Features of 8086 Microprocessor:

The features of 8086 Microprocessor are:

1) The 8086 is a 16-bit microprocessor. The term "16-bit" means that its arithmetic logic unit, internal registers and most of its instructions are designed to work with 16-bit binary words.
2) The 8086 has a 16-bit data bus, so it can read data from or write data to memory and ports either 16 bits or 8 bits at a time. The 8088, however, has an 8-bit data bus, soil can only read data from or write data to memory and ports 8 bits at a time.
3) The 8086 has a 20-bit address bus, so it can directly access 2^{20} or 10,48,576 (1Mb) memory locations. Each of the 10, 48, 576 memory locations is byte Therefore, a sixteen-bit words are stored in two consecutive memory locations. The 8088 also has a 20-bit address bus, so it can also address 2^{20} or 10, 48, 576 memory locations.
4) The Features of 8086 Microprocessor can generate 16-bit I/O address, hence it can access 2^{16} = 65536 I/O ports.
5) The 8086 provides fourteen 16-bit registers.
6) The 8086 has multiplexed address and data bus which reduces the number of pins needed, but does slow down the transfer of data (drawback).
7) The 8086 requires one phase clock with a 33% duty cycle to provide optimized internal timing.

Range of clock rates (refer Fig. 6.1) are :-

5 MHz for 8086

8 MHz for 8086-2

10 MHz for 8086-1

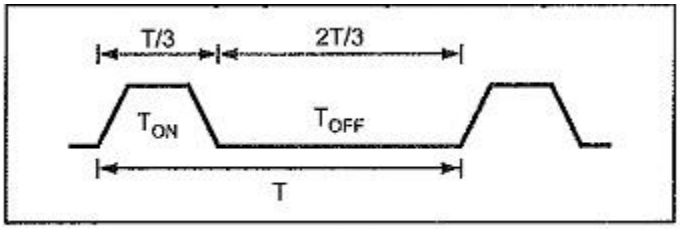

Fig. 6.1 Clock cycle

8) The Features of 8086 Microprocessor is possible to perform bit, byte, word and block operations in 8086. It performs the arithmetic and logical operations on bit, byte, word and decimal numbers including multiply and divide.
9) The Intel 8086 is designed to operate in two modes, namely the minimum mode and the maximum mode. When only one 8086 CPU is to be used in a microcomputer system, the 8086 is used in the minimum mode of operation. In this mode the CPU issues the control signals required by memory and I/O In multiprocessor (more than one processor in the system) system 8086 operates in maximum mode. In maximum mode, control signals are generated with the help of external bus controller (8288).

10) The Intel 8086 supports multiprogramming. In multiprogramming, the code for two or more processes is in memory at the same time and is executed in a time-multiplexed fashion.
11) An interesting feature of the 8086 is that it fetches up to six instruction bytes (4 instruction bytes for 8088) from memory and queue stores them in order to speed up instruction execution.
12) The Features of 8086 Microprocessor provides powerful instruction set with the following addressing modes: Register, immediate, direct, indirect through an index or base, indirect through the sum of a base and an index register, relative and implied.

8086 Internal Architecture:

Fig. 6.2 shows a block diagram of the 8086 internal architecture. It is internally divided into two separate functional units. These are the Bus Interface Unit (BIU) and the Execution Unit (EU). These two functional units can work simultaneously to increase system speed and hence the throughput. Throughput is a measure of number of instructions executed per unit time.

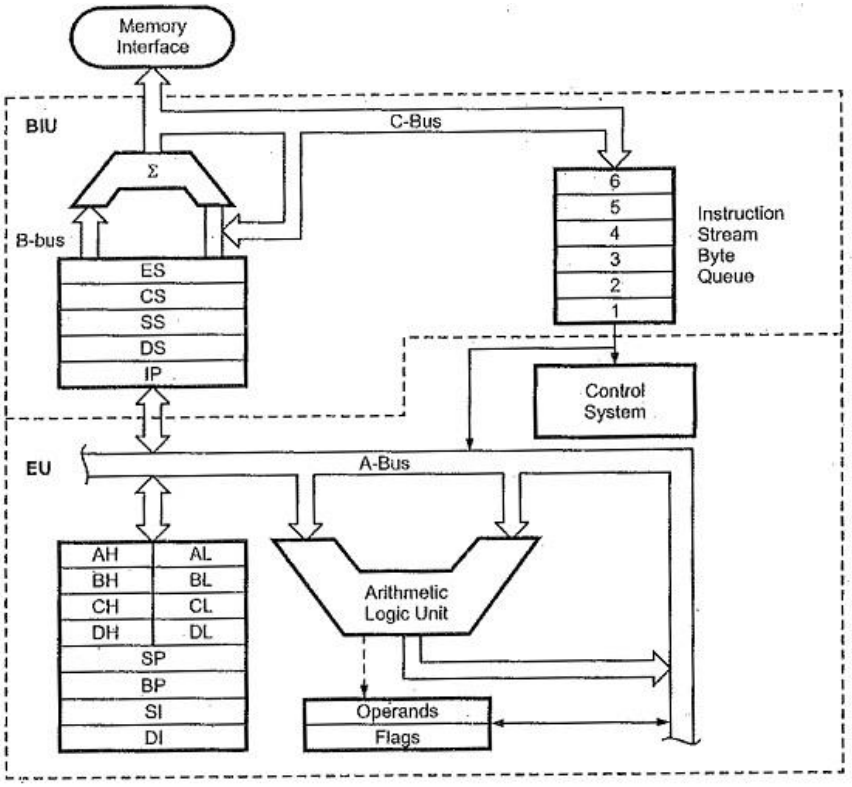

Fig. 6.2 8086 Internal block diagram

Bus Interface Unit:

The bus interface unit is the 8086 Internal Architecture to the outside world. It provides a full 16-bit bi-directional data bus and 20-bit address bus. The bus interface unit is responsible for performing all external bus operations, as listed below.

Functions of Bus Interface Unit

- It sends address of the memory or I/O.
- It fetches instruction from memory.
- It reads data from port/memory.
- It Writes data into port/memory.
- It supports instruction queuing.
- It provides the address relocation facility.

To implement these functions the BIU contains the instruction queue, segment registers instruction pointer, address summer and bus control logic.

Instruction Queue:

To speed up program execution, the BIU fetches six instruction bytes ahead of time from the memory. These pre-fetched instruction bytes are held for the execution unit in a group of registers called Queue. With the help of queue it is possible to fetch next instruction when current instruction is in execution. For example, current instruction in execution is a multiplication instruction. In 8086 Internal Architecture, operands for multiplication operations are within registers. Still it requires 100 clock cycles to execute multiply instruction. Like multiplication there are number of other instructions in 3086 which need a quite a large number- of clock cycles for execution. During this execution time the BIU fetches the next instruction or instructions from memory into the instruction queue instead of remaining idle. The BIU continues this process as long as the queue is not full. Due to this, execution unit gets the ready instruction in the queue and instruction fetch time is eliminated. This is illustrated in Fig. 6.3.

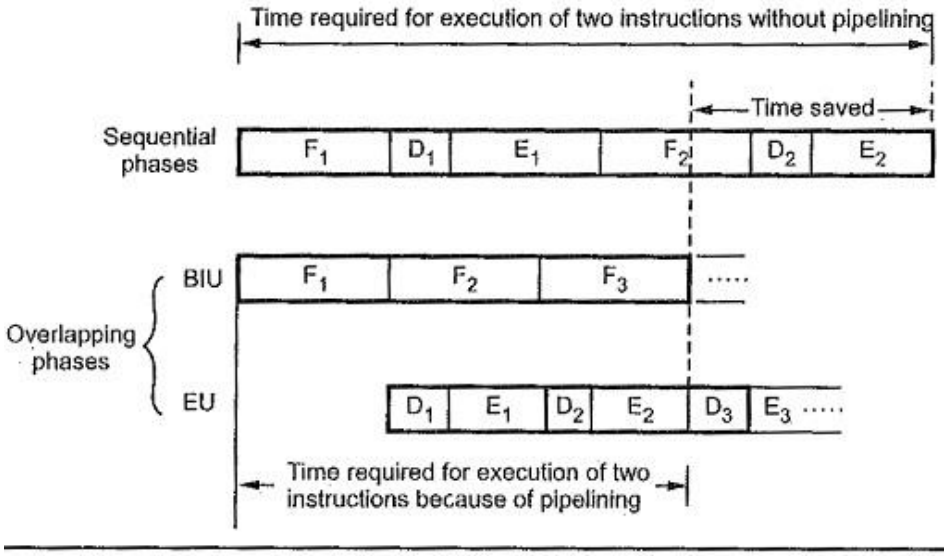

Fig. 6.3 Pipelining

The queue operates on the principle first in first out (FIFO). So that the execution unit gets the instructions for execution in the order they are fetched. In case of JUMP and CALL instructions, instructions already fetched in queue are of no use. Hence, in these cases queue is dumped and newly formed by loading instructions from new address specified by JUMP or CALL instruction. Feature of fetching the next instruction while the current instruction is executing is called pipelining.

Segment Registers:

The physical address of the 8086 Internal Architecture is 20-bits wide to access 1 Mbyte memory locations. However, its registers and memory locations which contain logical addresses are just 16-bits wide. Hence 8086 uses memory segmentation. It treats the 1 Mbyte of memory as divided into segments, with a maximum size of a segment as 64 Kbytes. Thus any location within the segment can be accessed using 16 bits. The 8086 Internal Architecture allows only four active segments at a time, as shown in the Fig. 6.4. For the selection of the four active segments the 16-bit segment registers are provided within the BIU of the 8086. These four registers are :

Code segment (CS) register, the data segment (DS) register, the stack segment (SS) register, and the extra segment (ES) register. These are used to hold the upper 16-bits of the starting addresses of the four memory segments, on which 8086 works at a particular time. For example, the value in CS identifies the starting address of 64 K-byte segment known as code segment. By “starting address“, we mean the lowest addressed byte in the active code segment. The starting address is also known as base address or segment base.

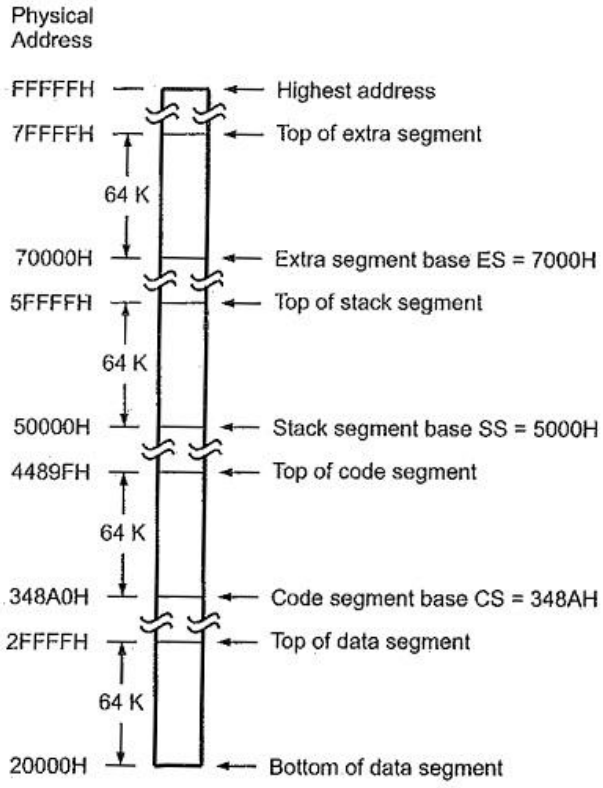

Fig. 6.4 Memory segmentation

The BIT) always inserts zeros for the lowest 4 bits (nibble) in the contents of segment register to generate 20-bit base address. For example, if the code segment register contains 348AH, then code segment will start at address 348A0H.

Functions of Segment Registers:

- The CS register holds the upper 16-bits of the starting address of the segment from which the BIU is currently fetching the instruction code byte.
- The SS register is used for the upper 16-bits of the starting address for the program stack (all stack related instructions will operate on stack)
- ES register and DS register are used to hold the upper 16-bits of the starting address of the two memory segments which are used for data.

Rules for Memory Segmentation:

1. The four segments can overlap for small programs. In a minimum system all four segments can start at the address 00000H.
2. The segment can begin/start at any memory address which is divisible by 16.

Advantages of Memory Segmentation:

1. It allows the memory addressing capacity to be 1 Mbyte even though the address associated with individual instruction is only 16-bit.
2. It allows instruction code, data, stack, and portion of program to be more than 64 KB long by using more than one code, data, stack segment, and extra
3. It facilitates use of separate memory areas for program, data and stack.
4. It permits a program or its data to be put in different areas of memory, each time the program is executed i.e. program can be relocated which is very useful in multiprogramming.

Instruction Pointer:

The instruction pointer register holds the 16-bit address of the next Code byte within the code segment. The value contained in the IP is referred to as an **offset**. This value must be offset from (added to) the segment base address in CS to produce the required 20-bit physical address.

Generation of 20-bit Address:

The contents of the CS register are multiplied by 16 i.e. shifted by 4 position to the left by inserting 4 zero bits and then the offset i.e. the contents of IP register are added to the shifted contents of CS to generate physical address. As shown in the Fig 6.5, the contents of CS register are 348AH, therefore the shifted contents of CS register are 348A0H. When the BIU adds the offset of 4214H in the IP to this starting address, the result is 20-bit physical of 38AB4H.

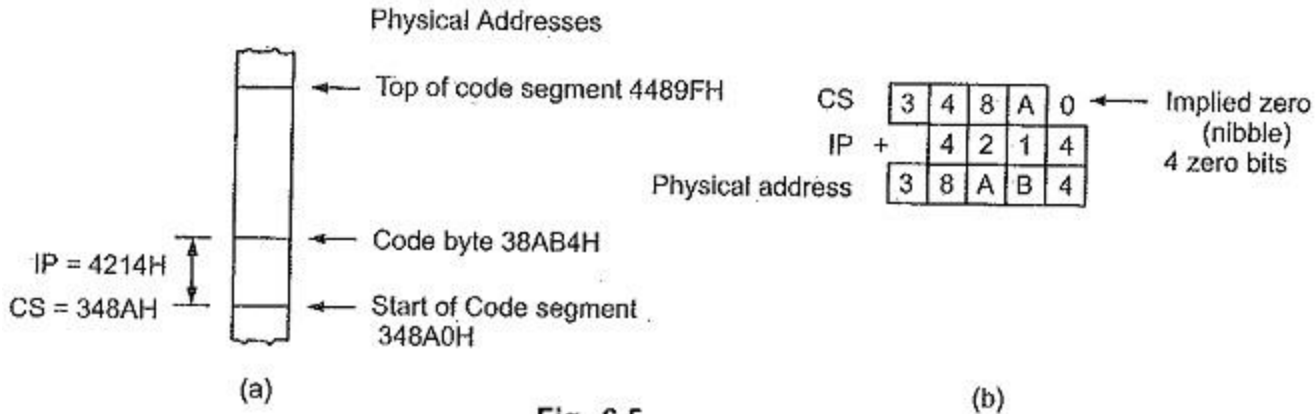

Fig. 6.5

Execution Unit [EU]:

The execution unit of 8086 Internal Architecture tells the BIU from where to fetch instructions or data, decodes instructions and executes instructions. It contains

- Control Circuitry
- Instruction Decoder
- Arithmetic Logic Unit (ALU)
- Flag Register
- General Purpose Registers
- Pointers and Index Registers

Control Circuitry, Instruction Decoder, ALU:

The control circuitry in the EU directs the internal operations. A decoder in the EU translates the instructions fetched from memory into a series of actions wlifeh the EU performs. ALU is 16-bit. It can add, subtract, AND, OR, XOR, increment, decrements, complement and shift binary numbers.

Flag Register:

A flag is a flip–flop which indicates some condition produced by the execution of an instruction or controls certain operations of the EU. The flag register contains nine active flags as shown in the Fig. 6.6.

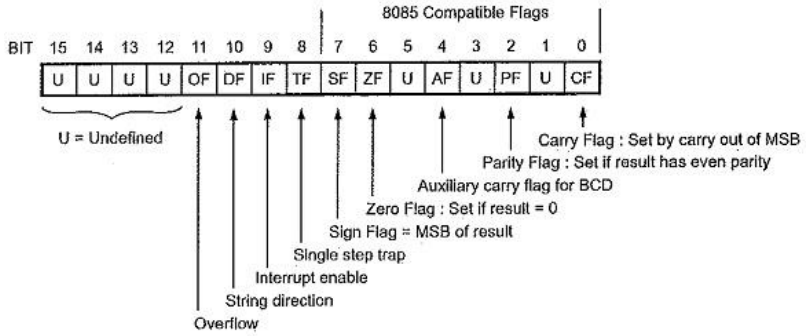

Fig. 6.6 8086 Flag register bit pattern

General Purpose Registers:

The EU has. 8 general purpose registers labeled ATI, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually for temporary storage of 8 bit data. The AL register is also called accumulator. Certain pairs of these general purpose registers can be used together to store 16-bit data, such as AX, BX, CX and DX.

Pointers and Index Registers:

All segment registers are 16-bit. But it is necessary to put 20-bit address (physical address) on the address bus. To get 20-bit physical address one more register is associated with each segment register the way IP is associated with CS.

These additional registers belong to the pointer and index group. The pointer and index group consists of instruction pointer (IP), stack pointer (SP), BP (base pointer), source index (SI) and destination index (DI) registers.

Stack Pointer (SP) : The stack pointer (SP) register contains the 16-bit offset from the start of the segment to the top of stack. For stack operation, physical address is produced by adding the contents of stack pointer register to the segment base address in SS. To do this the contents of the stack segment register are shifted four bits left and the contents of SP are added to the shifted result. If the contents of SP are 9F20H and SS are 4000H then the physical address is calculated as follows. (Refer Fig. 6.7)

SS = 4000H after shifting four bits left SS = 40000H

Now

SS	40000H
+ SP	9F20H
Physical address	49F20H

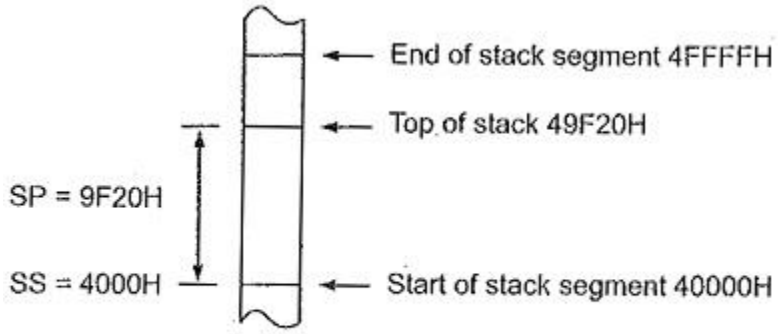

Fig. 6.7 Stack and stack pointer

Base Pointer, Source Index and Destination Index (BP, SI and DI)

These three 16-bit registers can be used as general purpose registers. However, their main use is to hold the 16-bit offset of the data word in one of the segments.

Base pointer : We can use the BP register instead of SP for accessing the stack using the based addressing mode. In this case, the 20-bit physical stack address is calculated from BP and SS. Addressing modes are discussed in later section.

Source Index : Source index (SI) can be used to hold the offset of a data word in the data segment. In this case, the 20-bit physical data address is calculated from SI and

Destination Index : The ES register .points to the extra segment in which data is stored. String instructions always use ES and DI to determine the 20-bit physical address for the destination.

Default and Alternate Register Assignments:

Table 6.1 shows that some memory references and their default and alternate segment definitions. For example, instruction codes can only be stored in the code segment with IP used as an offset. Similarly, for stack operations only SS and SP or BP registers can be used to give segment and offset addresses respectively. On the other hand, for accessing general data, string source; data pointed by BX and BP registers; it is possible to use alternate segments by using segment override prefix.

Type of Memory Reference	Default Segment	Alternate Segment	Offset (Logical Address)
Instruction fetch	CS	None	IP
Stack operation	SS	None	SP, BP
General data	DS	CS, ES, SS	Effective address
String source	DS	CS, ES, SS	SI
String destination	ES	None	DI
BX used as pointer	DS	CS, ES, SS	Effective Address
BP used as pointer	SS	CS, ES, DS	Effective Address