UNIT-I
INTRODUCTION TO 8086

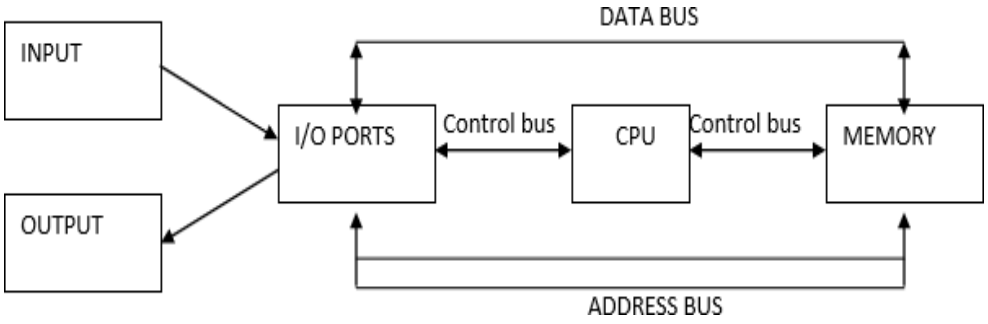
Contents to be covered

- ✓ Architecture of 8086 microprocessor
- ✓ Register organization
- ✓ 8086 flag register and its functions
- ✓ Addressing modes of 8086
- ✓ Pin diagram of 8086
- ✓ Minimum mode & Maximum mode system operation
- ✓ Timing diagrams

INTRODUCTION TO MICROPROCESSOR:

OVERVIEW OF A SIMPLE MICRO COMPUTER:

The major parts are the central processing unit or CPU, memory, and the input and output circuitry or I/O. Connecting these parts together are three sets of parallel lines called buses. The three buses are the address bus, the data bus, and the control bus.



Block diagram of simple computer or microcomputer.

i) MEMORY: The memory section usually consists of a mixture of RAM and ROM. It may also have magnetic floppy disks, magnetic hard disks, or laser optical disks. Memory has two purposes. The first purpose is to store the binary codes for the sequence of instructions you want the computer to carry out. When you write a computer program, what you are really doing is just writing a sequential list of instructions for the computer. The second purpose of the memory is to store the binary-coded data with which the computer is going to be working.

ii) INPUT/OUTPUT: The input/output or I/O section allows the computer to take in data from the outside world or send data to the outside world. These allow the user and the computer to communicate with each other. The actual physical devices used to interface the computer buses to external systems are often called ports.

iii) CPU: The central processing unit or CPU controls the operation of the computer. It fetches binary-coded instruction of the computer. It fetches binary-coded instructions from memory, decodes the instructions into a series of simple actions, and carries out these actions. The CPU contains an arithmetic logic unit, or ALU. Which can perform add, subtract, OR, AND, invert, or exclusive-OR operations on binary words when instructed to do so. The CPU also contains an address counter which is used to hold the address of the next instruction or data to be fetched from memory, general-purpose registers which are used for temporary storage of binary data, and circuitry which generates the control bus signals.

iv) ADDRESS BUS: The address bus consists of 16, 20, 24, or more parallel signal lines. On these lines the CPU sends out the address of the memory location that is to be written to or read from. The number of address lines determines the number of memory locations that the CPU can address. If the CPU has N address lines then it can directly address 2^N memory locations.

v) DATA BUS: The data bus consists of 8, 16, 32 or more parallel signal lines. As indicated by the double-ended arrows on the data bus line, the data bus lines are bi-directional. This means that the CPU can read data in on these lines from memory or from a port as well as send data out on these lines to memory location or to a port. Many devices in a system will have their outputs connected to the data bus, but the outputs of only one device at a time will be enabled.

vi) CONTROL BUS: The control bus consists of 4-10 parallel signal lines. The CPU sends out signals on the control bus to enable the outputs of addressed memory devices or port devices. Typical control bus signals are memory read, memory write, I/O read, and I/O writer. To read a byte of data from a memory location, for example, the CPU sends out the address of the desired byte on the address bus and then sends out a memory read signal on the control bus.

What is a Microprocessor?

- The word comes from the combination micro and processor.
  - Processor means a device that processes numbers, specifically binary numbers, 0's and 1's.
  - Micro is a new addition.
  - In the late 1960's, processors were built using discrete elements.
  - These devices performed the required operation, but were too large and too slow.
  - In the early 1970's the microchip was invented. All of the components that made up the processor were now placed on a single piece of silicon. The size became several thousand times smaller and the speed became several hundred times faster.
  - The "Micro" Processor was born.

Definition of Microprocessor:

- Microprocessor is a multipurpose, programmable device that accepts digital data as input, processes it according to instructions stored in its memory, and provides results as output.

or

- A microprocessor is a multipurpose, programmable, clock-driven, register-based electronic device that reads binary instructions from a storage device called memory accepts binary data as input and processes data according to instructions, and provides result as output.

MICROCONTROLLER:

- A **microcontroller** (sometimes abbreviated µC,or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.

or

- CPUs with integrated memory or peripheral interfaces

History of Microprocessors:

Processor	No. of bits	Clock speed (Hz)	Year of introduction
4004	4	740K	1971
8008	8	500K	1972
8080	8	2M	1974
8085	8	3M	1976

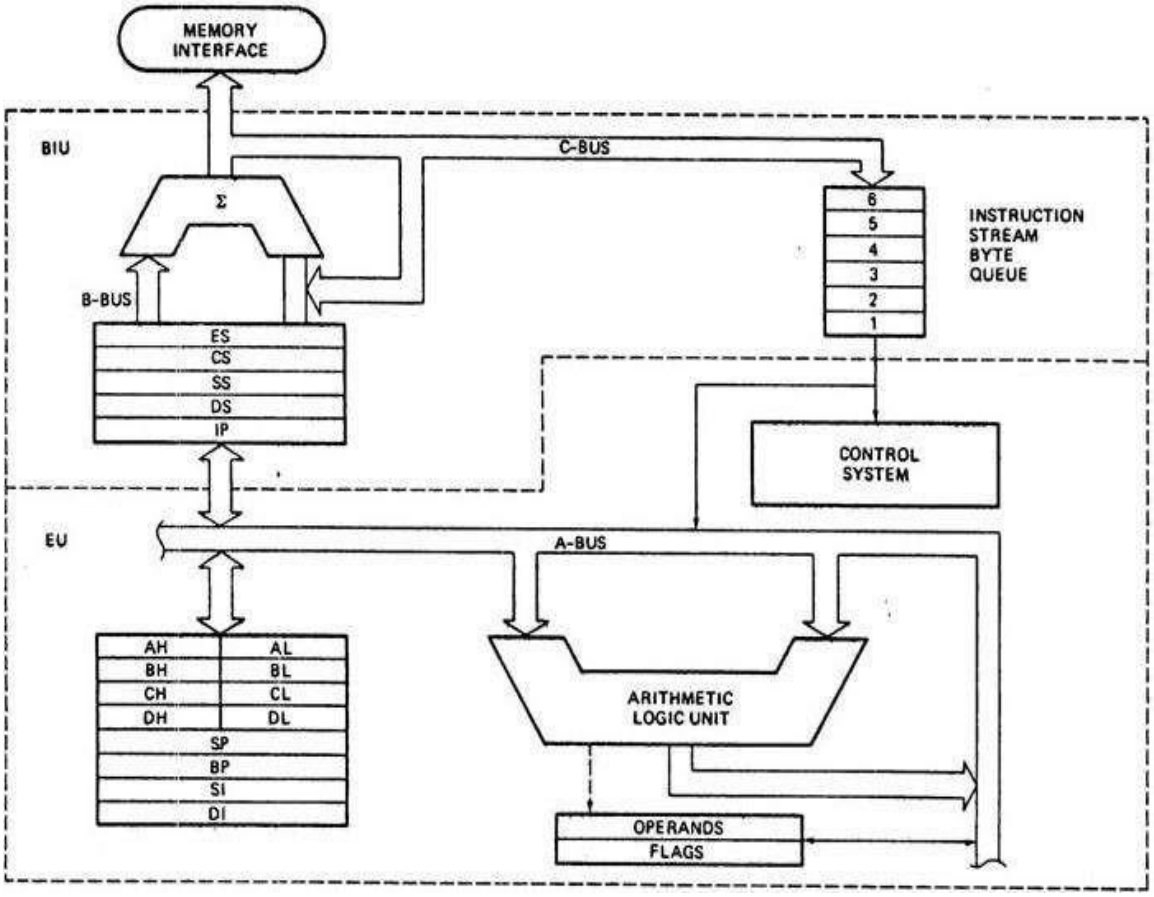
8086	16	5, 8 or 10M	1978
8088	16	5, 8 or 10M	1979
80186	16	6M	1982
80286	16	8M	1982
80386	32	16 to 33M	1986
80486	32	16 to 100M	1989
Pentium	32	66M	1993
Pentium II	32	233 to 500M	1997
Pentium III	32	500M to 1.4G	1999
Pentium IV	32	1.3 to 3.8G	2000
Dual core	32	1.2 to 3 G	2006
Core 2 Duo	64	1.2 to 3G	2006
i3, i5 and i7	64	2.4G to 3.6G	2010

8086 Microprocessor features:

1. It is 16-bit microprocessor
2. It has a 16-bit data bus, so it can read data from or write data to memory and ports either 16-bit or 8-bit at a time.
3. It has 20 bit address bus and can access up to 2^{20} memory locations (1MB).
4. It can support up to 64K I/O ports
5. It provides 14, 16-bit registers
6. It has multiplexed address and data bus AD_0-AD_{15} & A_{16}-A_{19}
7. It requires single phase clock with 33% duty cycle to provide internal timing.
8. Prefetches up to 6 instruction bytes from memory and queues them in order to speed up the processing.
9. 8086 supports 2 modes of operation
    a. Minimum mode
    b. Maximum mode

Architecture of 8086 microprocessor:

- As shown in the below figure, the 8086 CPU is divided into two independent functional parts
    - Bus Interface Unit (BIU)
    - Execution Unit (EU)
- Dividing the work between these two units' speeds up processing.



The Execution Unit (EU):

- The execution unit of the 8086 tells the BIU where to fetch instructions or data from, decodes instructions, and executes instructions.
- The EU contains **control circuitry,** which directs internal operations.
- A decoder in the EU translates instructions fetched from memory into a series of actions, which the EU carries out.
- The EU has a 16-bit **arithmetic logic unit** (ALU) which can add, subtract, AND, OR, XOR, increment, decrement, complement or shift binary numbers.
- The main functions of EU are:
    - Decoding of Instructions
    - Execution of instructions
        - ✓ Steps
            - EU extracts instructions from top of queue in BIU
            - Decode the instructions
            - Generates operands if necessary
            - Passes operands to BIU & requests it to perform read or write bus cycles to memory or I/O
            - Perform the operation specified by the instruction on operands

Bus Interface Unit (BIU):

- The BIU sends out addresses, fetches instructions from memory, reads data from ports and memory, and writes data to ports and memory.
- In simple words, the BIU handles all transfers of data and addresses on the buses for the execution unit.
- BIU interfaces with memory and I/O

8086 HAS PIPELINING ARCHITECTURE:

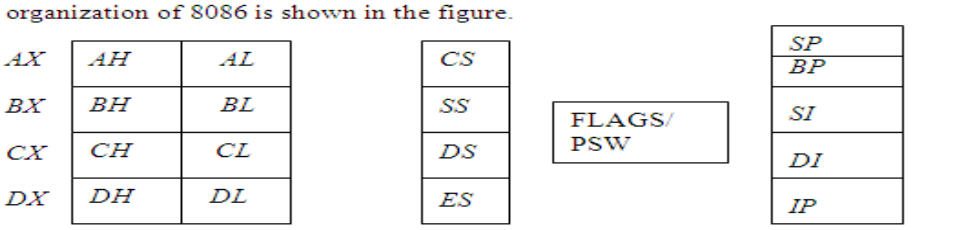
- While the EU is decoding an instruction or executing an instruction, which does not require use of the buses, the BIU fetches up to six instruction bytes for the following instructions.
- The BIU stores these pre-fetched bytes in a first-in-first-out register set called a *queue.*
- When the EU is ready for its next instruction from the queue in the BIU. This is much faster than sending out an address to the system memory and waiting for memory to send back the next instruction byte or bytes.
- Except in the case of JMP and CALL instructions, where the queue must be dumped and then reloaded starting from a new address, this pre-fetch and queue scheme greatly speeds up processing.
- Fetching the next instruction while the current instruction executes is called **pipelining**.

Register organization:

- 8086 has a powerful set of registers known as *general purpose registers* and *special purpose registers.*
- All of them are 16-bit registers.
- *General purpose registers:*
  - These registers can be used as either 8-bit registers or 16-bit registers.
  - They may be either used for holding data, variables and intermediate results temporarily or for other purposes like a counter or for storing offset address for some particular addressing modes etc.
- *Special purpose registers:*
  - These registers are used as segment registers, pointers, index registers or as offset storage registers for particular addressing modes.
- The 8086 registers are classified into the following types:
  - General Data Registers
  - Segment Registers
  - Pointers and Index Registers
  - Flag Register

General Data Registers:

The register set of 8086 can be categorized into 4 different groups. The register organization of 8086 is shown in the figure.



Register organization of 8086

- The registers *AX, BX, CX* and *DX* are the general purpose 16-bit registers.
- *AX* is used as 16-bit accumulator. The lower 8-bit is designated as *AL* and higher 8-bit is designated as *AH*. *AL* can be used as an 8-bit accumulator for 8-bit operation.
- All data register can be used as either 16 bit or 8 bit. *BX* is a 16 bit register, but *BL* indicates the lower 8-bit of *BX* and *BH* indicates the higher 8-bit of *BX*.
- The register *BX* is used as offset storage for forming physical address in case of certain addressing modes.
- The register *CX* is used default counter in case of string and loop instructions.
- *DX* register is a general-purpose register which may be used as an implicit operand or destination in case of a few instructions.

Segment Registers:

- There are 4 segment registers. They are:
  - Code Segment Register (CS)
  - Data Segment Register (DS)
  - Extra Segment Register (ES)
  - Stack Segment Register (SS)
- The 8086 architecture uses the concept of **segmented memory**. 8086 able to address a memory capacity of 1 megabyte and it is byte organized. This 1-megabyte memory is divided into 16 logical segments. Each segment contains 64 Kbytes of memory.
- Code segment register (CS): is used for addressing memory location in the code segment of the memory, where the executable program is stored.
- Data segment register (DS): points to the data segment of the memory where the data is stored.
- Extra Segment Register (ES) : also refers to a segment in the memory which is another data segment in the memory.
- Stack Segment Register (SS): is used for addressing stack segment of the memory. The stack segment is that segment of memory which is used to store stack data.
- While addressing any location in the memory bank, the **physical address** is calculated from two parts:

  Physical address= segment address + offset address
- The first is segment address, the segment registers contain 16-bit segment base addresses, related to different segment.
- The second part is the offset value in that segment.

Pointers and Index Registers:

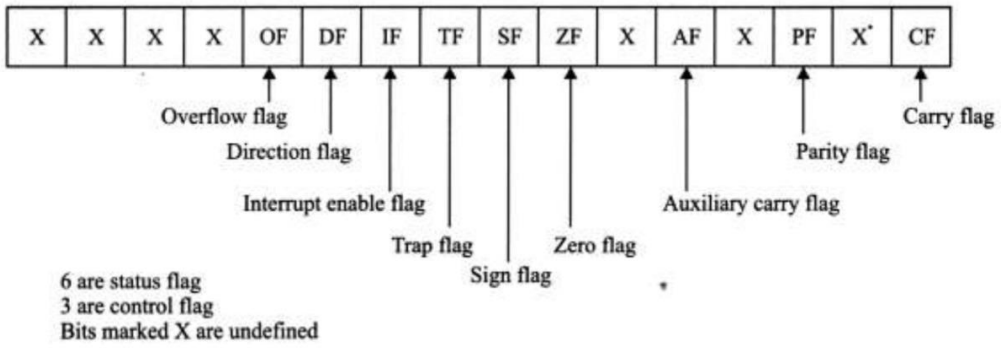
- The index and pointer registers are given below:
  - IP—Instruction pointer-store memory location of next instruction to be executed
  - BP—Base pointer
  - SP—Stack pointer
  - SI—Source index
  - DI—Destination index
- The pointers registers contain offset within the particular segments.
  - The pointer register *IP* contains offset within the code segment.
  - The pointer register *BP* contains offset within the data segment.
  - Thee pointer register *SP* contains offset within the stack segment.
- The index registers are used as general purpose registers as well as for offset storage in case of indexed, base indexed and relative base indexed addressing modes.
- The register *SI* is used to store the offset of source data in data segment.
- The register *DI* is used to store the offset of destination in data or extra segment.
- The index registers are particularly useful for string manipulation.

8086 flag register and its functions:

- The 8086 flag register contents indicate the results of computation in the *ALU*. It also contains some flag bits to control the *CPU* operations.
- A 16 bit flag register is used in 8086. It is divided into two parts .
  - Condition code or status flags
  - Machine control flags
- The **condition code flag register** is the lower byte of the 16-bit flag register. The condition code flag register is identical to 8085 flag register, with an additional overflow flag.

- The **control flag register** is the higher byte of the flag register. It contains three flags namely direction flag (*D*), interrupt flag (*I*) and trap flag (*T*).

Flag register configuration



The description of each flag bit is as follows:

***SF*- Sign Flag:** This flag is set, when the result of any computation is negative. For signed computations the sign flag equals the MSB of the result.

***ZF*- Zero Flag:** This flag is set, if the result of the computation or comparison performed by the previous instruction is zero.

***PF*- Parity Flag:** This flag is set to 1, if the lower byte of the result contains even number of 1's.

***CF*- Carry Flag:** This flag is set, when there is a carry out of MSB in case of addition or a borrow in case of subtraction.

***AF*-Auxilary Carry Flag:** This is set, if there is a carry from the lowest nibble, i.e, bit three during addition, or borrow for the lowest nibble, i.e, bit three, during subtraction.

***OF*- Over flow Flag:** This flag is set, if an overflow occurs, i.e, if the result of a signed operation is large enough to accommodate in a destination register. The result is of more than 7-bits in size in case of 8-bit signed operation and more than 15-bits in size in case of 16-bit sign operations, and then the overflow will be set.

***TF*- Tarp Flag:** If this flag is set, the processor enters the single step execution mode. The processor executes the current instruction and the control is transferred to the Trap interrupt service routine.

***IF*- Interrupt Flag:** If this flag is set, the mask able interrupts are recognized by the CPU, otherwise they are ignored.

***D*- Direction Flag:** This is used by string manipulation instructions. If this flag bit is '0', the string is processed beginning from the lowest address to the highest address, i.e., auto incrementing mode. Otherwise, the string is processed from the highest address towards the lowest address, i.e., auto decrementing mode.

Memory Segmentation:

- The memory in an 8086 based system is organized as segmented memory.
- The CPU 8086 is able to access 1MB of physical memory. The complete 1MB of memory can be divided into 16 segments, each of 64KB size and is addressed by one of the segment register.
- The 16-bit contents of the segment register actually point to the starting location of a particular segment. The address of the segments may be assigned as 0000H to F000h respectively.
- To address a specific memory location within a segment, we need an offset address. The offset address values are from 0000H to FFFFH so that the physical addresses range from 00000H to FFFFFH.

physical address is calculated as below:

Ex: Segment address ------- → 1005H

Offset address ----------→ 5555H

Segment address -------→ 1005H -------0001 0000 0000 0101

Shifted left by 4 Positions ------0001 0000 0000 0101 0000

+

Offset address --- 5555H ------ 0101 0101 0101 0101

Physical address -------155A5H 0001 0101 0101 1010 0101

Physical address = Segment address * 10H + Offset address.

The main advantages of the segmented memory scheme are as follows:

1. Allows the memory capacity to be 1MB although the actual addresses to be handled are of 16-bit size.
2. Allows the placing of code, data and stack portions of the same program in different parts (segments) of memory, for data and code protection.
3. Permits a program and/or its data to be put into different areas of memory each time the program is executed, i.e., provision for relocation is done.

Overlapping and Non-overlapping Memory segments:

- In the overlapping area locations physical address = CS1+IP1 = CS2+IP2. Where '+' indicates the procedure of physical address formation.

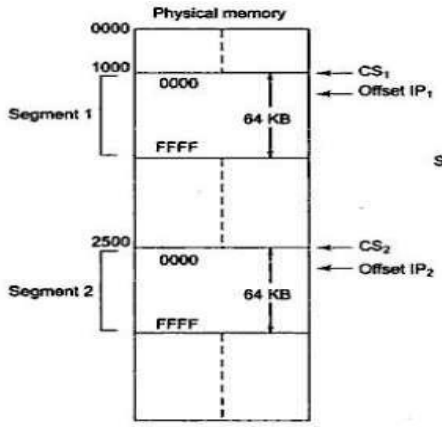


Fig. 1.3(a) *Non-overlapping Segments*

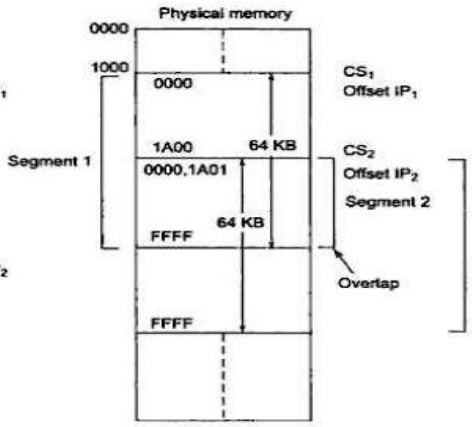


Fig. 1.3(b) *Overlapping Segments*