8086 Addressing Modes

The way of specifying data to be operated by an instruction is known as **addressing modes.** In the other word addressing modes refer to the different methods of addressing the operands. Depending upon the data types used in the instruction and the memory addressing modes, any instruction may belong to one or more addressing modes.

Instructions are operations performed by the CPU. An instruction is a statement that is executed at runtime. In 8086 instruction statement can consist of four parts:

- Label (optional)
- Instruction (required)
- Operands (instruction specific)
- Comment (optional)

```
[ Label:]        Instruction        Operands        [; Comment ]
```

Operands are entities operated upon by the instruction. An 8086 instruction can have zero to two operands. For instructions with two operands, the first (lefthand) operand is the source operand, and the second (righthand) operand is the destination operand (that is, source->destination). Also, operands are separated by commas (,) :

```
Mov   ax, bx
```

Addresses are the locations in memory of specified data.

Machine code : may be one , two , three or four bytes in length. The first byte is actual operation called an opcode (is short for 'Operation Code') that tells the processor what should be done, and any other bytes that present are operand refrenced an immediate value , a register , or a memory location.

Types of Addressing Modes:

1.Immediate addressing mode: The destination field in the first operand defines the length of the data and may be a register or a memory location (the first operand is never an immediate value), and the second operand is the data itself (appears in the form of successive byte or bytes).

Example: MOV AX , 8705 h

In the above example, 8705 h is the immediate data. The immediate data may be 8-bit or 16-bit in size.

```
MOV  byteptr [6100 h] , 0F h
MOV  wordptr [6100 h] , 0F h
```

2. Register addressing mode: In the register addressing mode, the data is stored in a register and it is referred using the particular register (the operands are registers). All the registers, except IP, may be used in this mode.

```
Example:   ADD   BL  ,  AL
           MOV   SI   ,  CX
```

3. **Direct addressing mode:** In this mode , one operands references a memory location and other operand references a register.

```
Example:   MOV    AX  ,  [6107 h]
           SUB     [6107 h]  ,  CL
```

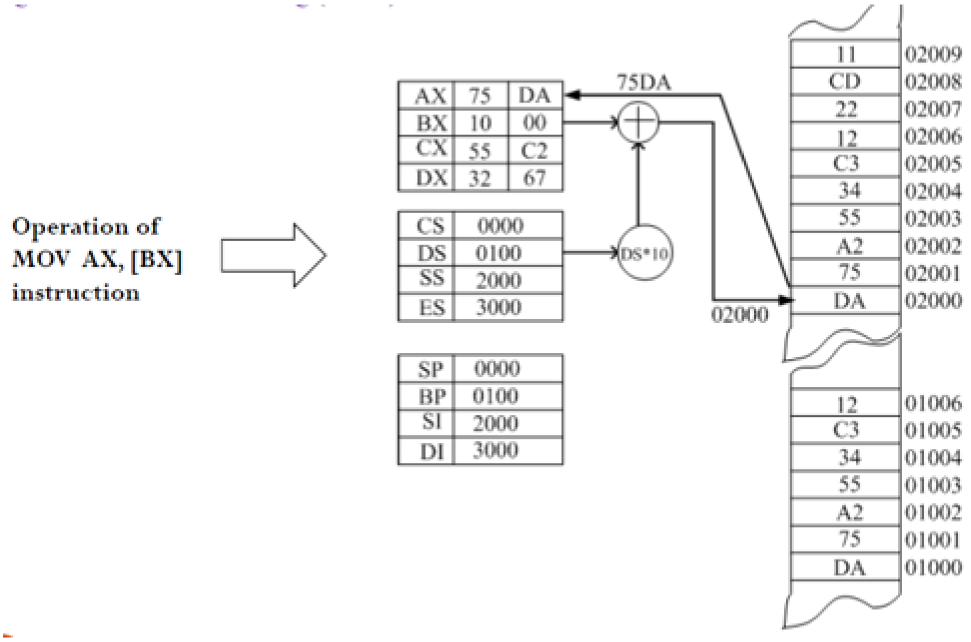
Here, data resides in a memory location in the data segment, whose physical address may be computed using 6107 h as the offset address and content of DS as segment address. The physical address , here , is : physical address = DS * 10 h + 6107 h.

4. Indirect addressing mode: When the one operand references a register. the other operand contains an indirect address as follows :

***Note*.** By default DS segment register is used for all modes except those with BP register , for these SS segment register is used.

a. Base : Here memory is [BX] or [BP].

```
Example:    MOV AX , [BX]        ;  segment is DS
            SBB  [BP] ,CX        ;  segment is SS
            INC   wordptr[BP]    ;  segment is SS
```

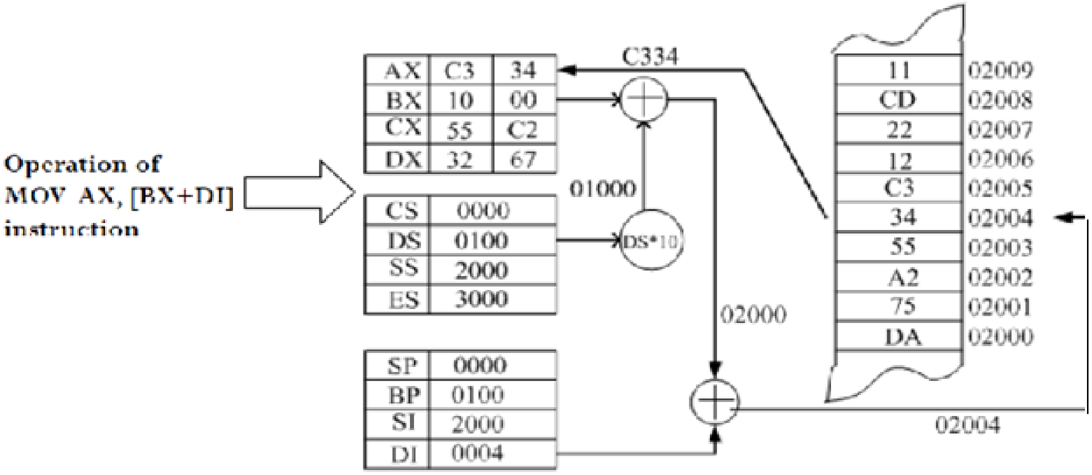


b . Index : Here memory is [SI] or [DI]. Here segment is DS.

```
Example:   AND  [SI] , DX
           SUB   BX , [DI]
           DEC  wordptr[DI]
```

c. Base index : Here memory is [BX+SI] or [BX+DI] or [BP+ SI] or [BP+ DI].

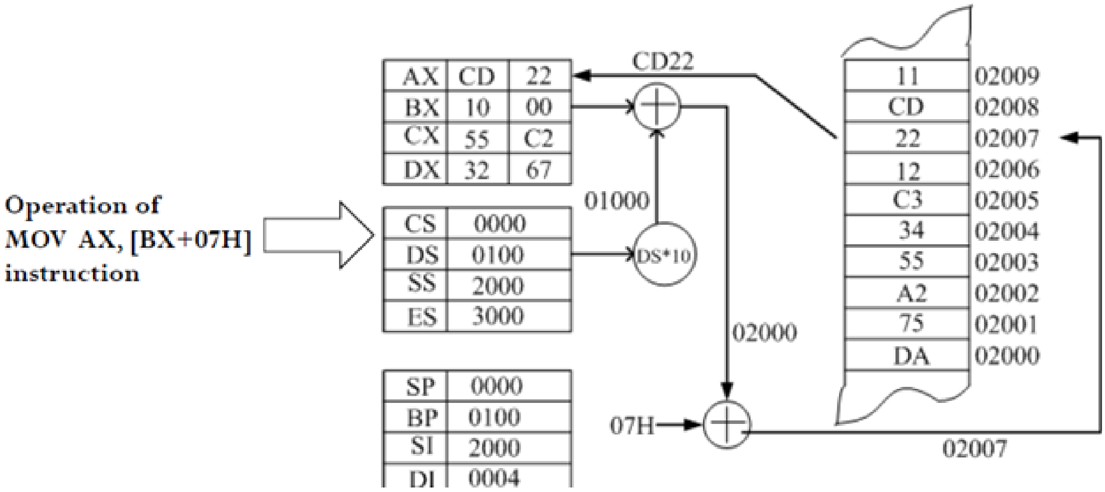
```
Example:    MOV  AL , [BX+SI]        ;  segment is DS
            ADC  [BX+DI] , SP        ;  segment is DS
            SBB   [BP+DI] , CL       ;  segment is SS
            INC   wordptr[BP+SI]     ;  segment is SS
            MOV  AX , [BX+DI]        ;  segment is DS
```



d. Base with displacement : Here memory is [BX + displacement] or [BP + displacement].

Note: Displacement is 8 bit or 16 bit signed value, so it can be both positive or negative.

```
Example:    DEC byteptr [BX+0825 h]        ;  segment is DS
             SBB  AX , [BP+ 9040 h]        ;  segment is SS
             XOR  CX,[BX-9000 h]           ;  segment is DS
             OR     wordptr[BP-7000 h],DX  ;  segment is SS
             MOV  AX ,[BX+07 h]            ;  segment is  DS
```



e. **Index with displacement:** Here memory is [SI+displacement] or [DI+displacement]. Here segment is DS.

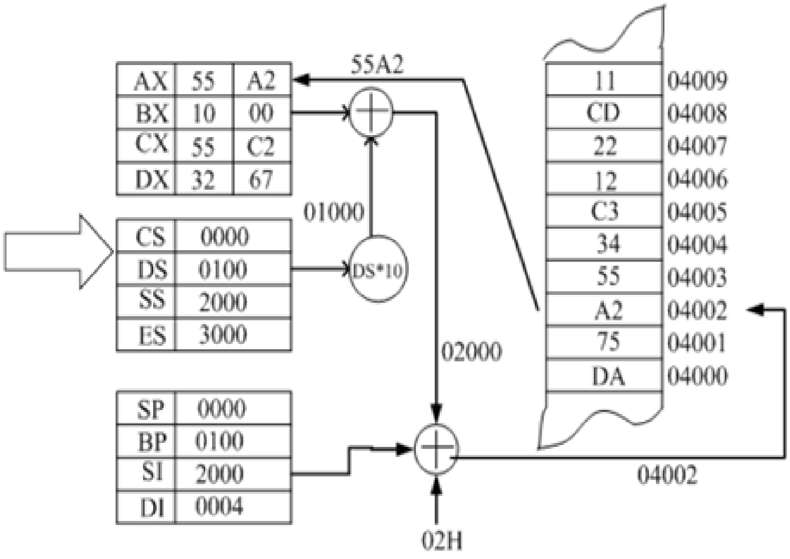
```
Example:    SUB   [SI+33 h] , CX
            ADC   BX , [DI -  F122 h]
```

f. Base index with displacement: Here memory is

[BX+SI+displacement] or [BX+DI+displacement] or [BP+SI+displacement] or [BP+DI+displacement].

```
Example:   ADC AX , [BX+SI-01]             ;  segment is DS
           MOV  DH , [BX+DI+15 h],         ;  segment is DS
           OR   [BP+SI-1F h] , CX          ;  segment is SS
           INC   wordptr[BP+DI+82 h]       ;  segment is SS
           MOV AX , [BX+SI+02 h ]          ;  segment is  DS
```

Operation of MOV AX, [BX+SI+02H] instruction



Note. Only the registers (BX, SI, DI, BP) can be used inside square brackets (as memory pointers).

5. Stack addressing mode:

```
Example:     PUSH    AX
             POP    [DI]
```

6. String addressing mode:

Example : MOVSB , MOVSW , LODSB , LODSW , STOSB , STOSW , CMPSB , CMPSW , SCASB , SCASW .

7. Input output addressing mode:

```
Example:    IN      AL ,  7
             IN      AX , DX
             OUT   DX , AX
```

8.Implied addressing mode:

Example : AAA, AAD, AAM, AAS , DAA, DAS, XLATB .