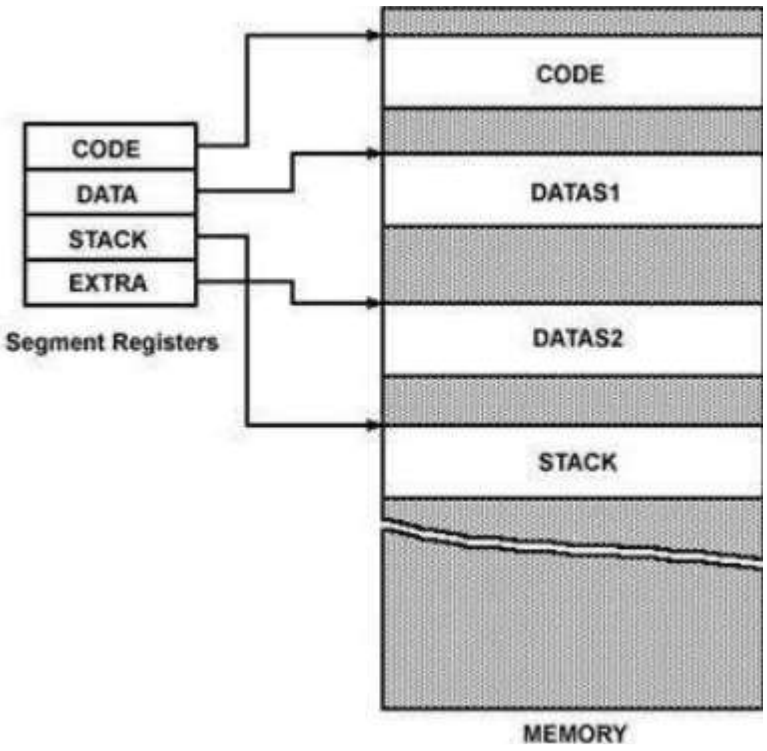
Software Systems-1

BASIC COMPONENTS OF COMPUTER SYSTEM- PART 1

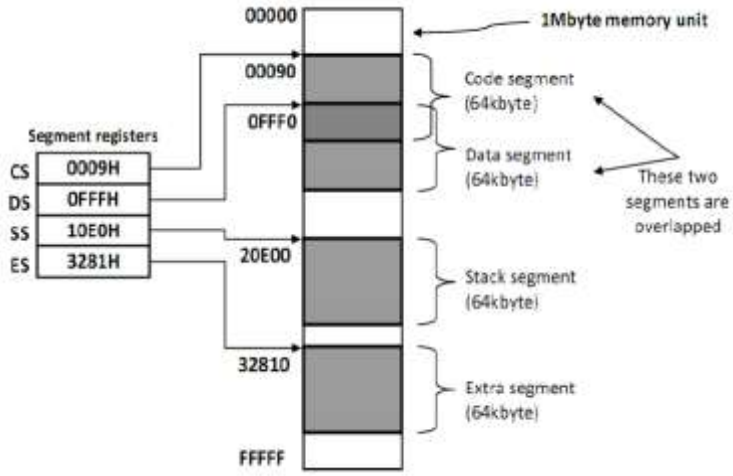
Dr. Azam E. Al-Rawachy
COLLEGE OF COMPUTER SCIENCE AND MATHEMATICS

1 MEMORY SEGMENTATION

The 8086 microprocessor operate in the **Real mode memory addressing**. Typically, much operating system code and almost all application programs run in **protected mode** *to ensure that essential data is not unintentionally overwritten*. **Real mode** operation allows the microprocessor to address only the first **1M byte** of memory space. The first 1M byte of memory is called either the **real memory** or **conventional memory** system. Even though the 8086 has a 1M byte address space, not all this memory is active at one time. Actually, the 1M bytes of memory are partitioned into 64K byte (65,536) segments. The 8086 microprocessor allow four memory segments.



Think of segments as windows that can be moved over any area of memory to access data or code. Also note that a program can have more than four segments, but can only access four segments at a time. Let the segment registers be assigned as follow: **CS = 0009H**, **DS = 0FFFH**, **SS = 10E0H**, and **ES = 3281H**.



We note here that code segment and data segment are overlapped while other segments are disjointed.

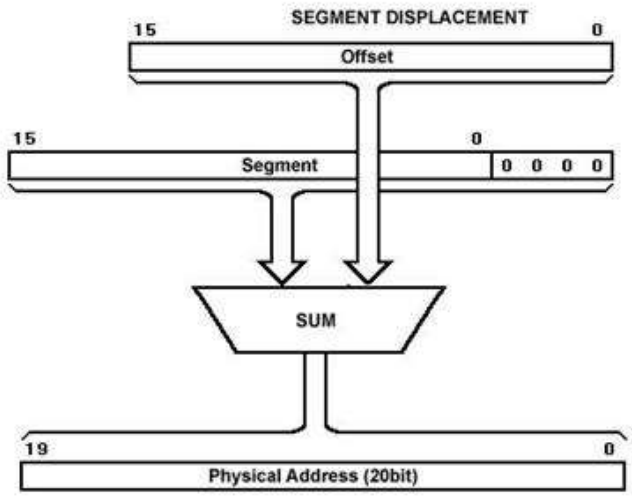
In the real mode a combinational of a segment address and offset address access a memory location. All real mode memory address must consist of a segment address plus an offset address. The microprocessor has a set of rules that apply to segments whenever memory is addressed. These rules define the segment register and offset register combination as follow:

Segment	Offset	Special Purpose
CS	IP	Instruction address
SS	BP	Stack address
SS	SP	Top of the stack
DS	BX, DI,SI, an 8-bit number, or a 16-bit number	Data address
ES	DI for string instructions	String destination address

This combination (CS: IP) locates the next instruction executed by the microprocessor. For example, if CS = 1400H and IP = 1200H, the microprocessor fetches its next instruction from memory location:

Physical address=Segment base address*10+Offset (Effective) address

PA = SBA * 10 + EA =1400H*10 +1200H=15200H.



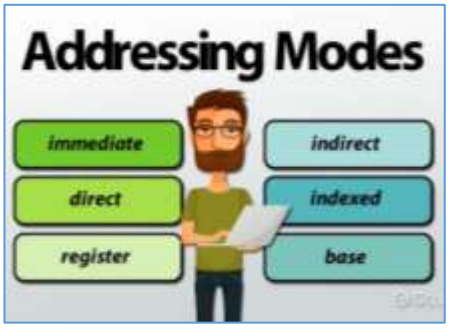
Segment memory addressing divides the memory into many segments. Each of these segment is addressed by a segment register. However, since the segment register is **16 bit** wide and the memory needs **20 bits** for an address the 8086 appends four bits' segment register to obtain the segment address. Therefore, to address the segment **10000H** by, say the **SS register**, the SS must contain **1000H**. The main advantage that memory segmentation has is that only **16 bit** registers are required both to store **segment base address** as well as **offset address**. This makes the internal circuitry easier to build as it removes the requirement for **20 bits** register in case the **linear addressing method** is used.

2 Addressing Mode of 8086

The addressing mode is the method to specify the operand of an instruction. The job of a microprocessor is to execute a set of instructions stored in memory to perform a specific task. Operations require the following:

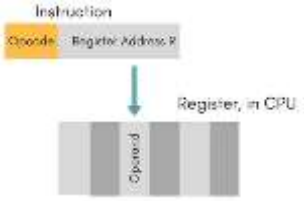
- The operator or opcode which determines what will be done.
- The operands which define the data to be used in the operation.

The 8086 addressing modes categorized into three types:



1. Register Addressing
2. Immediate Addressing
3. Memory Addressing

2.1 Register addressing mode



In this addressing mode, the operands may be:

- **reg16**: 16-bit general registers: AX, BX, CX, DX, SI, DI, SP or BP.
- **reg8:** 8-bit general registers: AH, BH, CH, DH, AL, BL, CL, or DL.
- **Sreg:** Segment registers: CS, DS, ES, or SS. For register addressing modes, there is no need to compute the effective address (Offset Address). The operand is in a register and to get the operand there is **no memory access involved**.

Some rules in register addressing modes:

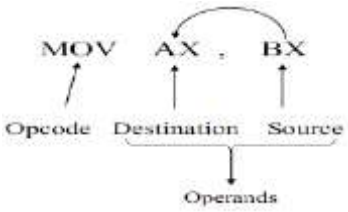
```
Example: Register Operands
MOV AX, BX      ;   mov reg16, reg16
ADD AX, SI      ;   add reg16, reg16
MOV DS, AX      ;   mov Sreg, reg16
```

- You may not specify CS as the destination operand, and should never use the segment registers as data registers to hold arbitrary values. They should only contain segment addresses.
  Example: MOV CS, 02H ➔ wrong
- Only one of the operands can be a segment register. You cannot move data from one segment register to another with a single MOV instruction. To copy the value of CS to DS, you would have to use some sequence like:

  MOV DS, CS ➔ wrong

  MOV AX, CS

  MOV DS, AX -> the way we do it



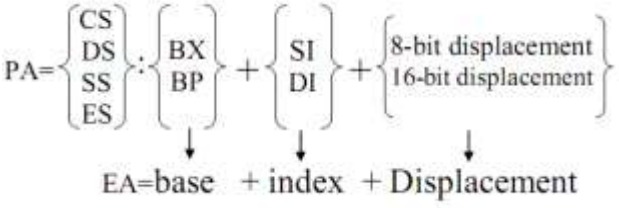
2.2 Immediate Addressing Mode

With immediate addressing mode, the actual data to be used as the operand is included in the instruction itself. **The immediate operand, which is stored along with the instruction, resides in the code segment, not in the data segment.** This addressing mode is also faster to execute an instruction because the operand is read with the instruction from memory. Here are some examples:

Example: Immediate Operands
MOV AL, 20 ; Copies a 20decimal into register AL
MOV BX,55H ; Copies a 0055H into register BX
MOV SI,0; Copies a 0000H into register SI
MOV CL, 10101001B ; Copies a 10101001binary into register CL

2.3 Memory Addressing Modes

To reference an operand in memory, the 8086 must calculate the physical address (**PA**) of the operand and then **initiate a read or write operation** of this storage location. The 8086 MPU is provided with a group of addressing modes known as the **memory operand addressing modes** for this purpose. Physical address can have computed from a segment base address (SBA) and an effective address (EA). SBA identifies the starting location of the segment in memory, and EA represents the offset of the operand from the beginning of this segment of memory.



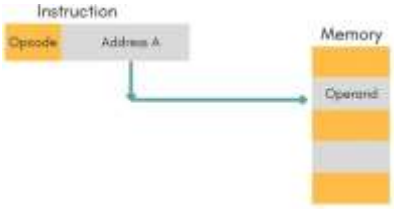
PA=SBA (segment): EA (offset)

PA=segment base: base + index + Displacement

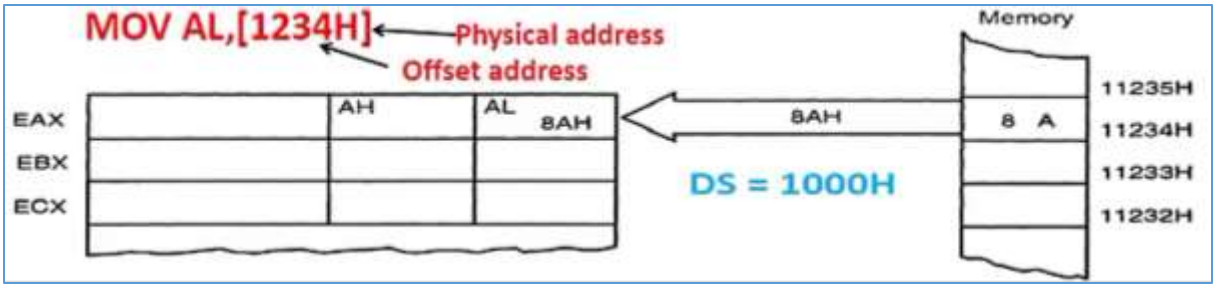
There are different forms of memory addressing modes

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

2.3.1 Direct Addressing Mode



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.

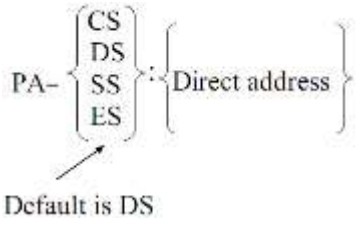


The Physical Address PA= DS * 10H+Offset Address

= 1000H*10H+1234H =11234H

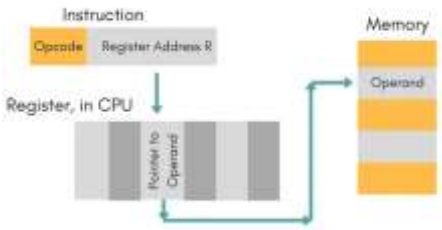
```
MOV AL, DS:[2000H]      ; move the contents of the memory location with offset 2000
or MOV AL, [2000H]        into register AL

MOV AL,DS:[8088H]       ; move the contents of the memory location with offset 8088
or MOV AL,[8088H]         into register AL
```



By default, all displacement-only values provide offsets into the data segment; DS. If you want to provide an offset into a different segment, you must use a **segment override prefix** before your address. For example, to access location 1234H in the extra segment (ES) you would use an instruction of the form MOV AX, ES: [1234H].

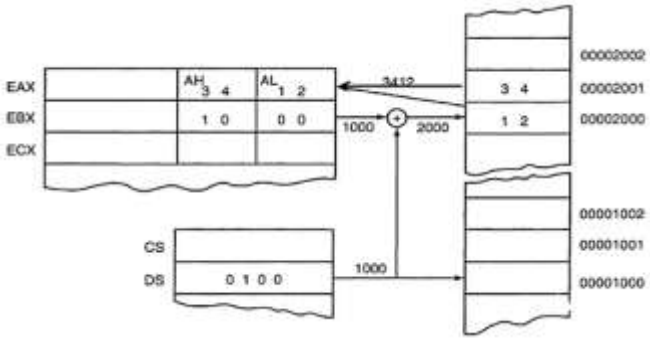
2.3.2 Register Indirect Addressing Mode



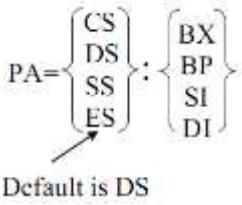
This mode is similar to the direct address except that the **effective address** held in any of the following register: BP, BX, SI, and DI.

MOV AL, [BX] OR MOV AL, [BP]
MOV AL, [SI] OR 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. The following instructions demonstrate the use of these overrides:

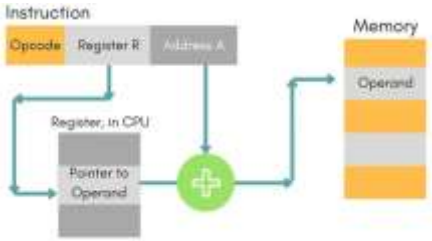


MOV AL, CS: [BX]
MOV AL, DS: [BP]
MOV AL, SS: [SI]
MOV AL, ES: [DI]



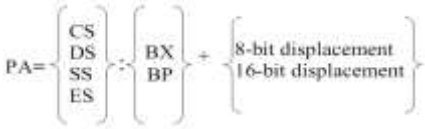
EX: Move AX, [BX], When DS =0100H

2.3.3 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. For example, if BX=1000, DS=0200, and AL=EDH, for the following instruction:

MOV [BX] + 1234H, AL



EA=BX+1234H = 1000H+1234H = **2234H**

PH=DS*10+EA =0200H*10+2234H = **04234H**

2.3.4 Indexed Addressing Modes

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.

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. THIS PERMITS ACCESS TO DATA IN THE STACK SEGMENT OF MEMORY.

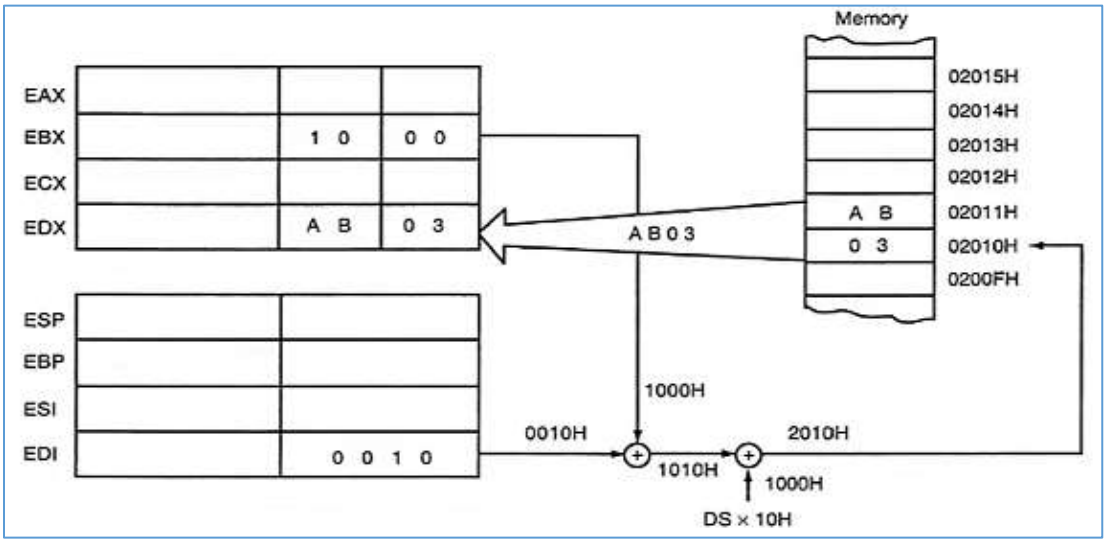
The indexed addressing modes use the following syntax:

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

2.3.5 Based Indexed Addressing Modes

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



2.3.6 Based indexed with displacement

It is a slight modification of Base/index addressing modes with the addition of an 8 bit/ 16 bit constant.

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

Assembly Language	*Size*	*Operation*
MOV CX,[BX+DI]	16-bits	Copies the word contents of the data segment memory location address by BX plus DI into CX
MOV CH,[BP+SI]	8-bits	Copies the byte contents of the stack segment memory location addressed by BP plus SI into CH
MOV [BX+SI],SP	16-bits	Copies SP into the data segment memory location addresses by BX plus SI
MOV [BP+DI],AH	8-bits	Copies AH into the stack segment memory location addressed by BP plus DI