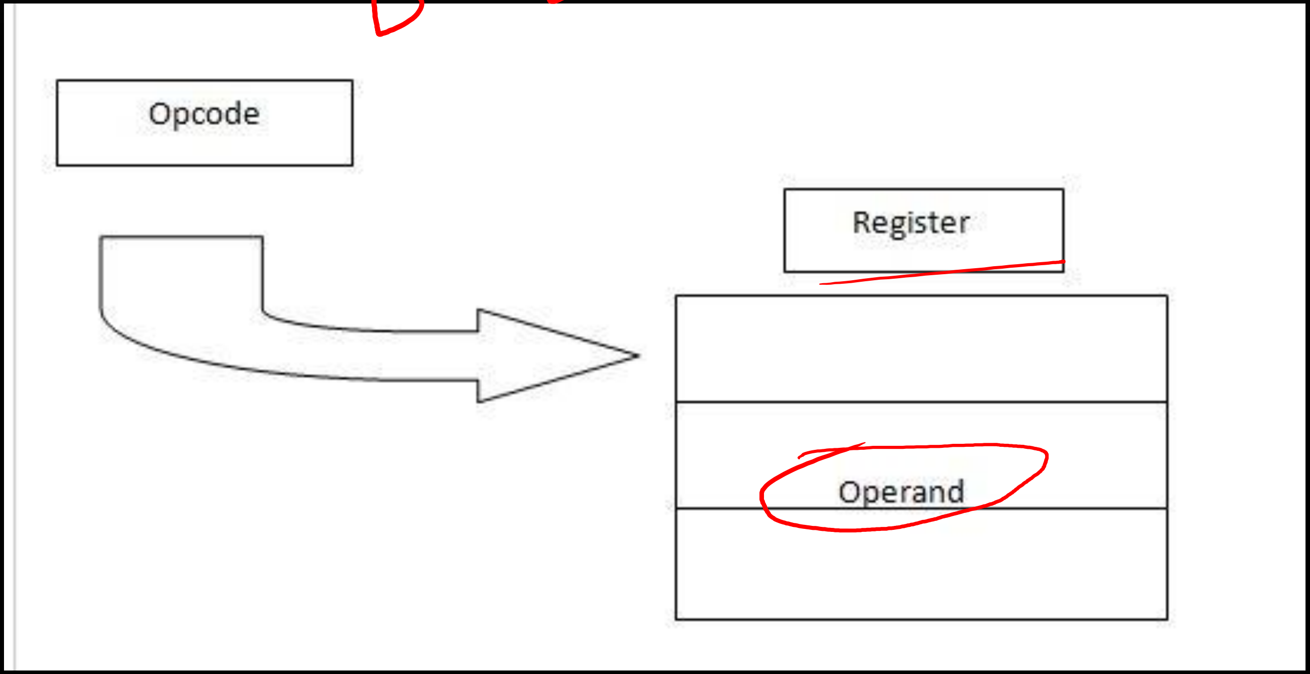
Syntax: <Opcode> <DestReg> <SourceReg>

e.g. MOV AX,BX — Word

ADD AL, BL — Byte

DATA AVAILABLE IN REGISTER/ FASTEST ACCESS

Fastest Mode



REGISTER AND IMMEDIATE ADDRESSING MODES

ii. Immediate Addressing Mode   Constant

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   eg MOV CL , 5H

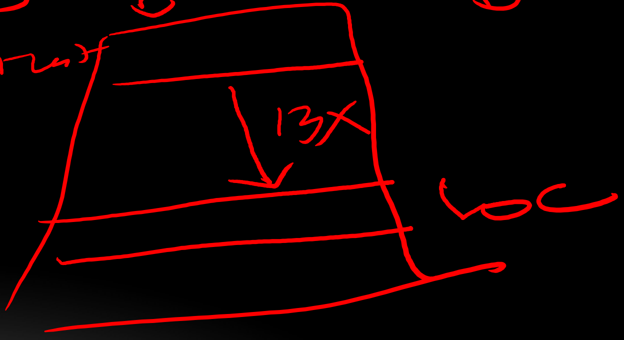
MOV [BX], 12H

DS Reg ?   DS

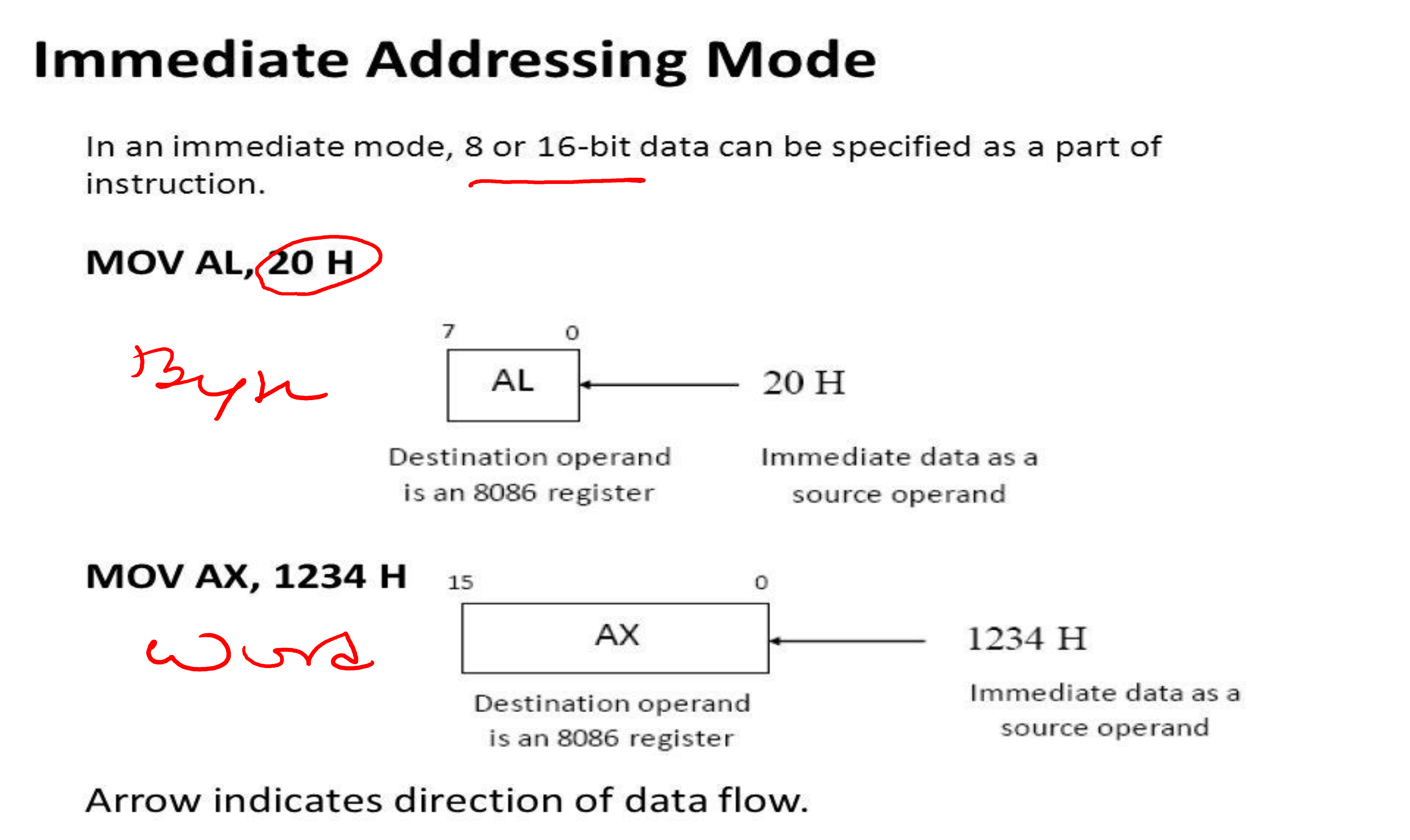
Source   Dest

Reg   Imm

Memory   Imm



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

Byte

7 0

AL ← 20 H

Destination operand is an 8086 register

Immediate data as a source operand

MOV AX, 1234 H

Word

15 0

AX ← 1234 H

Destination operand is an 8086 register

Immediate data as a source operand

Arrow indicates direction of data flow.

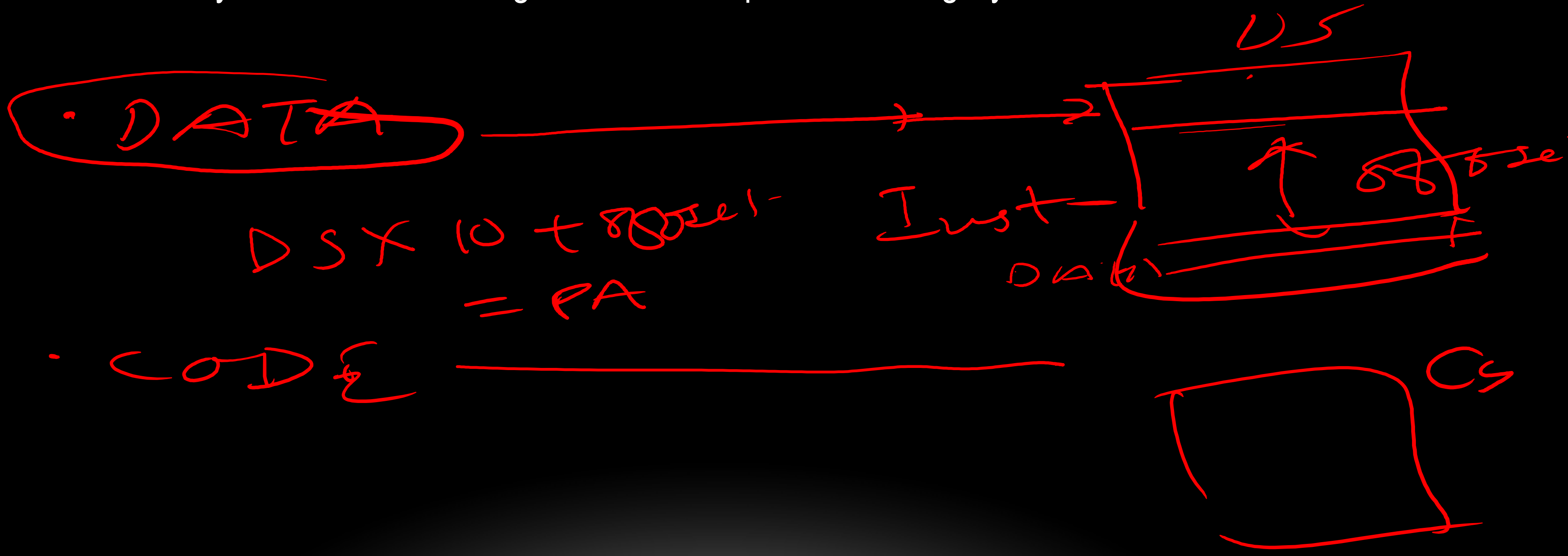
offset

IP — CS

MEMORY ADDRESSING MODE

ES – DI   DS — BX, SI, DI, SP, BP SS

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

DS X 10 + BX

offset

PUSH ~~AX~~ [BX]

put SS X 10 + SP =

REGISTER INDIRECT ADDRESSING MODE

Single Data

SS SS DS

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]

opcode source offset dest Reg

- 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

MOV AX, [BX]

PA = DS X 10 + BX
=

MOV AX, [BP]

SS X 10 + BP =

ADD AX, [BX]

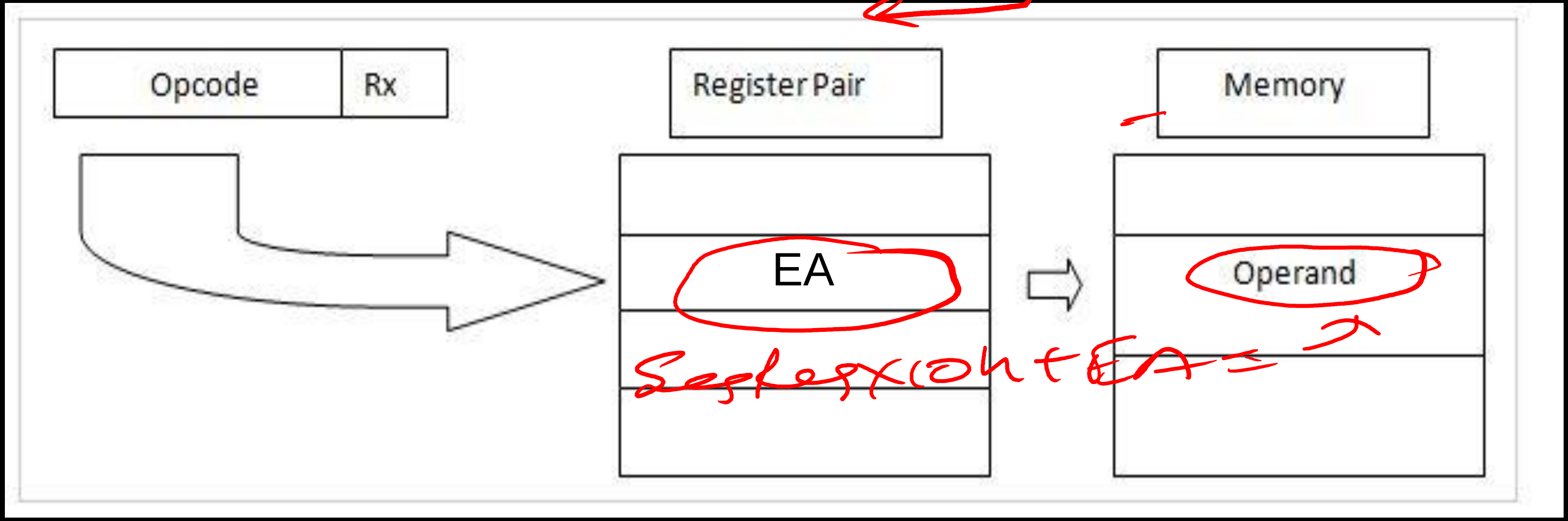
Memory

offset

REGISTER GIVING THE OFFSET/EA

~~ADD AX~~ DS×10 + BX

DATA



A db 5

mov AX, A

DIRECT ADDRESSING MODE

Single DATA

The EA is specified as an immediate value.

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

e.g. MOV AX, [1234h]

DS x 10 + 1234 = PA

DS = 0123H

PA= DS<< 4bit + 1234h

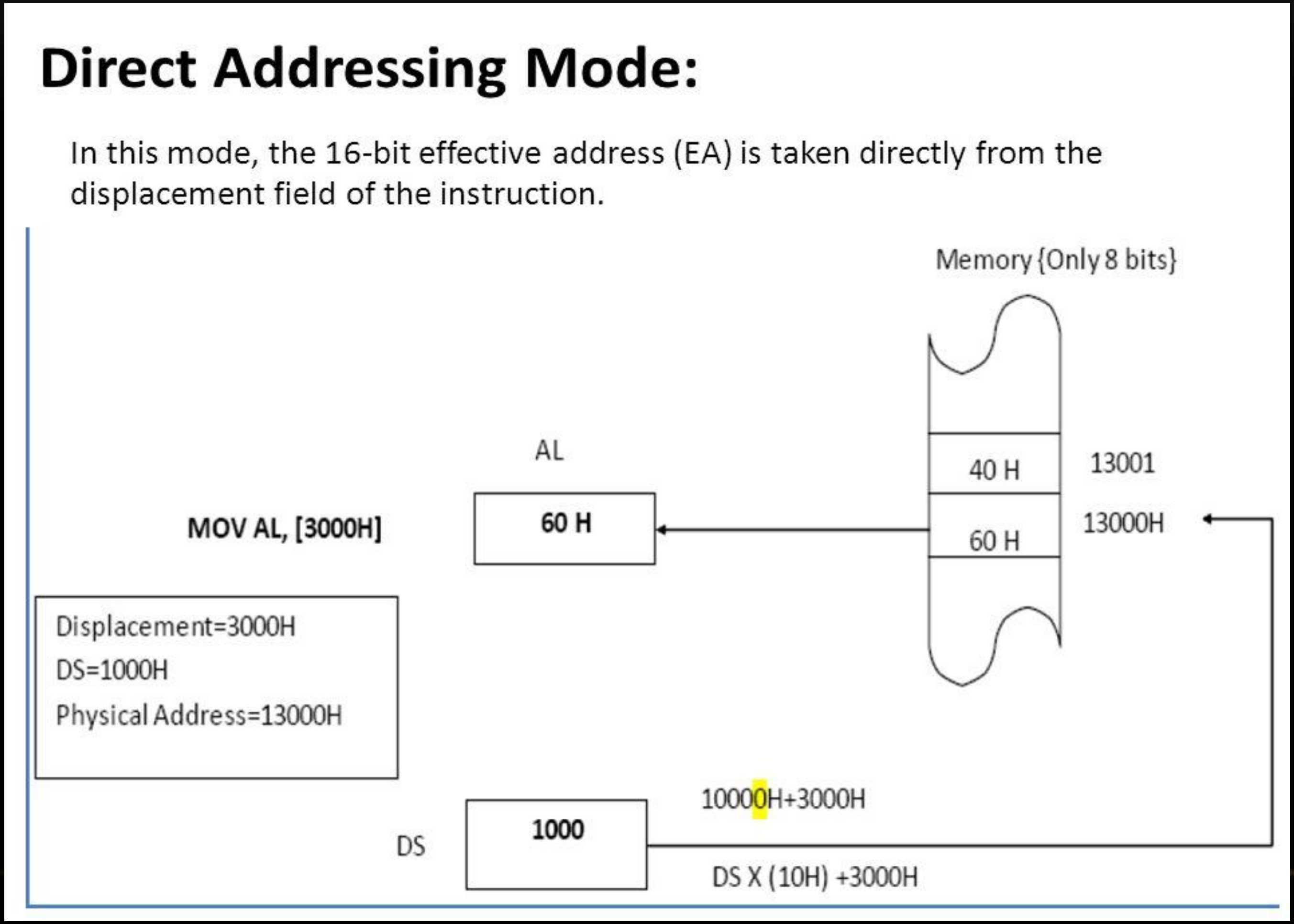
= 01230 + 1234

= 02464 h

02464 → AL

02465 → AH

OFFSET/EA IS GIVEN IN INSTRUCTION

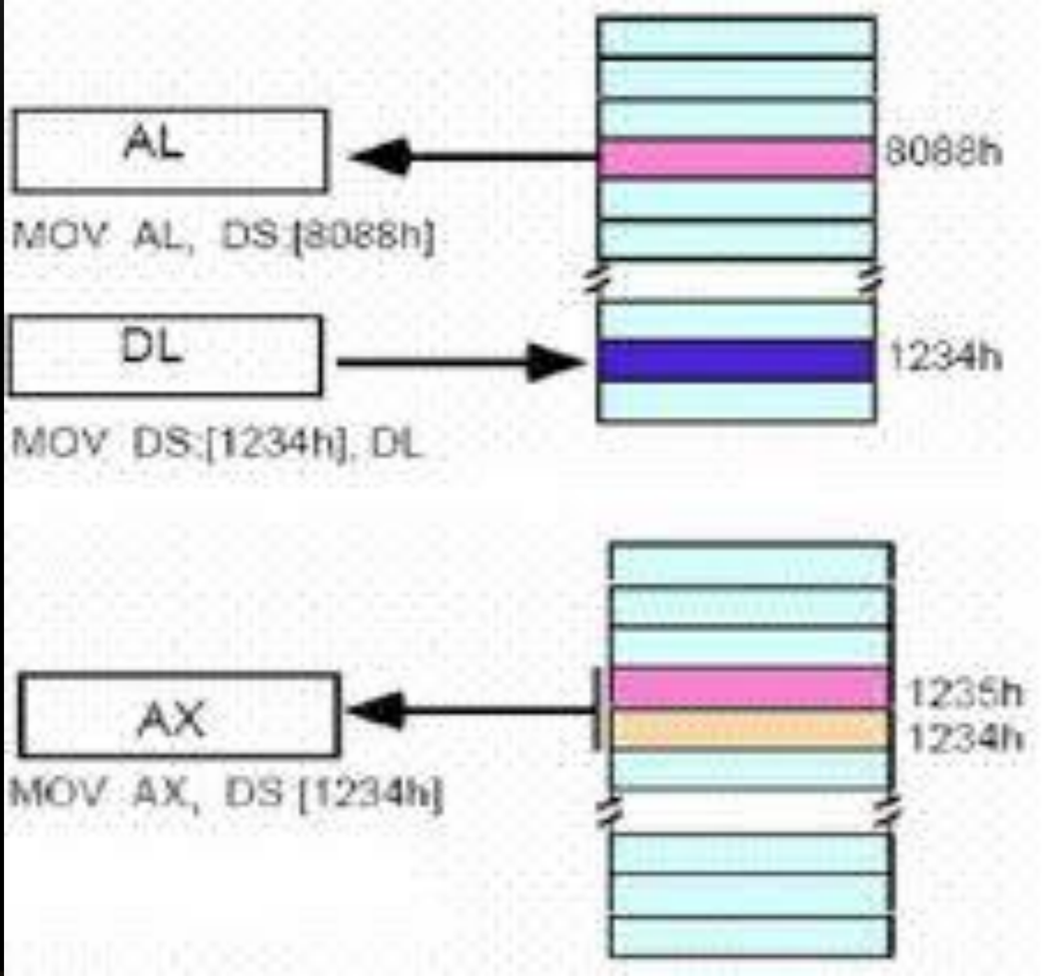


Direct Addressing Mode:

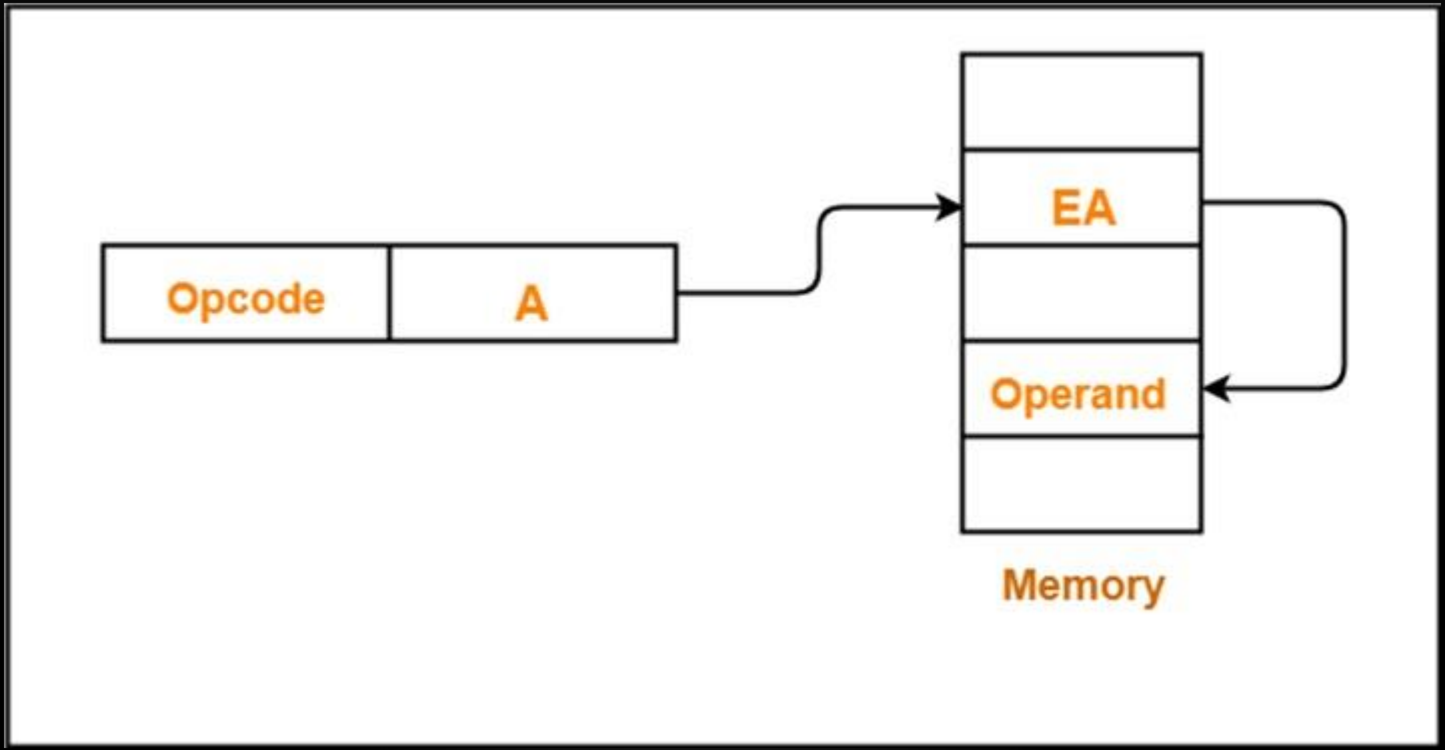
In this mode, the 16-bit effective address (EA) is taken directly from the displacement field of the instruction.

Displacement=3000H
DS=1000H
Physical Address=13000H

EXAMPLE


AL
8088h
MOV AL, DS:[8088h]
DL
1234h
MOV DS:[1234h], DL
AX
1235h
1234h
MOV AX, DS:[1234h]

EXAMPLE



BX + DISP(Index) = offset

DS x 10 + offset = PA

Listed Data Array

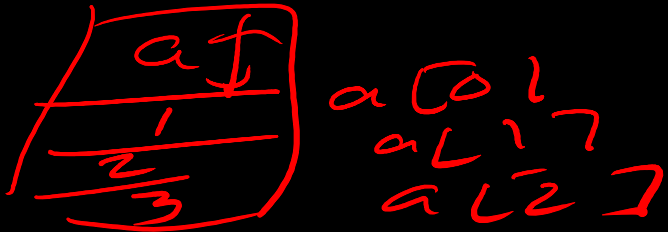
BASED ADDRESSING MODE

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.

a[2] = {1, 2, 3}

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

(BX/BP+displacement)



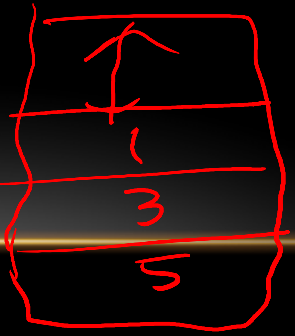
e.g. i. MOV AX,A[BX]

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

PA= DS: BX+A

a[2][2] = {1, 3, 5, 6, 9, 10, 11, 12, 13}

ii. MOV AX, [A+BP]

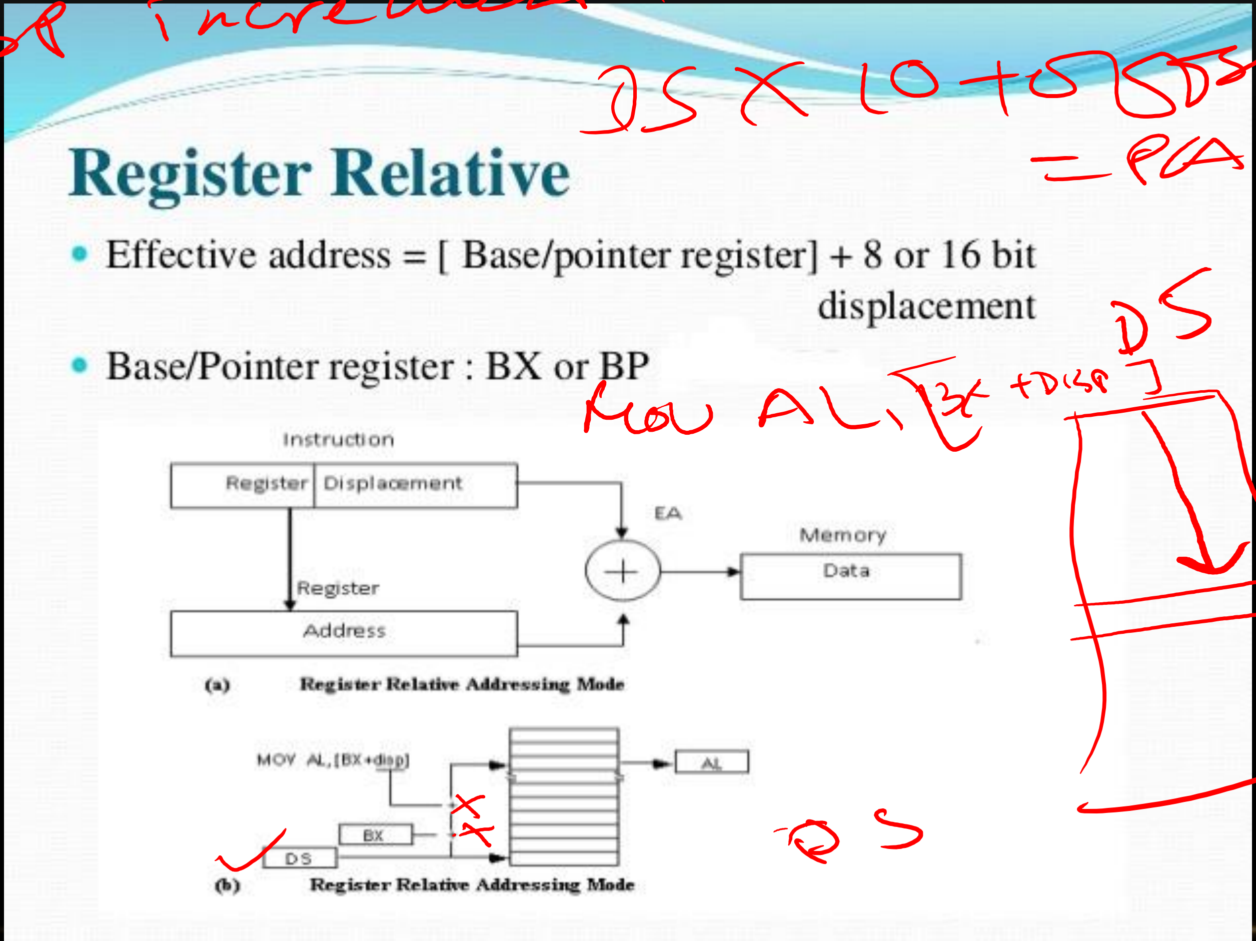


EA= BP+A

PA= SS: BP+A

TWO CONTENTS GIVING EA (BX/BP+ DISP)

Disp increment / Decrement



Register Relative

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

DS × 10 + offset = PA

MOV AL, [BX +DISP]

DS

BX + Disp = offset

BX disp displ

DS

EXAMPLES

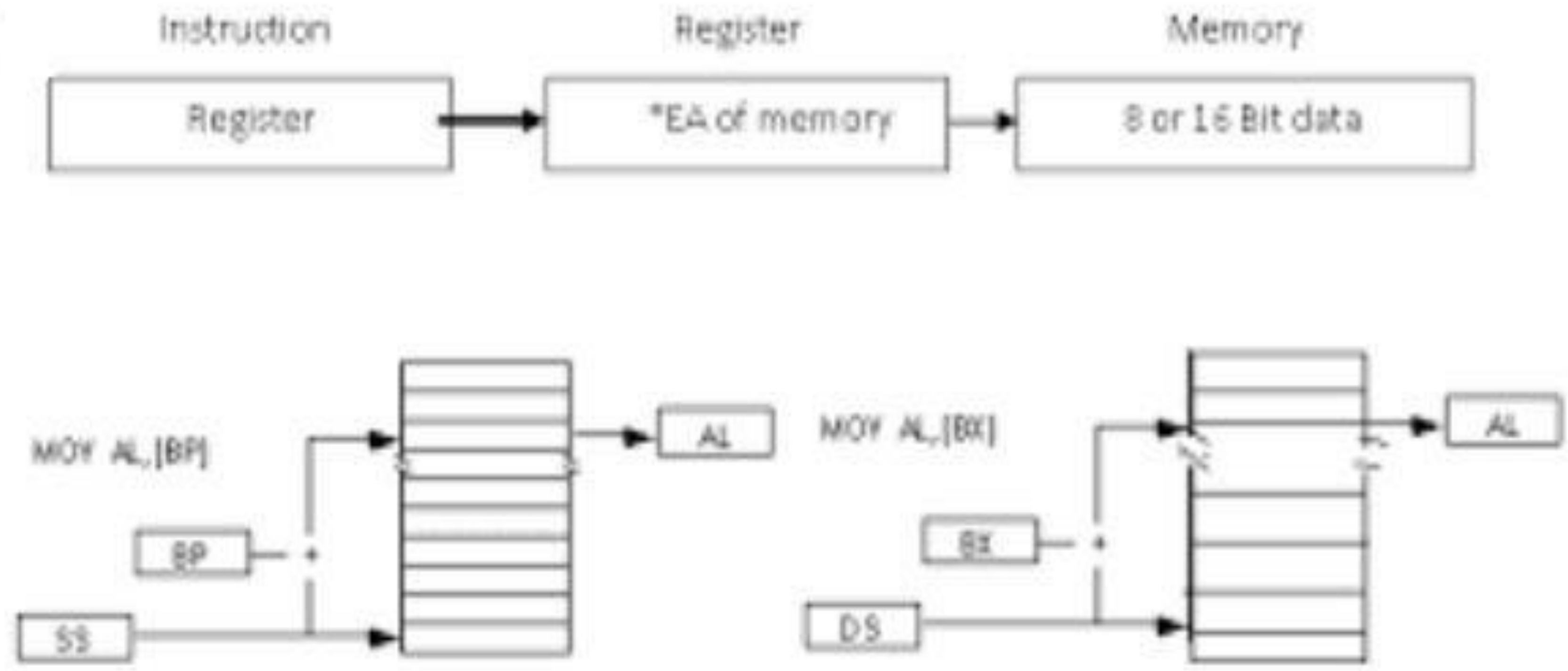


Figure: Register-indirect Addressing

INDEXED ADDRESSING MODE

SI/DI

↓ DS

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

offset (SI/DI + DISP)

PA = DS × 10 + offset

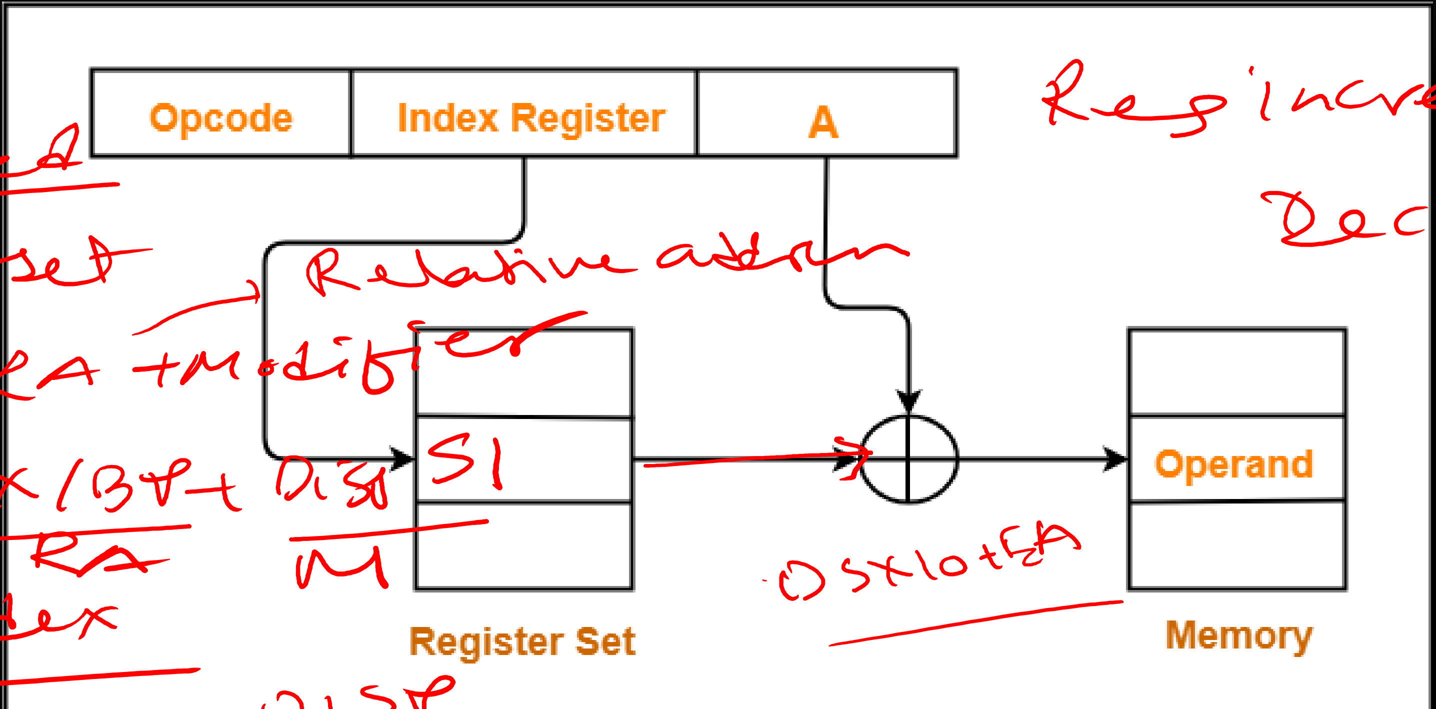
ii. MOV AX, [A+DI]

EA= DI+A

PA = DS*10h + EA

DF = 0 , SI/DI auto increment
= 1 SI/D ^ decrement

TWO CONTENTS GIVING EA (SI/DI+ DISP)



Indexed Addressing Mode

Reg increment
Decrement

Based

Offset = RA + Modifier

Relative addressing

BX/BP + Disp
RA M

DS x 10 + EA

Index

RA = DISP
M = SI/DI

Offset = 3 values BX/BP (Fixed) + (SI/DI) change + DISP Change

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)+displacement

a[i][i] = { 2, 3, 4, 5 };

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

Offset = RA + Modifier1 + Modifier2

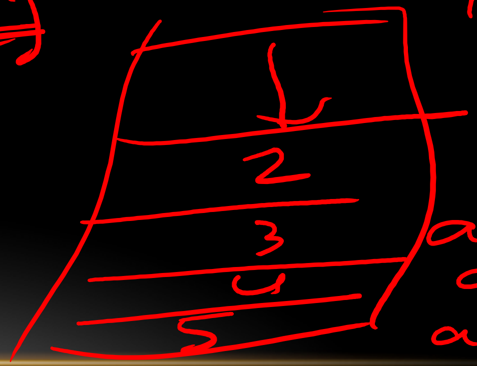
BX/BP SI/DI DISP

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

PA= DS: EA

DS

~~a[2]~~



a

a[0] [0]

a[0][1]

a[1] 6

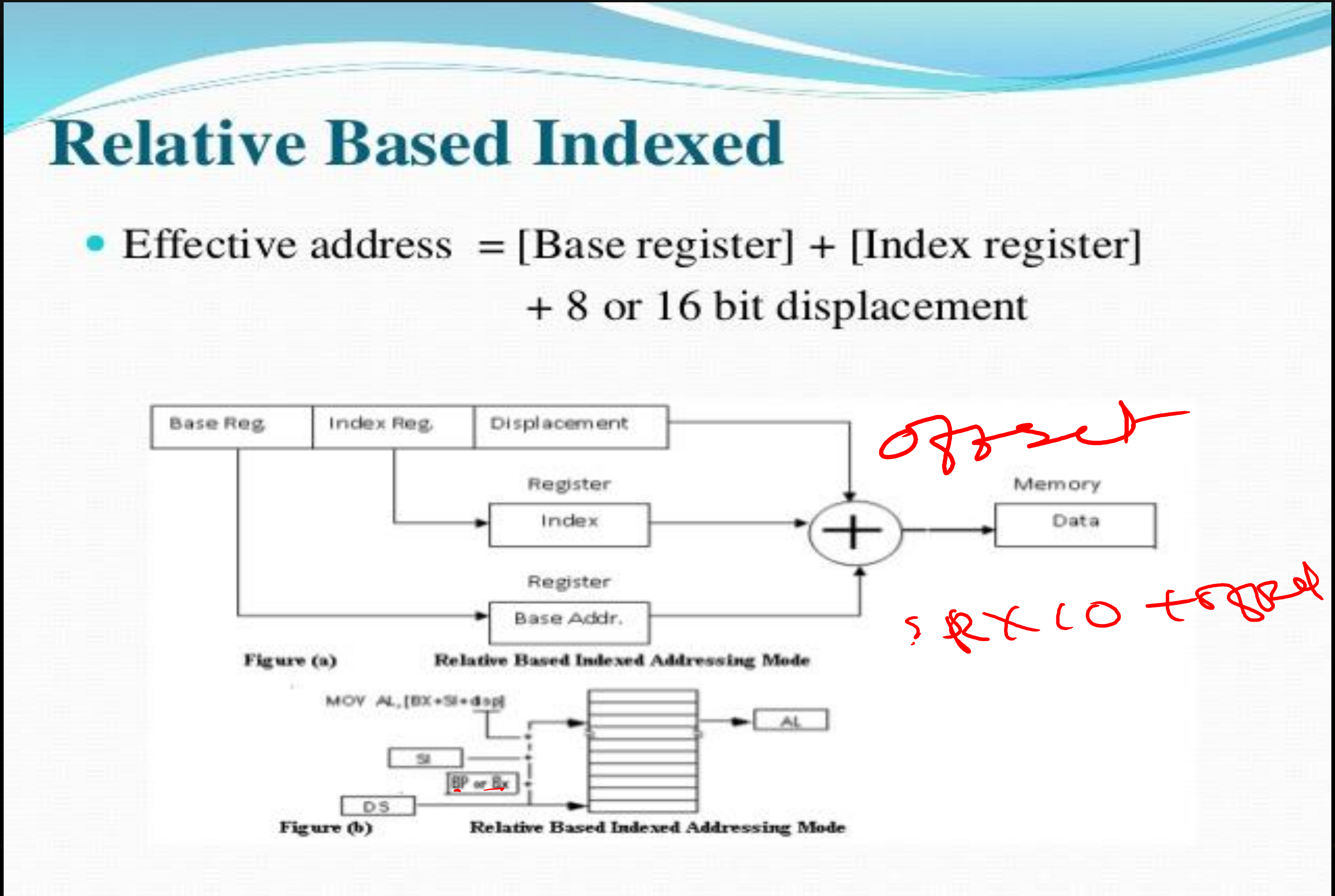
a[1][1]

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

offset

? BX (0 + offset

Figure (a) Relative Based Indexed Addressing Mode

Figure (b) Relative Based Indexed Addressing Mode

LEA SI, MSG1

LEA DI, MSG2

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

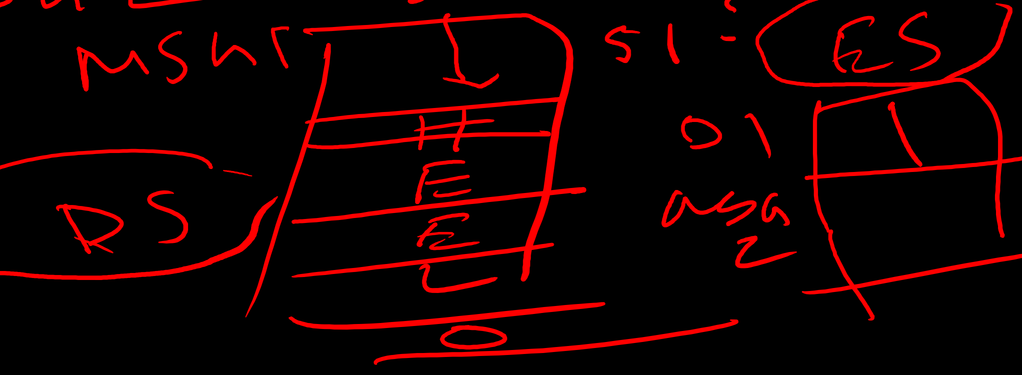
Data from 03575h moved to 068A7h

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

MSG1 DB 'HELLO$'

MSG2 DB ?



Source offset DS X 10 + SI

Dest " ES X 10 + DI

I/O ADDRESSING MODE

64KB port

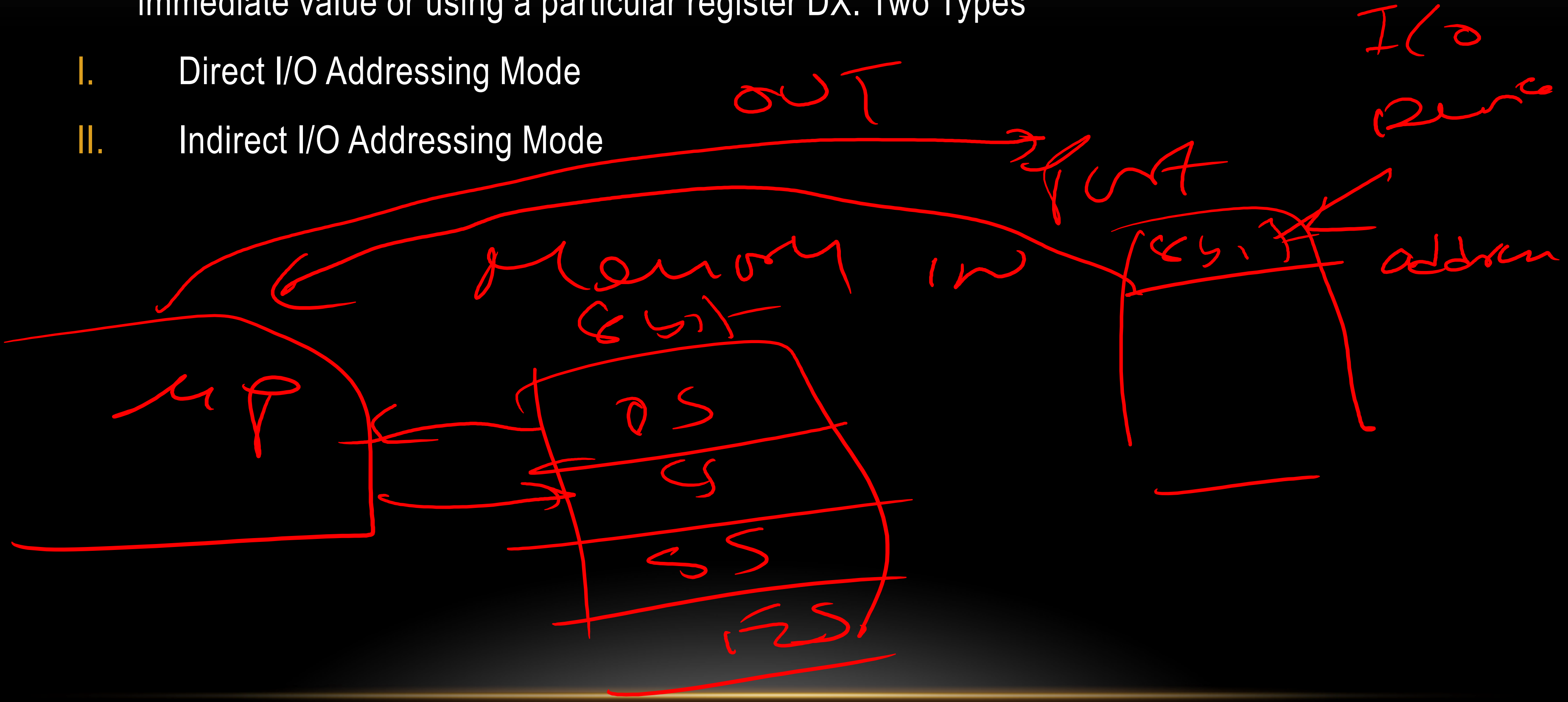
address 16 bit

2^{16} × 8 bit

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

8000H 80~~[illegible]~~H

~~8100~~

The port address is given as an immediate value.

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

PortA EQU 8000H

e.g. IN AX, Port_A (Word Operation)

IN AL, Port_A (Byte Operation)

IN AX, PortA word

OUT PortA, AX

IN AL, PortA

OUT PortA, AL

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)

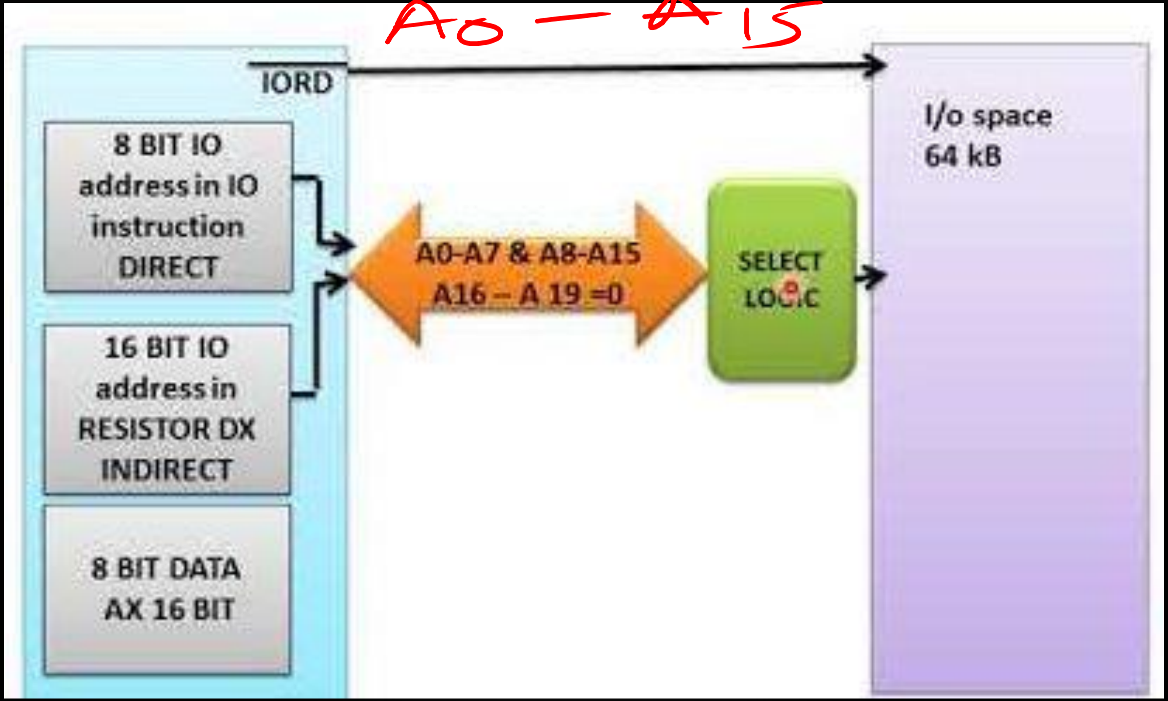
MOV DX, 8000H

OUT DX, AL

DIRECT AND INDIRECT MAPPING

A0 – A19 20bit

A0 – A15


IORD
8 BIT IO address in IO instruction DIRECT
16 BIT IO address in RESISTOR DX INDIRECT
8 BIT DATA AX 16 BIT
A0-A7 & A8-A15 A16 – A 19 =0
SELECT LOGIC
I/o space 64 kB

RELATIVE ADDRESSING MODE

JMP L

CS IP = IP + L

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.

offset = IP + L

PA = CS × 10 + offset

Syntax: <Opcode> <displacement>

IP →

e.g. JUMP L

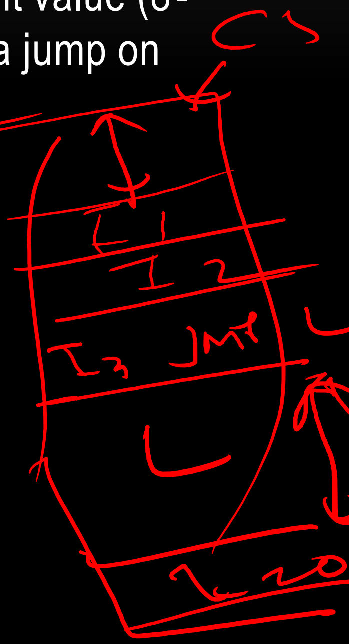
JNZ L

CALL SUM

JMP

RET

LOOP

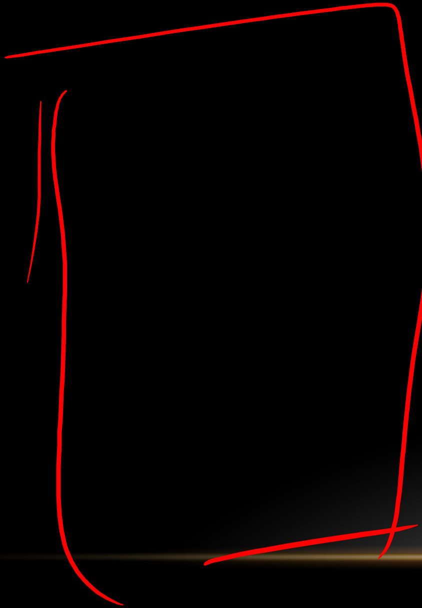


Two Types

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

Intersegment Jump: This modifies both the IP and CS contents

single CS



JMP is inside 64KB

jump beyond 64KB

IMPLIED ADDRESSING MODE

According to this mode the instruction carries no operand.

Syntax: <Opcode>

e.g. HLT, NOP, CLC, CLS

Halt — HLT

No operation — NOP

CLC — Clear CF

CLS — Clear Screen