

8086 is a 16-bit microprocessor and was designed in 1976 by Intel. Unlike, 8085, an 8086 microprocessor has **20-bit address bus**. Thus, is able to access 220 i.e., 1 MB address in the memory.

**Bus Interface Unit (BIU)**

The segment registers, instruction pointer and 6-byte instruction queue are associated with the bus interface unit (BIU).

The BIU:

* Handles transfer of data and addresses,
* Fetches instruction codes, stores fetched instruction codes in first-in-first-out register set called a **queue**,
* Reads data from memory and I/O devices,
* Writes data to memory and I/O devices,
* It relocates addresses of operands since it gets un-relocated operand addresses from EU. The EU tells the BIU from where to fetch instructions or where to read data.

It has the following functional parts:

* **Instruction Queue:** When EU executes instructions, the BIU gets 6-bytes of the next instruction and stores them in the instruction queue and this process is known as instruction pre fetch. This process increases the speed of the processor.
* **Segment Registers:** A segment register contains the addresses of instructions and data in memory which are used by the processor to access memory locations. It points to the starting address of a memory segment currently being used.  
  There are 4 segment registers in 8086 as given below:
  + **Code Segment Register (CS):** Code segment of the memory holds instruction codes of a program.
  + **Data Segment Register (DS):** The data, variables and constants given in the program are held in the data segment of the memory.
  + **Stack Segment Register (SS):** Stack segment holds addresses and data of subroutines. It also holds the contents of registers and memory locations given in PUSH instruction.
  + **Extra Segment Register (ES):** Extra segment holds the destination addresses of some data of certain string instructions.
* **Instruction Pointer (IP):** The instruction pointer in the 8086 microprocessor acts as a program counter. It indicates to the address of the next instruction to be executed.

## **Execution Unit (EU)**

* The **EU** receives opcode of an instruction from the queue, decodes it and then executes it. While Execution, unit decodes or executes an instruction, then the BIU fetches instruction codes from the memory and stores them in the queue.
* The BIU and EU operate in parallel independently. This makes processing faster.
* General purpose registers, stack pointer, base pointer and index registers, ALU, flag registers (FLAGS), instruction decoder and timing and control unit constitute execution unit (EU). Let's discuss them:
* **General Purpose Registers:** There are four 16-bit general purpose registers: AX (Accumulator Register), BX (Base Register), CX (Counter) and DX. Each of these 16-bit registers are further subdivided into 8-bit registers as shown below:

|  |  |  |
| --- | --- | --- |
| **16-bit registers** | **8-bit high-order registers** | **8-bit low-order registers** |
| AX | AH | AL |
| BX | BH | BL |
| CX | CH | CL |
| DX | DH | DL |

* **Index Register:** The following four registers are in the group of pointer and index registers:
  + Stack Pointer (SP)
  + Base Pointer (BP)
  + Source Index (SI)
  + Destination Index (DI)
* **ALU:** It handles all arithmetic and logical operations. Such as addition, subtraction, multiplication, division, AND, OR, NOT operations.
* **Flag Register:** It is a 16?bit register which exactly behaves like a flip-flop, means it changes states according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups i.e. conditional and control flags.
  + **Conditional Flags/Status Flags:** This flag represents the result of the last arithmetic or logical instruction executed. Conditional flags are:
    - Carry Flag
    - Auxiliary Flag
    - Parity Flag
    - Zero Flag
    - Sign Flag
    - Overflow Flag
  + **Control Flags:** It controls the operations of the execution unit. Control flags are:
    - Trap Flag
    - Interrupt Flag
    - Direction Flag

| **Bits** | **D15** | **D14** | **D13** | **D12** | **D11** | **D10** | **D9** | **D8** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Flags** |  |  |  |  | O | D | I | T | S | Z |  | AC |  | P |  | CY |

| **Flag Bit** | **Function** |
| --- | --- |
| **S** | After any operation if the MSB is 1, then it indicates that the number is negative. And this flag is set to 1 |
| **Z** | If the total register is zero, then only the Z flag is set |
| **AC** | When some arithmetic operations generates carry after the lower half and sends it to upper half, the AC will be 1 |
| **P** | This is even parity flag. When result has even number of 1, it will be set to 1, otherwise 0 for odd number of 1s |
| **CY** | This is carry bit. If some operations are generating carry after the operation this flag is set to 1 |
| **O** | The overflow flag is set to 1 when the result of a signed operation is too large to fit. |

## **Control Flags**

| **Flag Bit** | **Function** |
| --- | --- |
| **D** | This is directional flag. This is used in string related operations. D = 1, then the string will be accessed from higher memory address to lower memory address, and if D = 0, it will do the reverse. |
| **I** | This is interrupt flag. If I = 1, then MPU will recognize the interrupts from peripherals. For I = 0, the interrupts will be ignored |
| **T** | This trap flag is used for on-chip debugging. When T = 1, it will work in a single step mode. After each instruction, one internal interrupt is generated. It helps to execute some program instruction by instruction. |