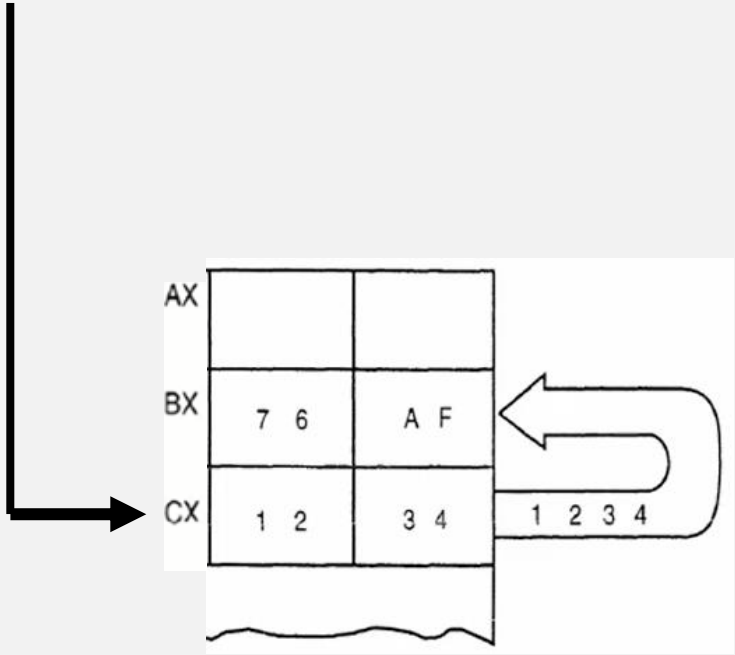
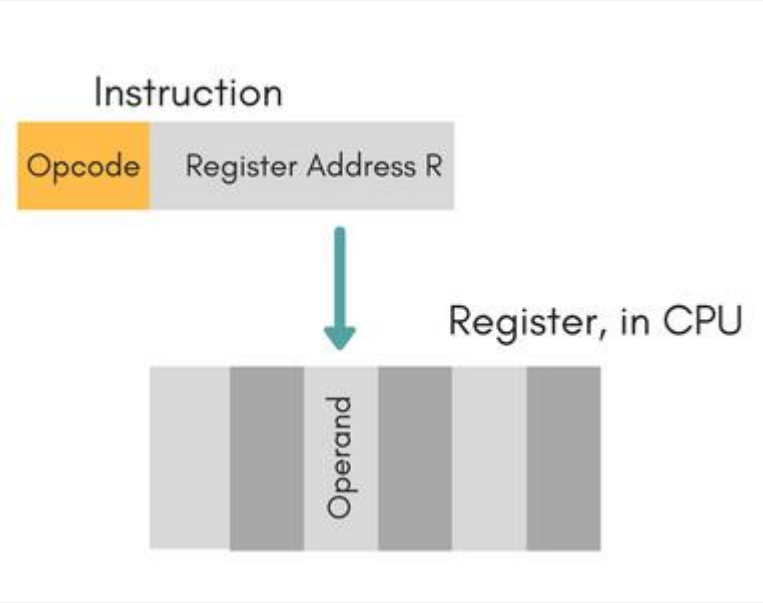
# *Data Addressing Mode*

REVIEW OF ASSEMBLY LANGUAGE

# REGISTER ADDRESSING

- MOV bl, al ; This instruction is used to copy the contents of register 8-bit al to bl
- MOV bx, cx ; This instruction is used to copy the contents of register 16-bit cx to bx



# *REGISTER ADDRESSING- CONTINUE*

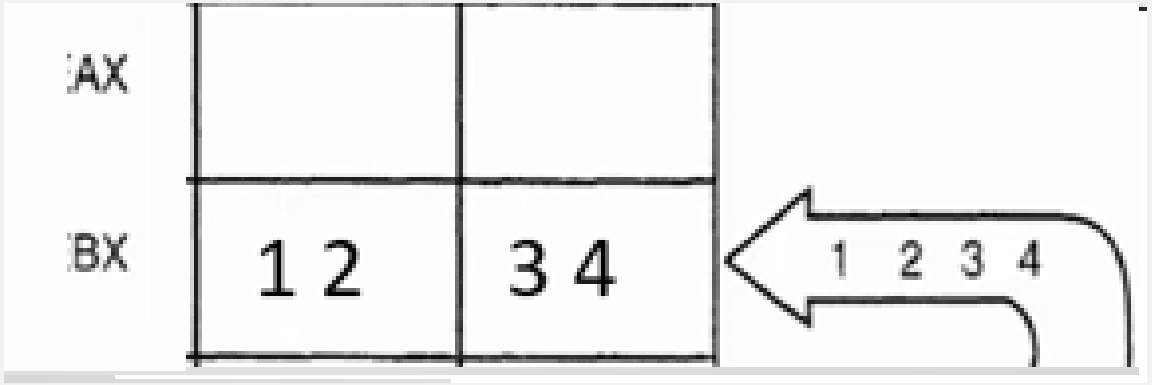
Note: The following must be observed in 8086 instructions:

- **Never** mix an ***8-bit*** register with ***16-bit***, it is not allowed in microprocessor.
- Code segment register (***CS***) and Instruction Pointer (***IP***) are never used as ***destination***.
- ***Segment*** with ***segment*** is not allowed.
- ***Memory*** with ***memory*** is not allowed.

# *IMMEDIATE ADDRESSING*

Transfer a constant data into a register or memory location

- **MOV BX, 1234h**




In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field.

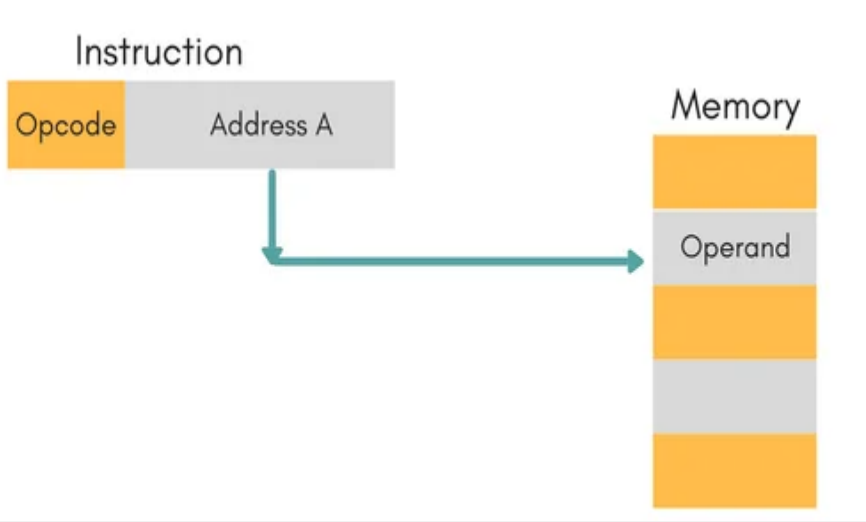
# *MEMORY ADDRESSING MODES*

***There are different forms of memory addressing modes***

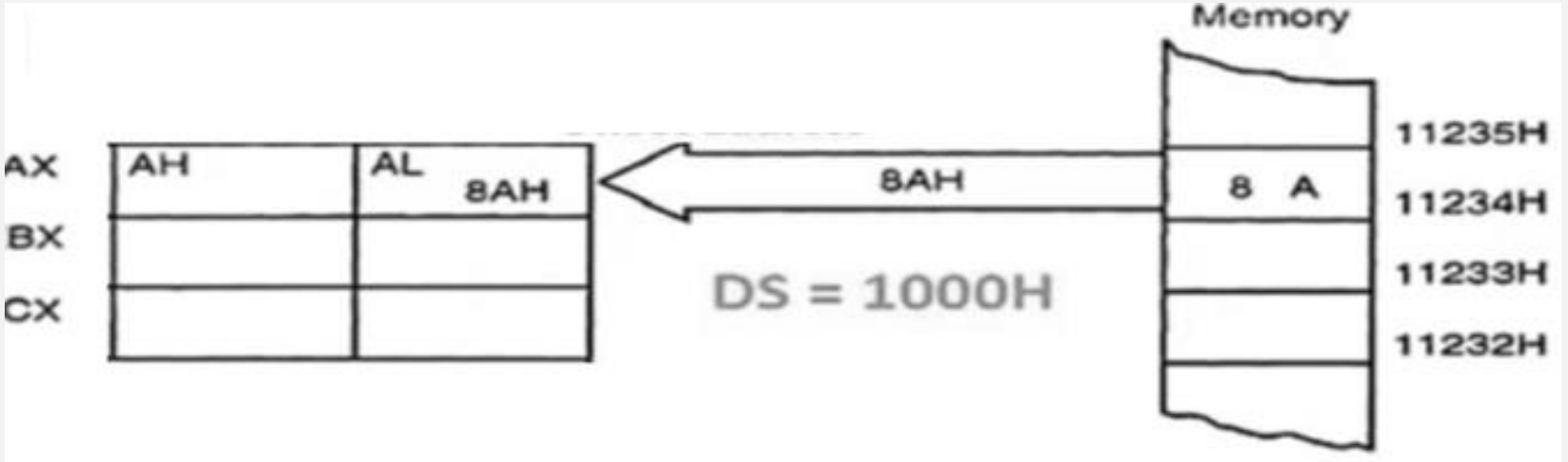
- Direct Addressing
- Register indirect addressing
- Based addressing
- Indexed addressing
- Based indexed addressing
- Based indexed with displacement

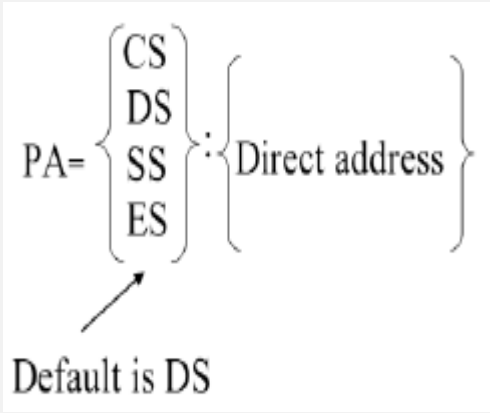
# MEMORY ADDRESSING -DIRECT ADDRESSING

- Direct addressing mode is similar to immediate addressing in that information is encoded directly into the instruction. However, in this case, the instruction opcode is followed by an **effective address**, instead of the **data**.




**MOV AL,[1234]** ➔ 1000*10 + 1234= 11234H

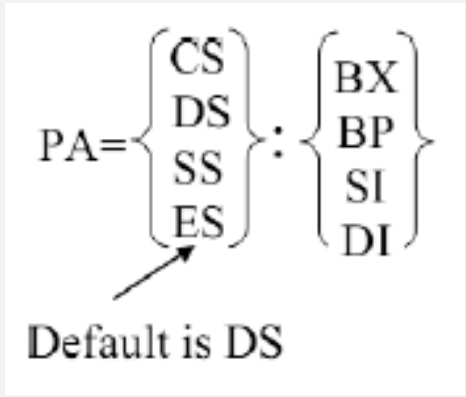


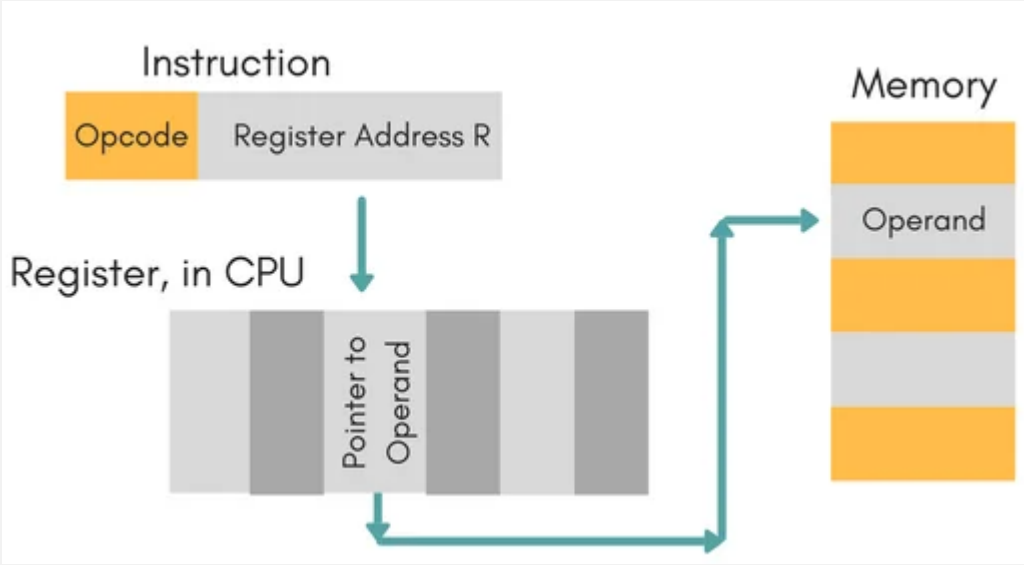
# *MEMORY ADDRESSING – REGISTER INDIRECT ADDRESSING*

- This mode is similar to the direct address except that the effective address held in any of the following register: BP, BX, SI, and DI As shown below:

```
MOV AL, [BX]
MOV AL, [BP]
MOV AL, [SI]
MOV AL, [DI]
```

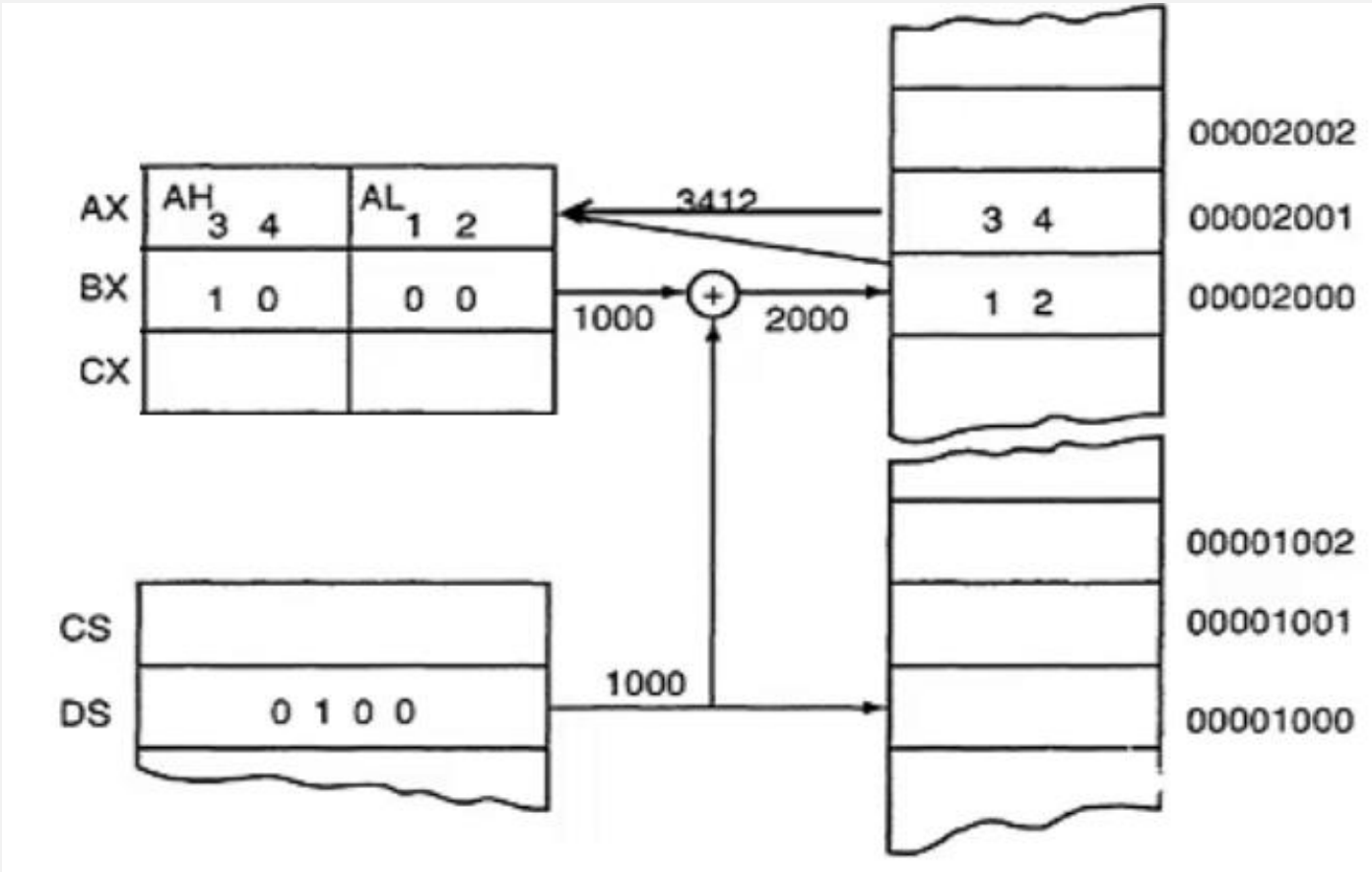



- The [**BX**], [**SI**], and [**DI**] modes use the **DS** segment by default. The [**BP**] addressing mode uses the stack segment (**SS**) by default. You can use the segment override prefix symbols if you wish to access data in different segments.



# *MEMORY ADDRESSING – REGISTER INDIRECT ADDRESSING*

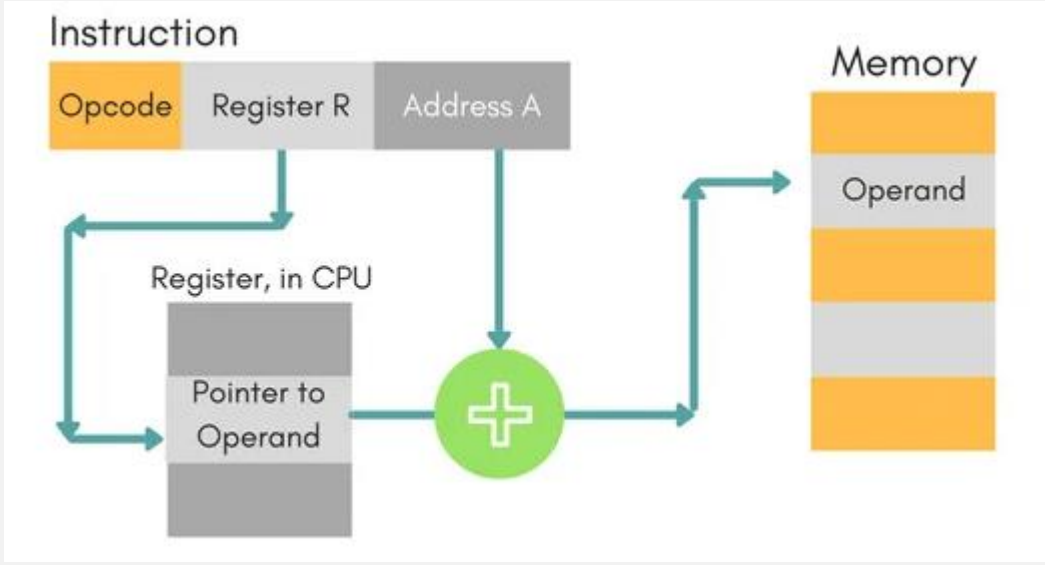
- ***EX:*** **MOV AX, DS:[BX]** When **BX=1000** and **DS =0100h**



# BASED ADDRESSING MODE

- In the based addressing mode, the effective address of the operand is obtained by ***adding*** a direct or indirect displacement to the contents of either base register ***BX*** or base pointer register ***BP***.
- If BP is used instead of **BX**, the calculation of the physical address is performed using the contents of the stack segment (SS) register instead of **DS**.

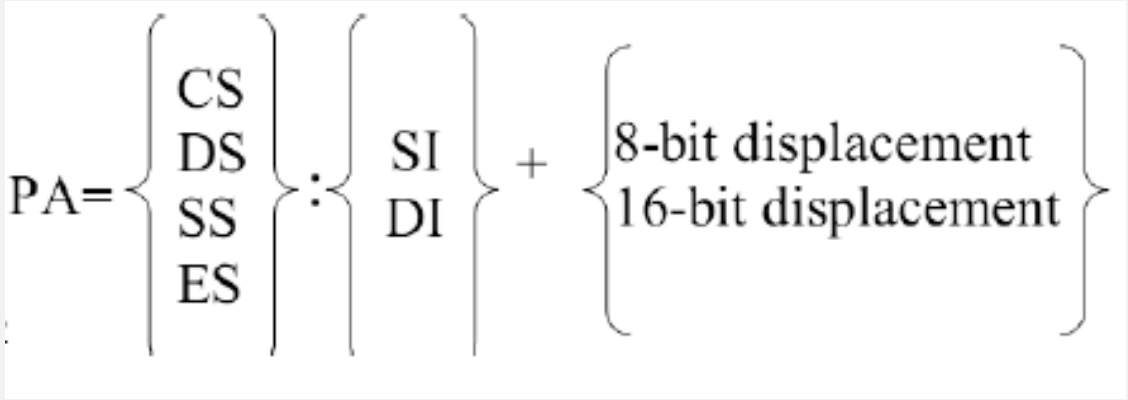
$$PA = \{CS, DS, SS, ES\} : \{BX, BP\} + \{8\text{-bit displacement}, 16\text{-bit displacement}\}$$



# INDEX ADDRESSING MODE

- In the Indexed addressing mode, the effective address of the operand is obtained by adding a direct or indirect displacement to the contents of either SI or DI register. The indexed addressing modes use the following syntax:

MOV AL, [SI]
MOV AL, [DI]
MOV AL, [SI+DISP]
MOV AL, [DI+DISP]

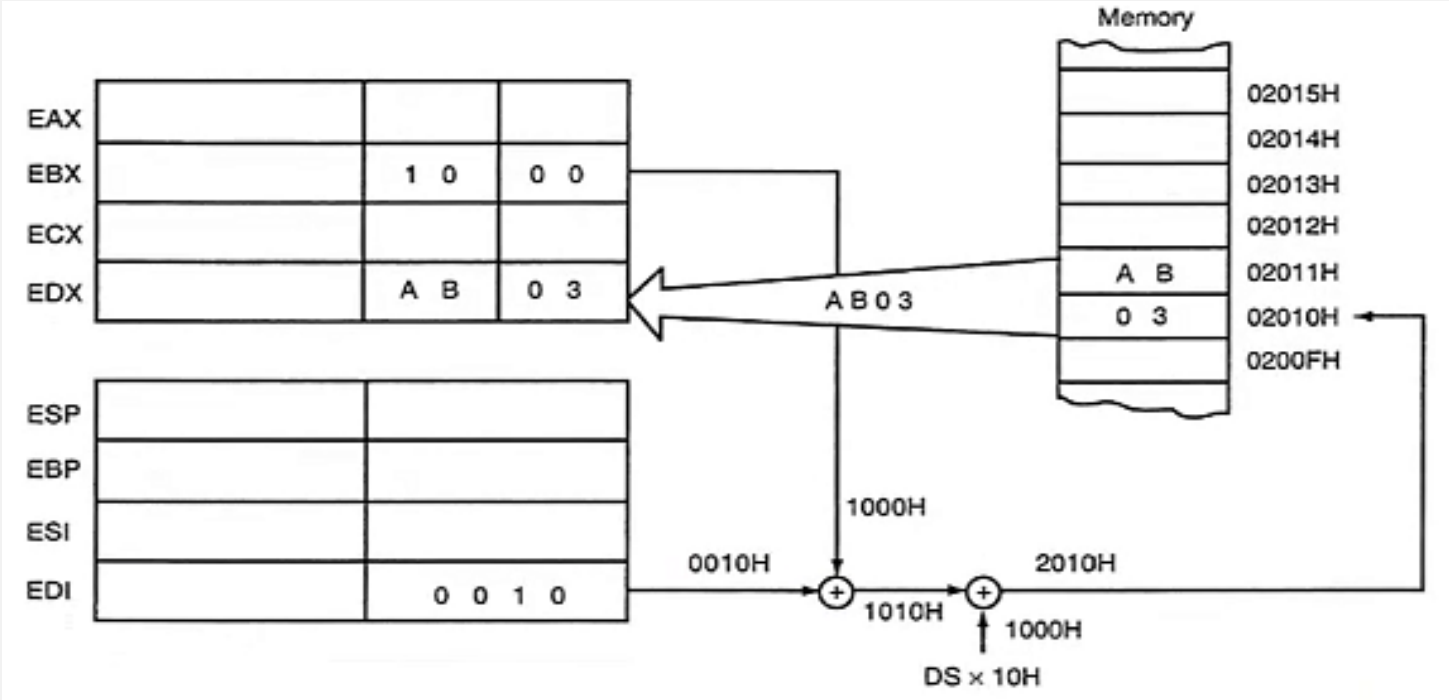


# BASED INDEXED ADDRESSING MODE

- Combining the based addressing mode and the indexed addressing mode results in a new, more powerful mode known as based-indexed addressing mode.

MOV DX, [BX+DI]
When BX=1000H, DI=0010H
and DS=0100H

$$PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \end{Bmatrix} : \begin{Bmatrix} SI \\ DI \end{Bmatrix} + \begin{Bmatrix} \text{8 bit displacement} \\ \text{16 bit displacement} \end{Bmatrix}$$



# *BASED INDEXED ADDRESSING MODE WITH DISPLACEMENT*

- It is a slight modification of Base/index addressing modes with the addition of an 8 bit/ 16 bit constant. The following are some examples of these addressing mode

```
MOV     AL, DISP[BX][SI]
MOV     AL, DISP[BX+DI]
MOV     AL, [BP+SI+DISP]
MOV     AL, [BP][DI][DISP]
```

# *DATA TYPES AND DATA DEFINITIONS*

- A data definition statement sets aside storage in memory for a variable, with an optional name. A data definition has the following syntax:

Variable name directive initializer data

Var1 DW 2345

Var2 DB ‘Hello’

Var3 DB 0FFh, 00h, 11100000b

| 8-bit |
| --- |
| 8-bit |
| 45 |
| 23 |

- A question mark (?) initializer leaves the variable uninitialized, implying it will be assigned a value at runtime:

var4 DB ?

# *DATA TYPES AND DATA DEFINITIONS*

- DD (Define DoubleWord) and DQ (Define QuadWord)

Variable name DD data → **Assign 4-byte memory location**

Variable name DQ data → **Assign 8-byte memory location**

- **DUP (Duplicate) Directive**
  - The DUP operator allocates storage for multiple data items, using a constant expression as a counter. It is particularly useful when allocating space for a string or array, and can be used with initialized or uninitialized data:

Variable name DB Number of Duplications DUP (data)

Variable1 DB 5Ah, 5Ah, 5Ah, 5Ah, 5Ah, 5Ah, 5Ah

Variable2 DB 7 DUP (5Ah) Variable2 DB 7 DUP (?)

# *DATA TYPES AND DATA DEFINITIONS*

## • EQU (Equate) Directive

EQU statement Name EQU Data

- The **EQU directive** only tells the **assembler** to substitute a value for a symbol or label, and doesn't involve any type of ROM or RAM.
- **EQU directives** are typically placed at the beginning of an **assembly** program.
- It is usually used for the counter in the code statement

```
Count EQU 5
MOV CX, Count
```

# DATA TYPES AND DATA DEFINITIONS -CONTINUE

- ORG directive is used to indicate the beginning of the offset address. The number that comes after ORG should be in Hex.

ORG (Origin) directive

ORG 100h

# *EXAMPLES*

**Ex1: Write an assembly language to perform the following tasks:**

1. Initialize AX and SI registers with the values 1520H and 0300H respectively.
2. Save the value 3040h at the data segment memory location addressed by SI
3. Add the word contents at the data segment memory location addressed by SI to AX with the sum stored at the AX register.

**Answer:**

```
;1
mov ax, 1520h
mov SI, 0300h
;2
mov [si], 3040h
;3
ADD ax,[si]
Ret
```

# EXAMPLES

- ***Ex2: Write an assembly language to perform the following tasks:***

  a. Initialize AL, BL, CL and DL registers with the values 10h, 20h, 30h and 40h.respectively.

  b. Copy the contents of AL, BL, CL and DL registers into BH, CH, DH and AH registers respectively.

  c. Swap between the contents of AX and BX registers.

  d. Copy the contents of AX register into the data segment memory location addressed by 0200h then Copy the contents of BX and CX registers at the consecutive offset addresses.

  e. Copy the contents of DX register into the stack segment memory location addressed by 0100h then Copy the data 4433h and 2211h at the consecutive offset addresses.

**Answer:**

```
mov al, 10h
mov bl, 20h
mov cl, 30h
mov dl, 40h
;b
mov bh, al
mov ch, bl
mov dh, cl
mov ah, dl
;c
 xchg ax, bx
;d
 mov [200h], ax
 mov [202h], bx
 mov [204h], cx
;e
 mov SS: [100h], dx
 mov SS: [102h], 4433h
 mov SS: [104h], 2211h
Ret
```

# EXAMPLES

- ***Ex3: Write an assembly language to perform the following tasks:***

a. Copy the string data ‘NO’ into AX register. Initialize the source index register (SI) with the initial value 0200h and the base register (BP) with initial value 0100h **then** copy the contents of AX register into the stack segment **memory locations** addressed by SI+BP+20h.

b. Initialize the destination index register (DI) with initial value 0300h then copy the string data ‘HELLO’ into the extra data segment memory locations addressed by DI+100h.

**Answer:**

```
;a
MOV ax, 'NO'
MOV si,  200h
MOV bp, 100h
MOV ss: [si+bp+20h], ax

;b
MOV di, 300h
MOV es: [di+100h], 'EH'
MOV es: [di+102h], 'LL'
MOV es: [di+104h], 'O'
Ret
```

# *EXAMPLES*

- ***Ex4: Write an assembly language to perform the following tasks:***
  a. Initialize the AX, BX and Di registers with the immediate values 1020h, 3040h and 0200h respectively.

  b. One’s complement the low byte of AX register.

  c. Tow’s complement the high byte of the BX register.

  d. Save the high byte of AX and the low byte of BX at the data segment memory locations addressed by DI respectively.

# ANSWER

***Answer4:***

```
;a
mov ax, 1020h
mov bx, 3040h
mov di, 200h
;b
 not al          ;20h→ 0010 0000b  → 1101 1111→ DF
;c
neg bh           ;30h→ 0011 0000b  → 1101 0000 = D0
;d
mov [di], ah
mov [di + 1], bl
Ret
```

# *EXAMPLES*

- ***Ex5: Write an assembly language to perform the following tasks:***

a. Loading the registers AL & BL by the values 0AH & 26H respectively.

b. Calculate the expression $\boldsymbol{CX = AX^2 + BX}$.

c. Store the results at location 0100h.

# ANSWER

***Answer5:***

```
;a
mov ax, 0000h
mov ds, ax
mov al, 0Ah
mov bx, 26h
;b
mul al
add ax,bx
mov cx,ax
;c
mov [0100h], cx
Ret
```

# *EXAMPLES*

**EX6: Write an ALP to transfer a block of 256 bytes stored at locations starting at 34000H to locations starting at 36000H.**

```
        MOV AX , 3000H
        MOV DS , AX                 ; DS=3000*10H=30000
        MOV BX , 0000H              ; Set the offset address
        MOV CX , 256
NEXT:   MOV AL , [BX + 4000H]
        MOV [BX + 6000H] , AL
        INC BX
        Loop Next
        HLT
```

# THANK YOU