Microprocessor - 8086 Overview

8086 Microprocessor is an enhanced version of 8085 Microprocessor that was designed by Intel in 1976. It is a 16-bit Microprocessor having 20 address lines and 16 data lines that provides up to 1MB storage.

It consists of powerful instruction set, which provides operations like multiplication and division easily.

It supports two modes of operation, i.e. Maximum mode and Minimum mode. Maximum mode is suitable for system having multiple processors and Minimum mode is suitable for system having a single processor.

Features of 8086

The most prominent features of a 8086 microprocessor are as follows –

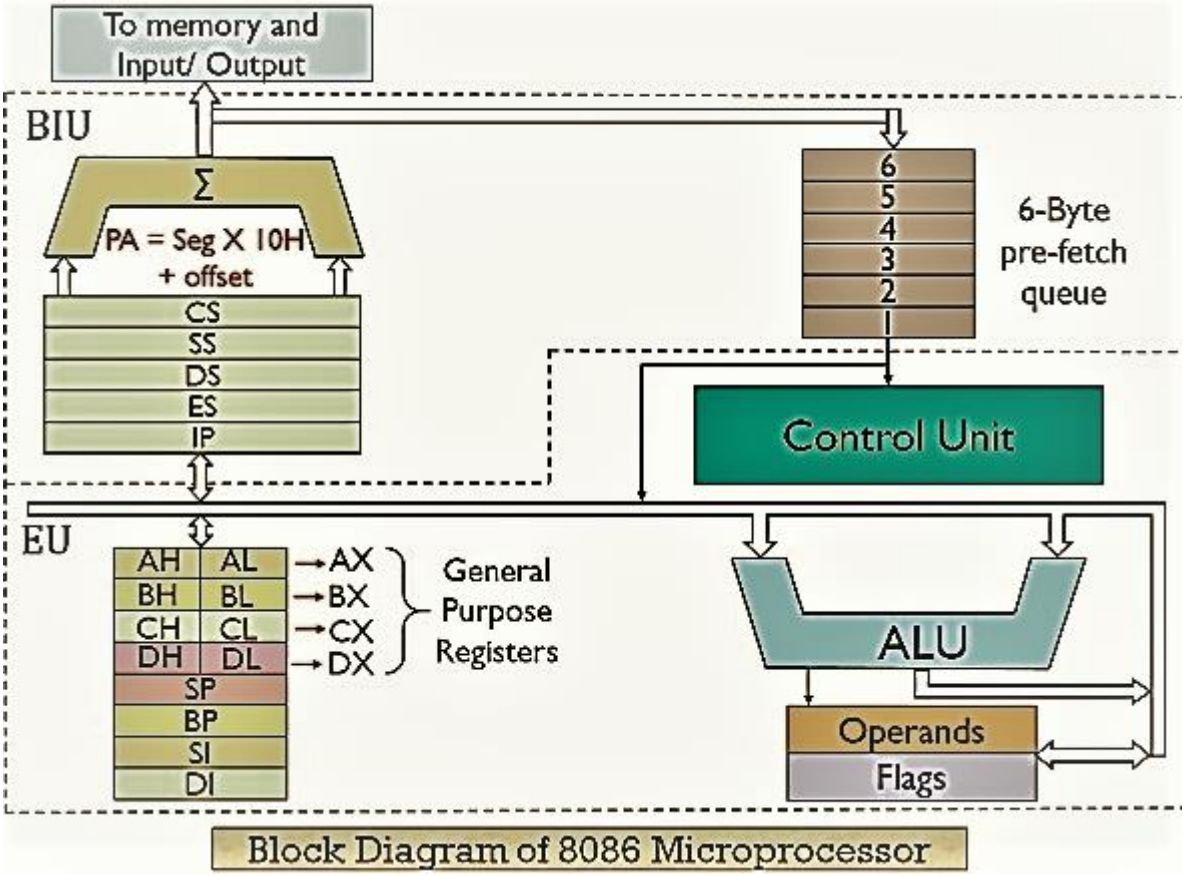
- It has an instruction queue, which is capable of storing six instruction bytes from the memory resulting in faster processing.
- It was the first 16-bit processor having 16-bit ALU, 16-bit registers, internal data bus, and 16-bit external data bus resulting in faster processing.
- It is available in 3 versions based on the frequency of operation –
    - 8086 → 5MHz
    - 8086-2 → 8MHz
    - 8086-1 → 10 MHz
- It uses two stages of pipelining, i.e. Fetch Stage and Execute Stage, which improves performance.
- Fetch stage can prefetch up to 6 bytes of instructions and stores them in the queue.
- Execute stage executes these instructions.
- It has 256 vectored interrupts.
- It consists of 29,000 transistors.

Comparison between 8085 & 8086 Microprocessor

- **Size** – 8085 is 8-bit microprocessor, whereas 8086 is 16-bit microprocessor.
- **Address Bus** – 8085 has 16-bit address bus while 8086 has 20-bit address bus.
- **Memory** – 8085 can access up to 64Kb, whereas 8086 can access up to 1 Mb of memory.
- **Instruction** – 8085 doesn't have an instruction queue, whereas 8086 has an instruction queue.
- **Pipelining** – 8085 doesn't support a pipelined architecture while 8086 supports a pipelined architecture.
- **I/O** – 8085 can address 2^8 = 256 I/O's, whereas 8086 can access 2^16 = 65,536 I/O's.
- **Cost** – The cost of 8085 is low whereas that of 8086 is high.

Architecture of 8086

The following diagram depicts the architecture of a 8086 Microprocessor –



Block Diagram of 8086 Microprocessor

8086 Microprocessor is divided into two functional units, i.e., **EU** (Execution Unit) and **BIU** (Bus Interface Unit).

EU (Execution Unit)

Execution unit gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU.

EU has no direct connection with system buses as shown in the above figure, it performs operations over data through BIU.

Let us now discuss the functional parts of 8086 microprocessors.

ALU

It handles all arithmetic and logical operations, like +, –, ×, /, OR, AND, NOT operations.

Flag Register

It is a 16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups – Conditional Flags and Control Flags.

Conditional Flags

It represents the result of the last arithmetic or logical instruction executed. Following is the list of conditional flags –

- **Carry flag** – This flag indicates an overflow condition for arithmetic operations.
- **Auxiliary flag** – When an operation is performed at ALU, it results in a carry/borrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), then this flag is set,
  i.e. carry given by D3 bit to D4 is AF flag.
- **Parity flag** – This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1's, then the Parity Flag is set. For odd number of 1's, the Parity Flag is reset.
- **Zero flag** – This flag is set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.
- **Sign flag** – This flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.
- **Overflow flag** – This flag represents the result when the system capacity is exceeded.

Control Flags

Control flags controls the operations of the execution unit. Following is the list of control flags –

- **Trap flag** – It is used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.
- **Interrupt flag** – It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.
- **Direction flag** – It is used in string operation. As the name suggests when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-a-versa.

General purpose register

There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pairs to store 16 bit data. The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively.

- **AX register** – It is also known as accumulator register. It is used to store operands for arithmetic operations.
  ADD AX, AX

- **BX register** – It is used as a base register. It is used to store the starting base address of the memory area within the data segment.
  MOV BL, [500]

- **CX register** – It is referred to as counter. It is used in loop instruction to store the loop counter.

  MOV CX, 0005

  LOOP

- **DX register** – This is the data register. It is of 16 bits and is divided into two 8-bit registers DH and DL to also perform 8-bit instructions. It is used in the multiplication and input/output port addressing
  MUL BX (DX, AX = AX * BX)

Stack pointer register

It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack.

Base pointer register

This is the base pointer. It is of 16 bits. It is primarily used in accessing parameters passed by the stack. Its offset address is relative to the stack segment.

Source Index register

This is the source index register. It is of 16 bits. It is used in the pointer addressing of data and as a source in some string-related operations. Its offset is relative to the data segment.

Destination Index register

This is the destination index register. It is of 16 bits. It is used in the pointer addressing of data and as a destination in some string-related operations. Its offset is relative to the extra segment.

BIU (Bus Interface Unit)

BIU takes care of all data and address transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direct connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.

It has the following functional parts –

- **Instruction queue** – BIU contains the instruction queue. BIU gets upto 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed.

- Fetching the next instruction while the current instruction executes is called **pipelining**.
- **Segment register** – BIU has 4 segment buses, i.e. CS, DS, SS & ES. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations. It also contains 1 pointer register IP, which holds the address of the next instruction to be executed by the EU.
    - **Code Segment Register** – CS holds the base address for the Code Segment. All programs are stored in the Code Segment and accessed via the IP.
    - **Data Segment Register**– DS holds the base address for the Data Segment.
    - **SS** – SS holds the base address for the Stack Segment.
    - **ES** – ES holds the base address for the Extra Segment. ES is additional data segment, which is used to hold the extra destination data.
- **Instruction pointer** – It is a 16-bit register used to hold the address of the next instruction to be executed.

- **Please note that segments are present in memory and segment registers are present in Microprocessor.**
- **Segment registers store starting address of each segment in memory.**