2.1 8086 Microprocessor (µp):

The main features of 8086 µp are:

1. It is a 16-bit Microprocessor (µp). It's ALU, internal registers works with 16bit binary word.
2. 8086 has a 20 bit address bus can access up to 2^{20}= 1 MB memory locations.
3. 8086 has a 16bit data bus. It can read or write data to a memory/port either 16bits or 8 bit at a time.
4. It can support up to 64K I/O ports (2^{16}=64K).
5. Frequency range of 8086 is 6-10 MHz.
6. It has multiplexed address and data bus AD0- AD15 and A16 – A19.
7. It can pre-fetch up to 6 instruction bytes from memory and queues them in order to speed up instruction execution.
8. It requires +5V power supply.
9. A 40 pin dual in line package.
10. 8086 is designed to operate in two modes, Minimum mode and Maximum mode.
    a. The minimum mode is selected by applying logic 1 to the $MN/\overline{MX}$ input pin. This is a single microprocessor configuration.
    b. The maximum mode is selected by applying logic 0 to the $MN/\overline{MX}$ input pin. This is a multi-microprocessors configuration.

2.2 Architecture or Functional Block Diagram of 8086:

The micro-architecture of a processor is its internal architecture-that is, the circuit building blocks that implement the software and hardware architectures of the 8086 microprocessors. The micro-architecture of the 8086 microprocessors employs parallel processing-that is, they are implemented with several simultaneously operating processing units. Figure 2-1 shows the internal architecture of the 8086 microprocessors. They contain two processing units: the ***Bus Interface Unit*** (BIU) and the ***Execution Unit*** (EU).

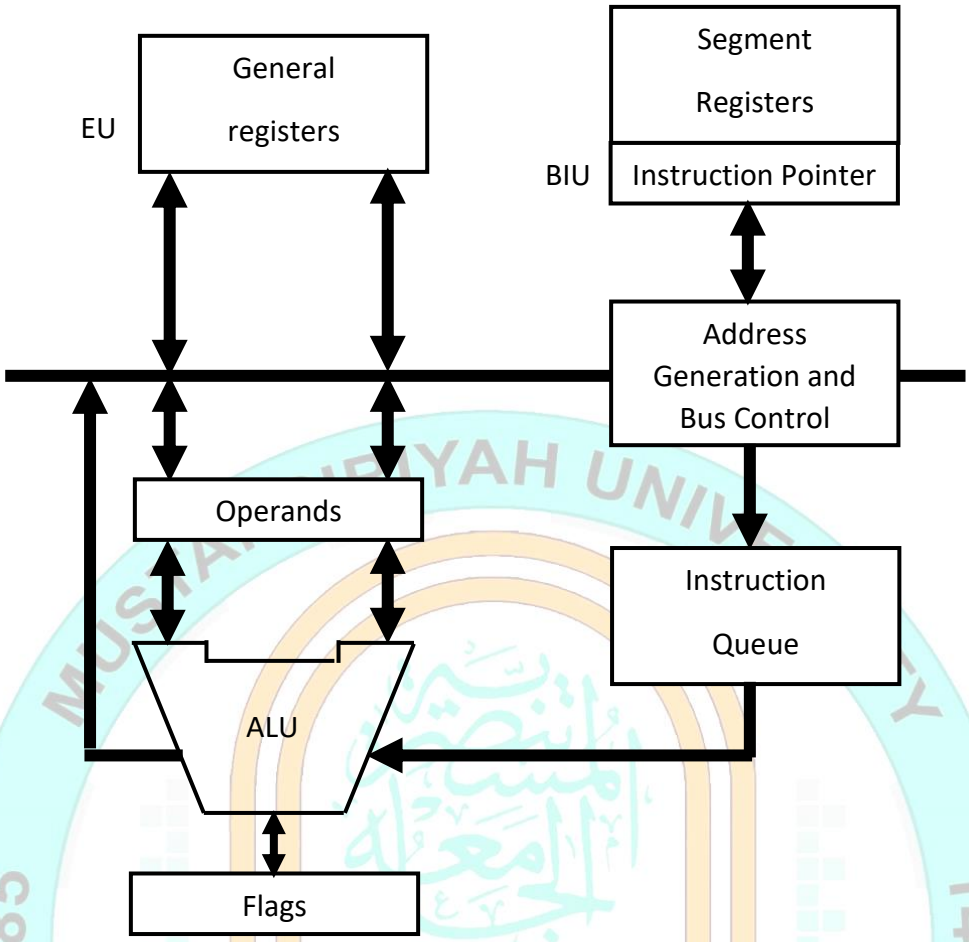


Fig. (2-1): Internal architecture of the 8086 microprocessor

2.2.1 BUS INTERFACE UNIT (BIU):

1. It provides a full 16 bit bidirectional data bus and 20 bit address bus.
2. The bus interface unit connects the microprocessor to external devices. BIU performs following operations:
   a. Instruction fetching.
   b. Reading and writing data of data operands for memory.
   c. Inputting/outputting data for input/output peripherals.
   d. And other functions related to instruction and data acquisition.
3. To implement above functions, the BIU contains the segment registers, the instruction pointer, address generation adder, bus control logic, and an instruction queue.
4. The BIU uses a mechanism known as an instruction stream queue to implement pipeline architecture.

2.2.2 EXECUTION UNIT (EU):

1. The Execution unit is responsible for decoding and executing all instructions.

2. The EU consists of arithmetic logic unit (ALU), status and control flags, general-purpose registers, and temporary-operand registers.
3. The EU extracts instructions from the top of the queue in the BIU, decodes them, generates operands if necessary, passes them to the BIU and requests it to perform the read or write by cycles to memory or I/O and perform the operation specified by the instruction on the operands.
4. During the execution of the instruction, the EU tests the status and control flags and updates them based on the results of executing the instruction.

2.3 Internal Microprocessor Architecture:

Before a program is written or any instruction investigated, the internal configuration of the microprocessor must be known.

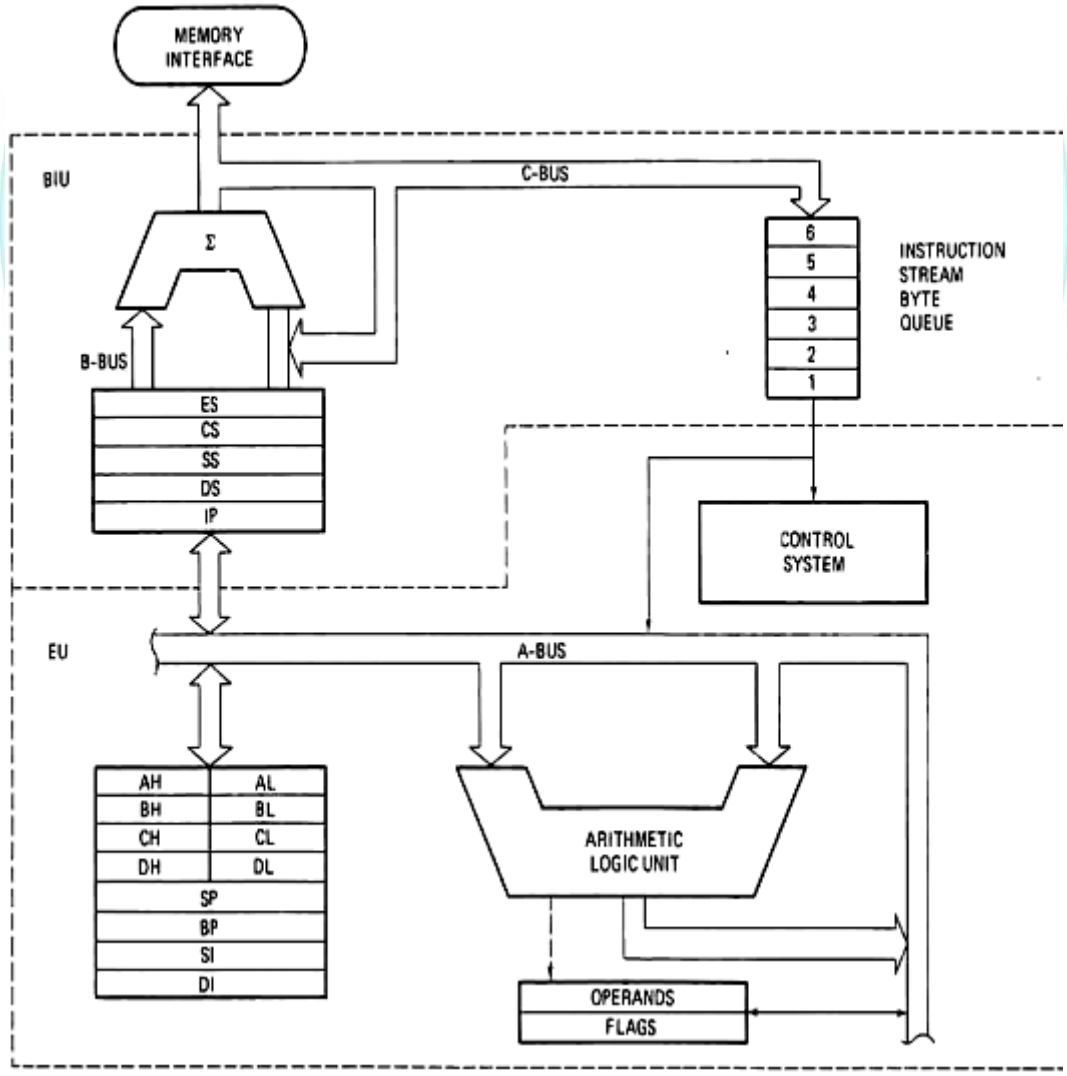


Fig. (2-2): Software Model of the 8086 microprocessor.

2.4 The Programming Model

The programming model of the 8086 microprocessor is considered to be program visible because its registers are used during application programming and are specified by the instructions. Other registers are considered to be program invisible because they are not addressable directly during applications programming, but may be used indirectly during system programming. Figure 2–3 illustrates the programming model of the 8086 microprocessor including the 16-bit extensions.

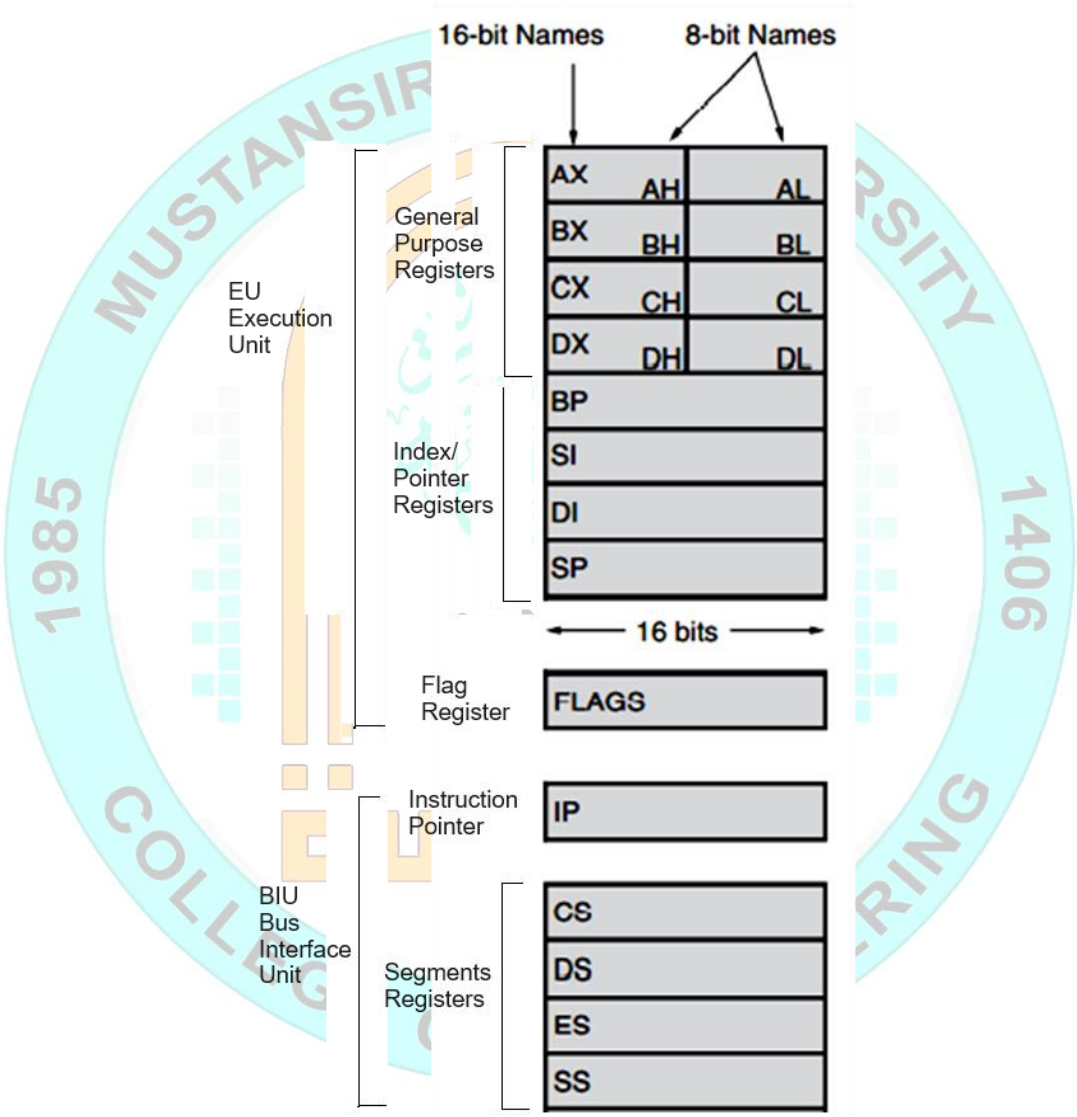
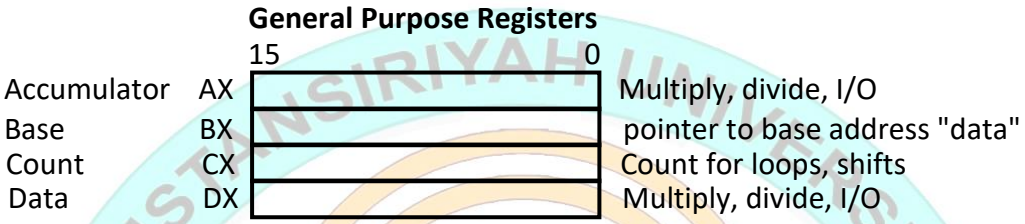


Fig. (2–3): The programming model of the 8086 microprocessor including the 16-bit extensions.

The 8-bit registers are AH, AL, BH, BL, CH, CL, DH, and DL and are referred to when an instruction is formed using these two-letter designations. For example, an ADD AL,AH instruction adds the 8-bit contents of AH to AL. (Only AL changes due to this instruction.) The 16-bit registers are AX, BX, CX, DX, SP, BP, DI, SI, IP, FLAGS, CS, DS, ES, SS, FS, and GS. Note that the first 4 16 registers contain a pair of 8-

bit registers. An example is AX, which contains AH and AL. The 16-bit registers are referenced with the two-letter designations such as AX. For example, an ADD DX, CX instruction adds the 16-bit contents of CX to DX. (Only DX changes due to this instruction).

1. The multipurpose registers (General Purpose Register + Index or Pointer Registers) include AX, BX, CX, DX, BP, DI, and SI. These registers hold various data sizes (bytes or words) and are used for almost any purpose, as dictated by a program.



AX (accumulator): AX is referenced as a 16-bit register (AX), or as either of two 8-bit registers (AH and AL). The accumulator is used for instructions such as multiplication, division, and some of the adjustment instructions. For these instructions, the accumulator has a special purpose, but is generally considered to be a multipurpose register. Also used for I/O operations and string manipulation.

BX (Base Index): BX is referenced as a 16-bit register (BX), or as either of two 8-bit registers (BH and BL). The BX register (base index) sometimes holds the offset address of a location in the memory system in all versions of the microprocessor. (data pointer used for based, indexed or register indirect addressing).

CX (count): CX is referenced as a 16-bit register (CX), or as either of two 8-bit registers (CH and CL). CX is a general-purpose register that also holds the count for various instructions also can address memory data. Instructions that use a count are the repeated string instructions, shift, rotate, and loop instructions. The shift and rotate instructions use CL as the count, the repeated string instructions use CX, and the LOOP/LOOPD instructions use CX.

DX (data): DX is referenced as a 16-bit register (DX), or as either of two 8-bit registers (DH and DL). DX is a general-purpose register that holds a part of the result from a multiplication or part of the dividend before a division. Also used for I/O port number, multiply and divide instructions.

Pointer and Index Registers

		15 … 0	
Stack Pointer	SP		Pointer to top of stack
Base pointer	BP		Pointer to base address (stack)
Source Index	SI		Source string/index pointer
Destination Index	DI		Destination string/index pointer

BP (base pointer): BP points to a memory location in all versions of the microprocessor for memory data transfers. Also points to data in stack segment and used by subroutines to locate variables that were passed on the stack by a calling program.

DI (destination index): DI often addresses string destination data for the string instructions. Used in conjunction with ES. DI & SI used for address holding.

SI (source index): The source index register often addresses source string data for the string instructions. Like DI, SI also functions as a general-purpose register. SI & DI used to point to data location in stack.

2. The Special-Purpose Registers include IP, SP, and FLAGS; and the segment registers include CS, DS, ES, SS, FS, and GS.

IP (instruction pointer): IP addresses the next instruction in a section of memory defined as a code segment. The instruction pointer, which points to the next instruction in a program, is used by the microprocessor to find the next sequential instruction in a program located within the code segment. The instruction pointer can be modified with a jump or a call instruction.

SP (Stack pointer): SP addresses an area of memory called the stack. The stack memory stores data through this pointer and is explained later in the text with the instructions that address stack data. So it used to hold the address of the top of stack. The stack is maintained as a LIFO with its bottom at the start of the stack segment (specified by the SS segment register). Unlike the SP register, the BP can be used to specify the offset of other program segments.

FLAGS: FLAGS indicate the condition (current state) of the microprocessor and control its operation. They are modified automatically by CPU after mathematical & logical operations. Figure 2–4 shows the flag registers of all versions of the microprocessor.

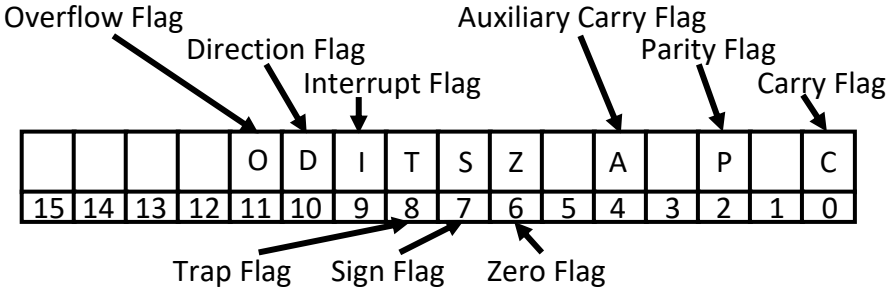


Fig. (2–4): The FLAG register counts for the entire 8086 microprocessor.

C (carry): Carry holds the carry after addition or the borrow after subtraction. The carry flag also indicates error conditions (overflow), as dictated by some programs and procedures.

P (parity): Parity is a logic 0 for odd parity and a logic 1 for even parity. Parity is the count of ones in a number expressed as even or odd. For example, if a number contains three binary one bits, it has odd parity. If a number contains no one bits, it has even parity. The parity flag finds little application in modern programming and was implemented in early Intel microprocessors for checking data in data communications environments. Today parity checking is often accomplished by the data communications equipment instead of the microprocessor.

A (auxiliary carry): The auxiliary carry holds the carry (half-carry) after addition or the borrow after subtraction between bit positions 3 and 4 of the result. This highly specialized flag bit is tested by the DAA and DAS instructions to adjust the value of AL after a BCD addition or subtraction. Otherwise, the A flag bit is not used by the microprocessor or any other instructions.

Z (zero): The zero flag shows that the result of an arithmetic or logic operation is zero. If Z=1, the result is zero; if Z=0, the result is not zero. This may be confusing, but that is how Intel decided to name this flag.

S (sign): The sign flag holds the arithmetic sign of the result after an arithmetic or logic instruction executes. If S=1, the sign bit (leftmost bit of a number) is set or negative; if S=0, the sign bit is cleared or positive.

T (trap): The trap flag enables trapping through an on-chip debugging feature. (A program is debugged to find an error or bug.) If the T flag is enabled (1), the microprocessor interrupts the flow of the program on conditions as indicated by the debug registers and control registers. If the T flag is a logic 0, the trapping (debugging) feature is disabled.

I (interrupt): The interrupt flag controls the operation of the INTR (interrupt request) input pin. If I=1, the INTR pin is enabled; if I=0, the

INTR pin is disabled. The state of the I flag bit is controlled by the STI (set I flag) and CLI (clear I flag) instructions.

D (direction): The direction flag selects either the increment or decrement mode for the DI and/or SI registers during string instructions. If D=1, the registers are automatically decremented; if D=0, the registers are automatically incremented. The D flag is set with the STD (set direction) and cleared with the CLD (clear direction) instructions.

O (overflow): Overflows occur when signed numbers are added or subtracted. An overflow indicates that the result has exceeded the capacity of the machine. For example, if 7FH (+127) is added—using an 8-bit addition—to 01H (+1), the result is 80H (–128). This result represents an overflow condition indicated by the overflow flag for signed addition. For unsigned operations, the overflow flag is ignored.

3. Segment Registers. Additional registers, called segment registers, generate memory addresses when combined with other registers in the microprocessor. A segment register functions differently in the real mode when compared to the protected mode operation of the microprocessor.

Segment Registers

Code Segment	CS	
Data Segment	DS	
Stack Segment	SS	
Extra Segment	ES	

CS (code): The code segment is a section of memory that holds the code (programs and procedures) used by the microprocessor. The code segment register defines the starting address of the section of memory holding code. In real mode operation, it defines the start of a 64K byte section of memory; in protected mode, it selects a descriptor that describes the starting address and length of a section of memory holding code.

DS (data): The data segment is a section of memory (64KB) that contains most data used by a program. Data are accessed in the data segment by an offset address or the contents of other registers AX, BX, CX, DX + SI, DI that hold the offset address. As with the code segment and other segments, the length is limited to 64K bytes in the 8086.

ES (extra): The extra segment is an additional data segment that is used by some of the string instructions to hold destination data and starting address of ES. It cannot be used in arithmetic operations.

SS (stack): The stack segment defines the area of memory (64KB) used for the stack. The stack entry point is determined by the stack segment

and stack pointer registers. The base pointer BP register also addresses data within the stack segment.

2.5 Real Mode Memory Addressing:

The 8086 operate exclusively in the real mode operation which allows the microprocessor to address only the first 1M byte of memory space. Note that the first 1M byte of memory is called the real memory, conventional memory, or DOS memory system. Real mode operation allows application software written for the 8086, which only contains 1M byte of memory. The upward compatibility of software is partially responsible for the continuing success of the Intel family of microprocessors. In all cases, each of these microprocessors begins operation in the real mode by default whenever power is applied or the microprocessor is reset.

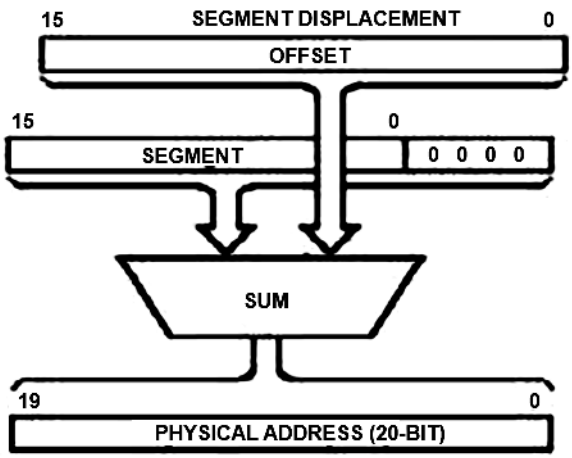


Fig. (A): Generating a physical address.

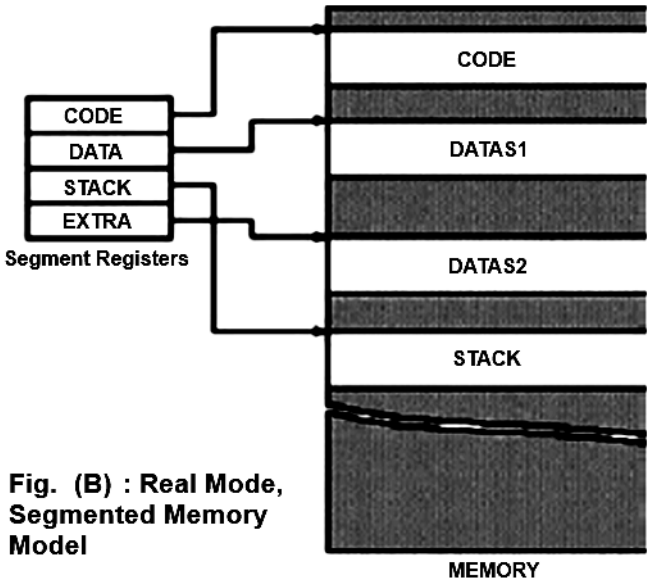


Fig. (B) : Real Mode, Segmented Memory Model

2.5.1 Segments and Offsets

A combination of a segment address and an offset address accesses a memory location in the real mode. All real mode memory addresses must consist of a segment address plus an offset address. The segment address, located within one of the segment registers, defines the beginning address of any 64K-byte memory segment. The offset address selects any location within the 64K byte memory segment. Segments in the real mode always have a length of 64K bytes. Figure 2–5 shows how the segment plus offset addressing scheme selects a memory location. Note that the offset or displacement is the distance above the start of the segment.

Paragraph: In the real mode, each segment register is internally appended with a 0H on its rightmost end. This forms a 20-bit memory

address, allowing it to access the start of a segment. The microprocessor must generate a 20-bit memory address to access a location within the first 1M of memory. For example, when a segment register contains 1200H, it addresses a 64K-byte memory segment beginning at location 12000H. Likewise, if a segment register contains 1201H, it addresses a memory segment beginning at location 12010H. Because of the internally appended 0H, real mode segments can begin only at a l6-byte boundary in the memory system.

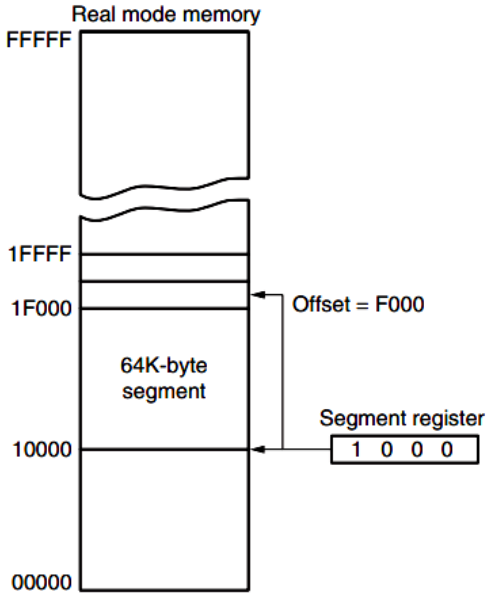


Fig. (2–5): The real mode memory-addressing scheme, using a segment address plus an offset.

Because a real mode segment of memory is 64K in length, once the beginning address is known, the ending address is found by adding FFFFH. For example, if a segment register contains 3000H, the first address of the segment is 30000H, and the last address is or 3FFFFH. Table 2–1 shows several examples of segment register contents and the starting and ending addresses of the memory segments selected by each segment address.

The offset address, which is a part of the address, is added to the start of the segment to address a memory location within the memory segment. For example, if the segment address is 1000H and the offset address is 2000H, the microprocessor addresses memory location 12000H. The offset address is always added to the starting address of the segment to locate the data. The segment and offset address is sometimes written as 1000:2000 for a segment address of 1000H with an offset of 2000H.

Some addressing modes combine more than one register and an offset value to form an offset address. When this occurs, the sum of these values may exceed FFFFH. For example, the address accessed in a segment whose segment address is 4000H and whose offset address is specified as

the sum of F000H plus 3000H will access memory location 42000H instead of location 52000H. When the F000H and 3000H are added, they form a l6-bit (modulo 16) sum of 2000H used as the offset address; not 12000H, the true sum. Note that the carry of 1 (F000H+3000H=12000H) is dropped for this addition to form the offset address of 2000H. The address is generated as 4000:2000 or 42000H.

Table (2–1) Example of real mode segment addresses.

Segment Register	*Starting Address*	*Ending Address*
2000H	20000H	2FFFFH
2001H	20010H	3000FH
2100H	21000H	30FFFH
AB00H	AB000H	BAFFFH
1234H	12340H	2233FH

2.5.2 Default Segment and Offset Registers

The microprocessor has a set of rules that apply to segments whenever memory is addressed. These rules, which apply in the real mode, define the segment register and offset register combination. For example, the code segment register is always used with the instruction pointer to address the next instruction in a program. This combination is CS:IP, depending upon the microprocessor's mode of operation. The code segment register defines the start of the code segment and the instruction pointer locates the next instruction within the code segment. 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 14000H+1200H or 15200H.

Physical Address=Segment Base Address*10+Offset (Effective) Address

PA = SBA * 10+ EA

Another of the default combinations is the stack. Stack data are referenced through the stack segment at the memory location addressed by either the stack pointer (SP) or the pointer (BP). These combinations are referred to as SS:SP, or SS:BP. For example, if SS=2000H and BP=300H, the microprocessor addresses memory location 23000H for the stack segment memory location. Note that in real mode, only the rightmost 16 bits of the extended register address a location within the memory segment. Other defaults are shown in Table 2–2 for addressing memory using 8086 microprocessor with 16-bit registers.

Table (2–2) Example of real mode segment addresses.

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

Figure 2–6 shows a system that contains four memory segments. Note that a memory segment can touch or even overlap if 64K bytes of memory are not required for a segment.

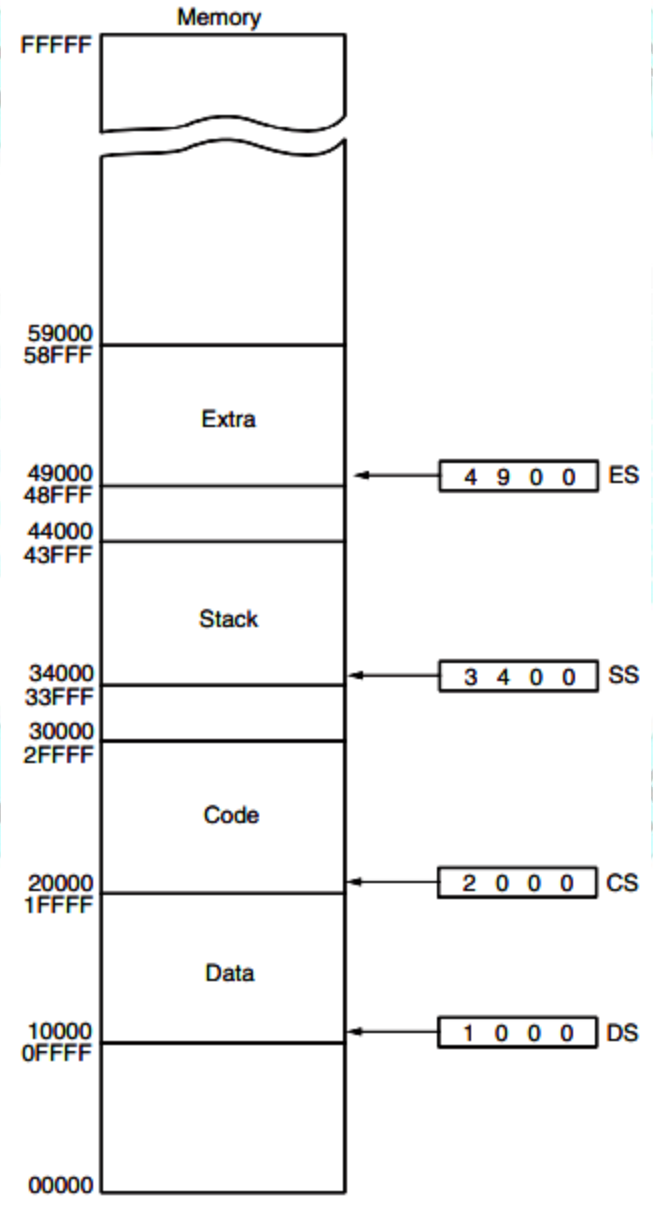


Fig. (2–6): A memory system showing the placement of four memory segments.

Q.1: What is segmentation? What are its advantages? How is segmentation implemented in typical microprocessors?

Ans.: Segment memory addressing divides the memory into many segments. Each of these segments can be considered as a linear memory space. 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 first 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. The second advantage is relocate-ability.

Q.2: Which register holds a count for some instructions?

Ans.: In the 8086 microprocessor, the CX (Count Register) is used to hold a count for certain instructions, particularly loop and string manipulation instructions.

Q.3: What is the purpose of the IP register?

Ans.: In the 8086 microprocessor, the IP (Instruction Pointer) register holds the offset address of the next instruction to be executed within the current code segment. The IP works together with the CS (Code Segment) register to form the full 20-bit address of the instruction. The physical address of the instruction is calculated as: Physical Address=(CS×16)+ IP

Q.4: The carry flag bit is not modified by which arithmetic operations?

Ans.: In the 8086 microprocessor, the Carry Flag (CF) is not modified by the following arithmetic operation:

INC (Increment) and DEC (Decrement) instructions:
INC destination → Increments the operand by 1.
DEC destination → Decrements the operand by 1.

Q.5: Will an overflow occur if a signed FFH is added to a signed 01H?

Ans.:

Step 1: Interpret operands as signed 8-bit values

- FFH (in binary): 1111 1111 → This is **–1** in signed 8-bit (2's complement) form.
- 01H (in binary): 0000 0001 → This is **+1** in signed 8-bit.

Step 2: Perform the addition –1 + 1 = 0

- Binary addition: 1111 1111 + 0000 0001 = 0000 0000

Step 3: Check for overflow, Overflow in signed arithmetic occurs when:

- Adding two **positive numbers** gives a **negative result**.
- Adding two **negative numbers** gives a **positive result**.

Here:

- One operand is negative (FFH=–1), and the other is positive (01H=+1).
- The result is 00H = 0, which is valid and expected.

∴ **No**, an overflow **will not occur** when signed FFH is added to signed 01H in the 8086 microprocessor.