Segment: Offset Memory address

- Since all registers in the 8086 are 16 bits wide, the address space is limited to \(2^{16}\), or 65,536 (64 K) locations.

- the memory is organized as a set of segments
- Each segment of memory is a linear contiguous sequence of up to 64K bytes

- In this segmented memory organization, we have to specify two components to identify a memory location: a **segment base** and an **offset**
This two-component specification is referred to as the *logical address*.

- The segment base specifies the start address of a segment in memory (called *paragraphs*) that can exist in a 20-bit address space.

- The offset specifies the address relative to the segment base. The offset is also referred to as the *effective address*.

- The Instruction Pointer register contains the offset address of the next sequential instruction to be executed.
Segment address 9000h

Physical Memory Space Address Range 00000h - FFFFFh

Offset Address

Physical Address Range

Physical Address = (Segment Address x 16) + Offset Address
Once the paragraph boundary is established by the segment register, the offset value (either a literal or register content) is sign extended and added to the shifted value from the segment register to form the full 20-bit address.
The 8086 architecture provides for 8 addressing modes. The first two modes operate on values that are stored in internal registers or are part of the instruction (immediate values).

1. *Register Operand Mode*: The operand is located in one of the 8 or 16 bit registers.

   MOV AX,DX
   MOV AL,BH
   INC AX
2. **Immediate Operand Mode**: The operand is part of the instruction.

MOV AX,0AC10h
ADD AL,0AAAh

- There is no prefix, such as the ‘#’ sign, to indicate that the operand is an immediate.
The next six addressing modes are used with memory operands, or data values that are stored in memory.

The effective addressing modes are used to calculate and construct the offset that is combined with the segment register to create the actual physical address used to retrieve or write the memory data.

The physical memory address is synthesized from the logical address contained in the segment register and an offset value, which may or may not be contained in a register.
3. **Direct Mode**: The memory offset value is contained as an 8-bit or 16-bit positive displacement value.

**MOV AX,[00AAh]**
copies the data from DS:00AA into the AX register, while,

**CS:MOV DX,[0FCh]**
copies the data from CS:00FC into the DX register

the square brackets are used to symbolize a memory instruction and the absence of brackets means an immediate value.
Also, two memory operands are not permitted for the MOV instruction. The instruction,

MOV [00AAh],[1000h]
is illegal. Unlike the 68K MOVE instruction, only one memory operand is permitted.
4. **Register Indirect Mode**: The operand offset is contained in one of the following registers:

- BP
- BX
- DI
- SI

The square brackets are used to indicate indirection. The following code snippet writes the value \(55h\) to memory address DS:0100.

```
MOV BX,100h
MOV AL,55h
MOV [BX],AL
```
5. **Based Mode**: The memory operand is the sum of the contents of the base register, BX or BP and the 8-bit or 16-bit displacement value. The following code snippet writes the value 0AAh to memory address DS:0104.

```
MOV BX,100h
MOV AL,0AAh
MOV [BX]4,AL
```

The Based Mode instruction can also be written: **MOV [BX-4],AL.** Since the displacement can be a positive or negative number, the above instruction is equivalent to: **MOV [BX + 0FFFCh],AL**
6. **Indexed Mode**: The memory operand is the sum of the contents of an index register, DI or SI, and the 8-bit or 16-bit displacement value.

The following code snippet writes the value 055h to memory address ES:0104.

```assembly
MOV DI,100h
MOV AL,0AAh
MOV [DI]4,AL
```
7. **Based Indexed Mode:** The memory operand offset is the sum of the contents of one of the base registers, BP or BX and one of the displacement registers, DI or SI.

- The DI register is normally paired with the ES register in the Indexed Mode, but when the DI register is used in this mode, the DS register is the default segment register.

The following code snippet writes the value 0AAh to memory location DS:0200h:

```assembly
MOV DX,1000h
MOV DS,DX
MOV DI,100h
MOV BX,DI
MOV AL,0AAh
MOV [DI+BX],AL
```

The instruction `MOV [DI+BX],AL` may also be written: `MOV [DI][BX],AL`