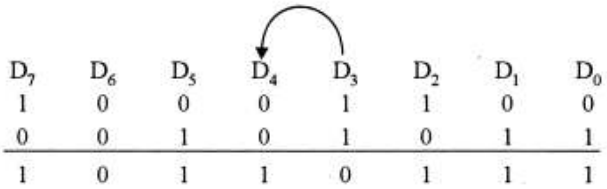
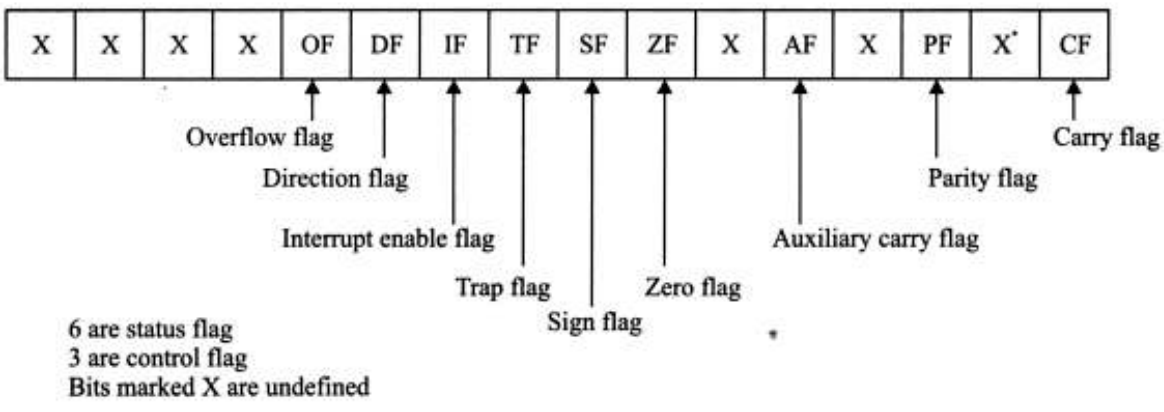
The Flags Register

It is a 16-bit register, also called Program Status Word (PSW). Seven bits remain unused while the rest nine are used. **Six** are status flags and **three** are control flags.

- The control flags can be set/reset by the programmer.
  1. **DF (Direction Flag) :** controls the direction of operation of string instructions. (DF=0 Ascending order DF=1 Descending order)
  2. **IF (Interrupt Flag):** controls the interrupt operation in 8086µP. (IF=0 Disable interrupt IF=1 Enable interrupt)
  3. **TF (Trap Flag):** controls the operation of the microprocessor. (TF=0 Normal operation TF=1 Single Step operation)
- The status flags are set/reset depending on the results of some arithmetic or logical operations during program execution.
  1. **CF (Carry Flag)** is set (CF=1) if there is a carry out of the MSB position resulting from an addition operation or subtraction.
  2. **AF (Auxiliary Carry Flag)** AF is set if there is a carry out of bit 3 resulting from an addition operation.



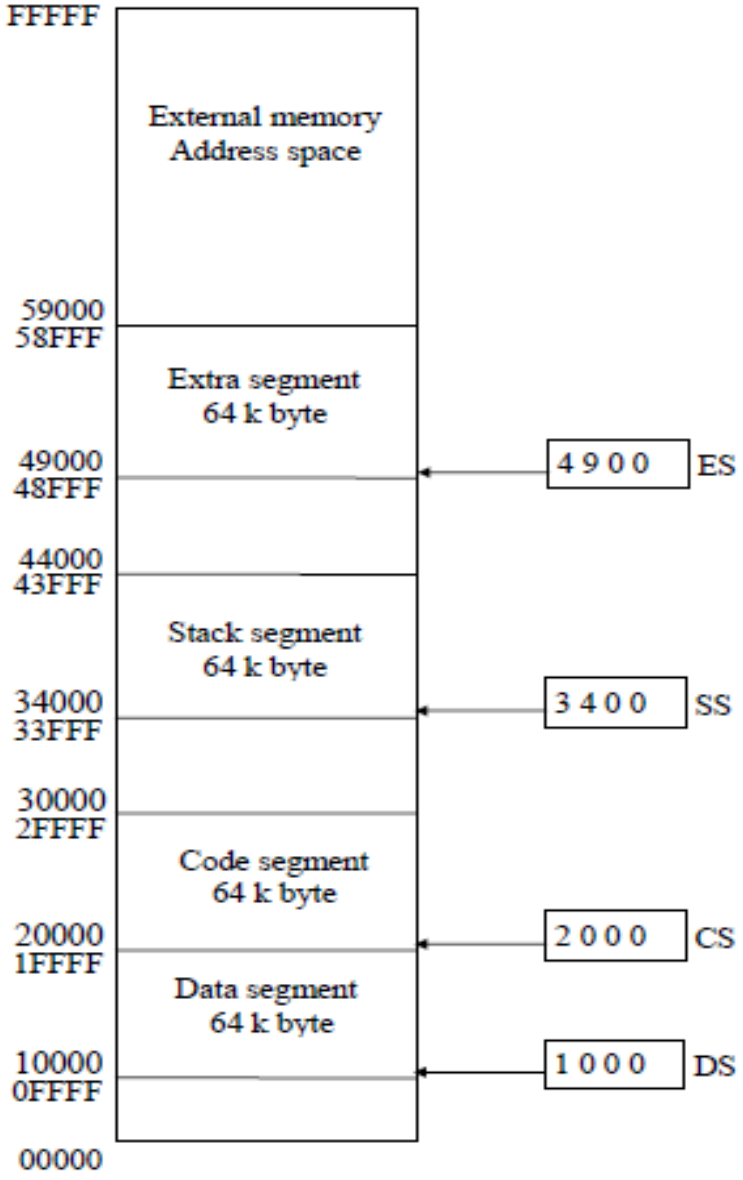
  3. **SF (Sign Flag)** set to **1** when result is negative. When result is positive it is set to **0**.
  4. **ZF (Zero Flag)** is set (ZF=1) when result of an arithmetic or logical operation is zero. For non-zero result this flag is reset (ZF=0).
  5. **PF (Parity Flag)** this flag is set to **1** when there is even number of one bits in result, and to **0** when there is odd number of one bits.
  6. **OF (Overflow Flag)** set to **1** when there is a **signed overflow**. For example, when you multiply **FFH by 11H** (result is not one byte).



Memory Organization

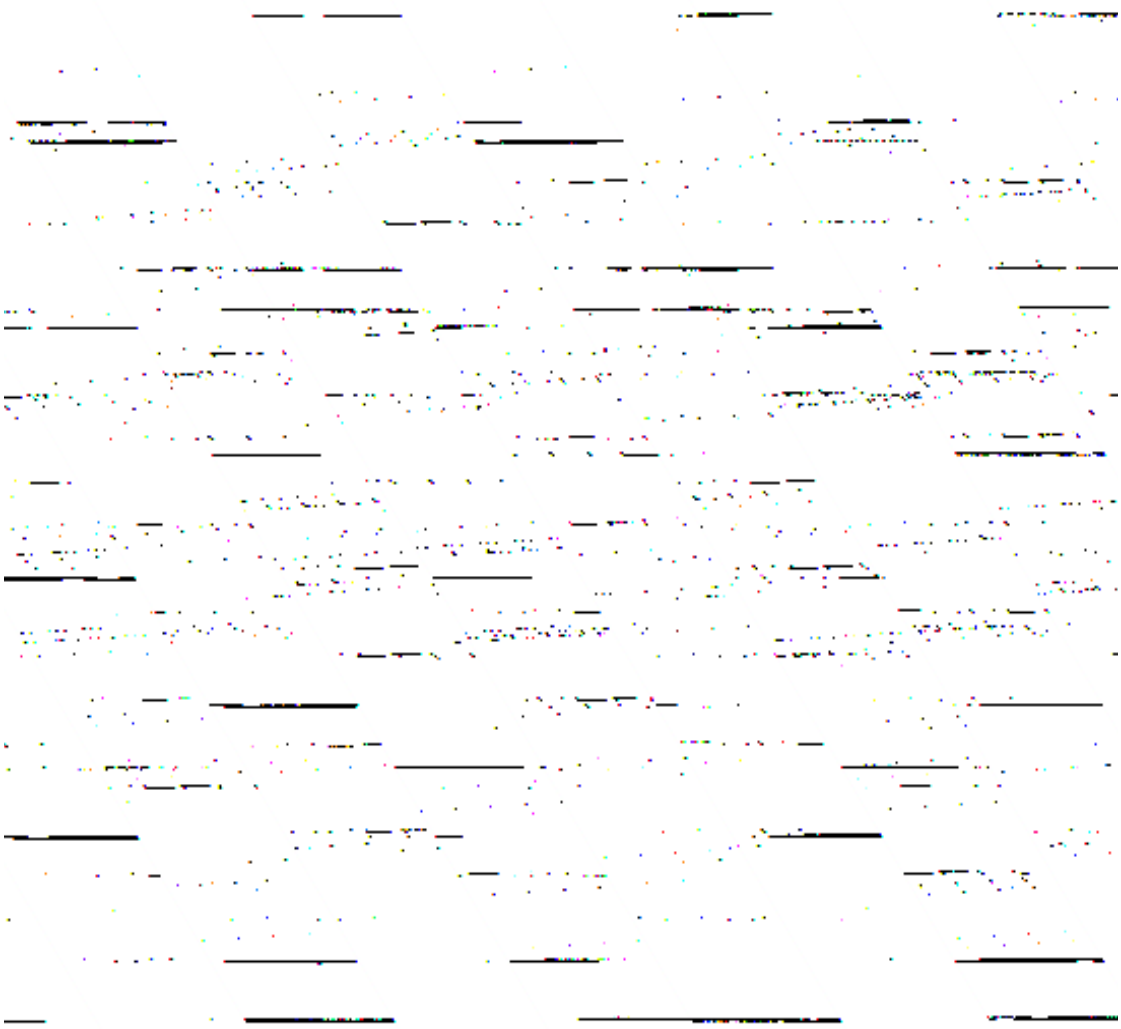
8086/8088µP, via its 20-bit address bus, can address 2^{20} = 1,048,576 bytes or (1 Mbyte) of memory locations. Thus the memory space of 8086µP is 1,048,576 bytes or 524,288 words. The memory map of 8086µP as shown, where the whole memory space starting from 00000H to FFFFFH.

The 8086µP operate in the **Real mode memory** addressing. **Real mode operation** allows the microprocessor to address only the first 1 Mbyte of memory. Even though the 8086 has a 1 Mbyte address space, not all this memory is active at one time. Actually, the 1 Mbytes of memory are partitioned into 64 Kbyte (65,536) **segments.** The 8086µP allows four memory segments. The figure shows these memory segments.



Software Model of 8086µP

What is important to the programmer is to know the various registers within the 8086µP and to understand their purpose, functions, operating capabilities, and limitations. The figure below illustrates the software architecture of the 8086µP.



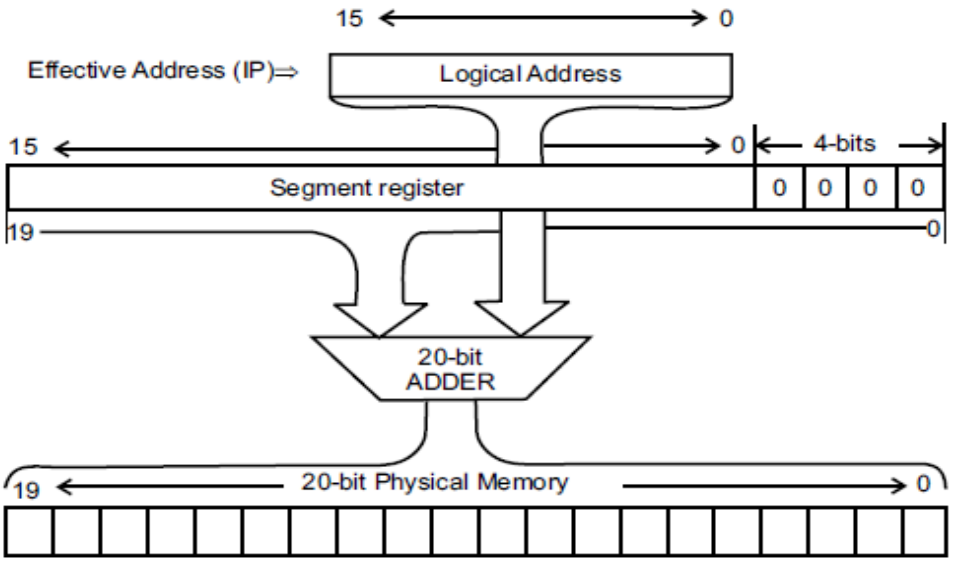
Physical Address Generation

Logical address, segment address and physical address are explained as follows:

- The logical address, also known as the effective address or offset address is contained in the registers IP, BP, SP, BX, SI or DI.
- The 16-bit content of one of the four segment registers (CS, DS, ES, SS) is known as the base segment address.
- Offset and base segment address are combined to form a 20-bit physical address (also called real address) that is used to access the memory.

The 20-bit physical address is generated by combining the offset (residing in BX, BP, IP, SP, SI or DI) and the content of one of the segment registers CS, DS, ES or SS. The process of combination is as follows:
The content of the segment register is internally appended with 0H (0000B) on its right most end to form a 20-bit memory address—this 20-bit address points to the start of the segment. The offset is then added to the above to get the physical address.

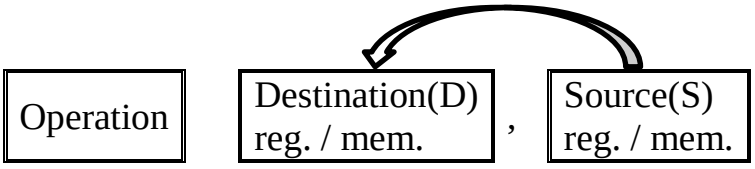
Physical Address = Segment Register : Offset (Effective Address)
= (Segment Register)0H + Offset (Effective Address)



For example if (DS)=2000H and the Logical address is 1234H then the physical address is: PA = (2000)*10 + 1234H = 20000H + 1234H = 21234H

Addressing Modes of 8086µP

The general form of an Assembly language instruction is :



An instruction consists of an op-code and operands. The operand may reside in the accumulator, or in a general purpose register or in a memory location.

Examples:

MOV AX , BX ; copy the content of BX to AX**.**
ADD AL , 12H ; add 12H to AL and store the result in AL.

An addressing mode is a method of specifying an operand. The following are the different addressing modes of 8086µP:

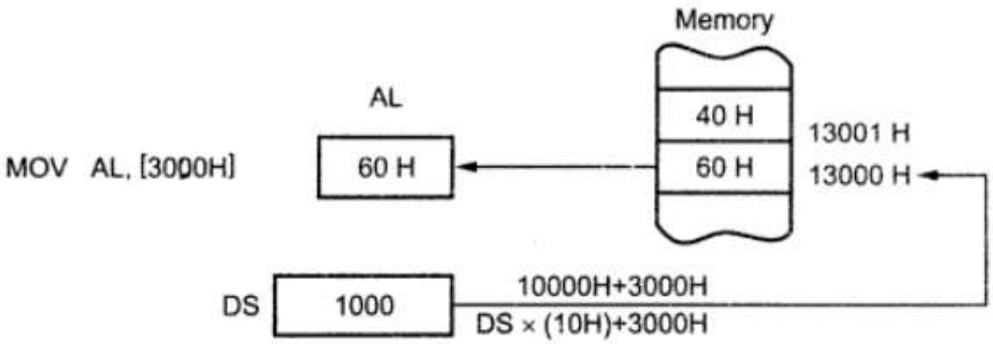
1. **Register Addressing Mode:** The operands both are registers (8-bit or 16-bit). Some examples are:
   MOV AX , BX
   ADD AL , BH
2. **Immediate Addressing Mode:** The source can be either 8-bit or 16-bit number and the destination is a register . Some examples are:
   MOV AL , 83H
   ADD BX , 1284H
3. **Memory Addressing Mode:** One of the operands is a memory location. The different memory addressing modes are:

   a) **Direct Addressing Mode:** The effective address of the specified memory location is given directly between [ ] brackets, for example: (DS)=1000H
   MOV AL , [3000H]

$$PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : \{Direct\ address\}$$

Note: The default segment register is DS

PA = (DS)0H + 3000H = 10000H + 3000H = 13000H

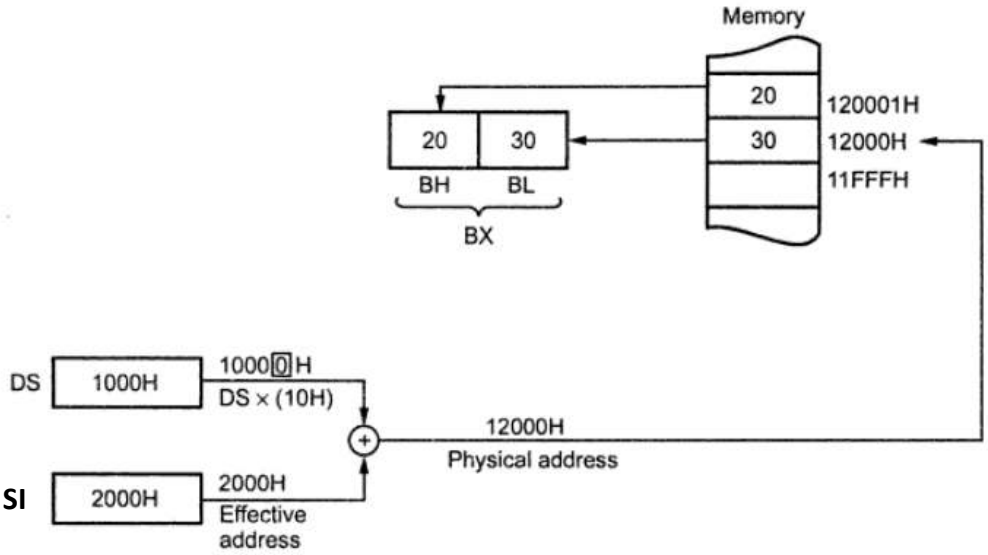


b) Register Indirect Addressing Mode: The effective address is located in any of the following registers : BP, BX , SI and DI.

$$\left.\begin{array}{ll} \mathbf{EA} = \left[Base \quad \begin{matrix} BX \\ BP \end{matrix}\right] & \textbf{Based Addressing} \\ \mathbf{EA} = \left[Index \quad \begin{matrix} SI \\ DI \end{matrix}\right] & \textbf{Indexed Addressing} \end{array}\right\} PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \\ SI \\ DI \end{Bmatrix}$$

For example if (SI)=2000H, and (DS)=1000H, then the instruction **MOV BX , [SI]** copies the word 2030H from locations 12000H and 12001H to BX.

PA=(DS)*10H + [2000H] = 10000H + 2000H = 12000H



c) Register Relative Addressing Mode: The effective address can be found as follows:

- **Based with displacement (Based-Relative) addressing mode.**

$$EA = \begin{matrix} BX \\ BP \end{matrix} + \textbf{displacement } (\mathbf{8-bit \text{ or } 16-bit})$$

- **Indexed with displacement(Indexed-Relative) addressing mode.**

$$\mathbf{EA} = \begin{matrix} SI \\ DI \end{matrix} + \textbf{displacement } (\mathbf{8-bit \text{ or } 16-bit})$$

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

For example if (BP)=1000H , (SS)=0100H , (AL)=FFH and a displacement of 100H then the instruction **MOV [BP+100H] , AL** copies the content of AL into memory location 02100H.

PA = (0100)*10H + [1000H + 100H] = 02100H

Note: the diagram is left as a H.W for the student

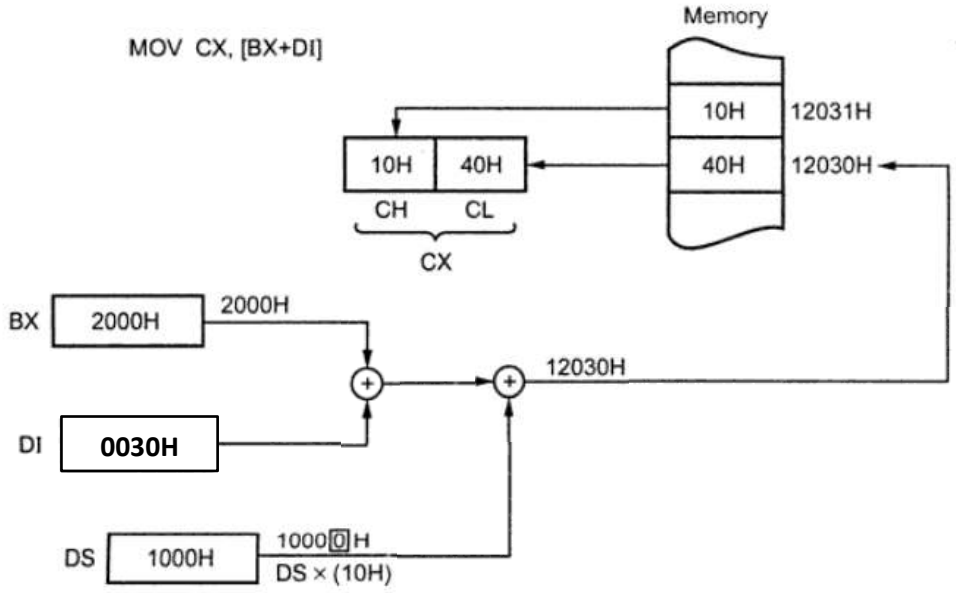
d) Based-Indexed Addressing Mode : The effective address can be found as follows:

$$\text{EA} = \left[Base \begin{matrix} BX \\ BP \end{matrix} + Index \begin{matrix} SI \\ DI \end{matrix}\right]$$

$$PA = \begin{Bmatrix} CS \\ DS \\ SS \\ ES \end{Bmatrix} : \begin{Bmatrix} BX \\ BP \end{Bmatrix} + \begin{Bmatrix} SI \\ DI \end{Bmatrix}$$

For example if (BX) = 2000H , (DI) = 0030H and (DS) = 1000H then the instruction **MOV CX , [BX + DI]** load the register CX with 1040H form memory locations 12030H and 12031H.

PA = 10000 + [2000H + 0030H] = 12030H



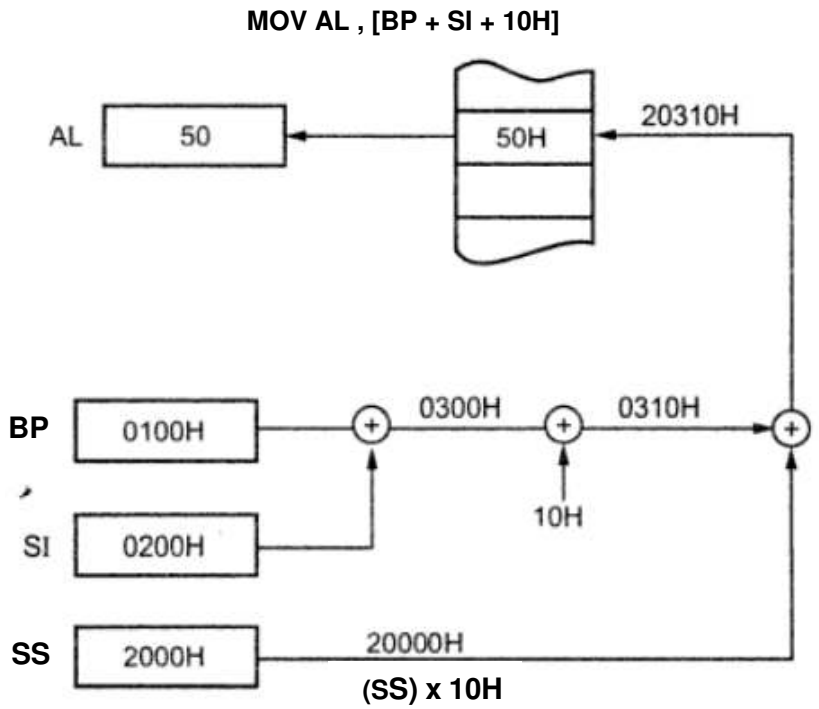
e) Based-Indexed with Displacement Mode : The effective address can be found as follows:

$$\textbf{Effective Address} = \left[Base \begin{matrix} BX \\ BP \end{matrix} + Index \begin{matrix} SI \\ DI \end{matrix} + disp. \begin{matrix} 8-bit \\ 16-bit \end{matrix}\right]$$

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

For example if (BP) = 0100H , (SI) = 0200H , (SS) = 2000H and a displacement of 10H then the instruction **MOV AL , [BP+SI+10H]** load the register AL with 50H form memory location 20310H.

PA = (2000)0H + [0100H + 0200H + 10H] = 20310H



The table below shows the operand that stores the effective address and the default segment register used to form the physical address

Addressing Mode	Operand	Default Segment
Direct	[offset]	DS
Register Indirect	[BX] or [SI] or [DI] [BP]	DS SS
Based Relative	[BX + disp] [BP + disp]	DS SS
Indexed Relative	[SI + disp] [DI + disp]	DS DS
Based-Indexed Relative	[BX + SI or DI + disp] [BP + SI or DI + disp]	DS SS

Note:

1. **CS : IP** together determine the address of the first instruction to execute. In this way the first instruction in **CS** being execution, if the first instruction is two byte long, the processor increment **IP** by **2** to indicate the next instruction.
2. **Segment override prefix** means that we can use any segment register to form the physical address as follows:

```
MOV AX , ES : [2000H]
ADD SS : [BX+SI] , DX
AND CS : [BP] , AL
```