# UNIT-2 INTRODUCTION TO 8086

# 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

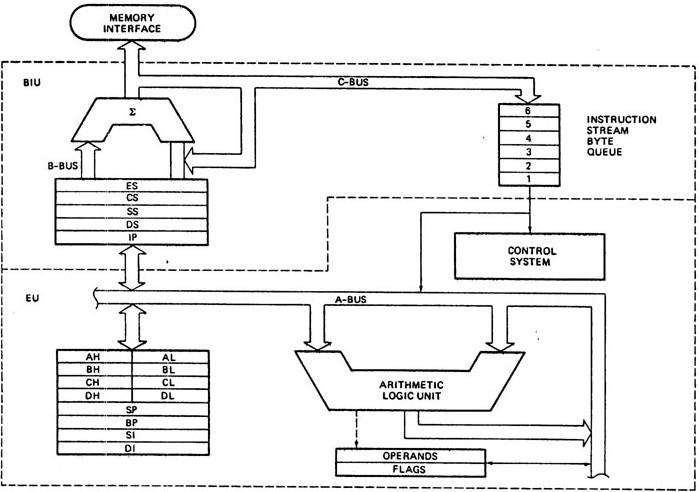
* A microprocessor is a multipurpose, programmable, clock-driven, register-based electronic device that reads binary instructions from a storage device called memory accepts binary data as input and processes data according to instructions, and provides result as output.

**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 220 memory locations (1 MB).
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 AD0-AD15 & A16-A19
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
   1. Minimum mode
   2. 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):

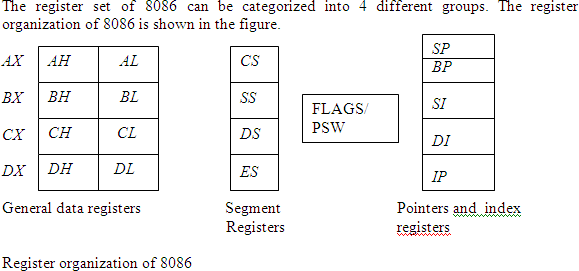
* The BIU sends out addresses, fetches instructions from memory, reads data from ports and memory, and writes data to ports and memory.
* In simple words, the BIU handles all transfers of data and addresses on the buses for the execution unit.

# 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 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.

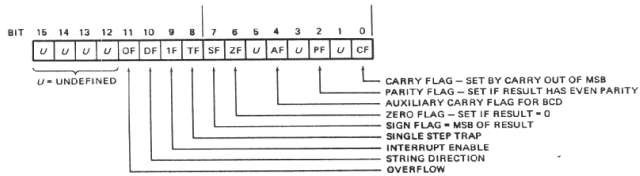
# Pointers and Index Registers:

* 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.

**Comparison between 8085 and 8086 microprocessor**

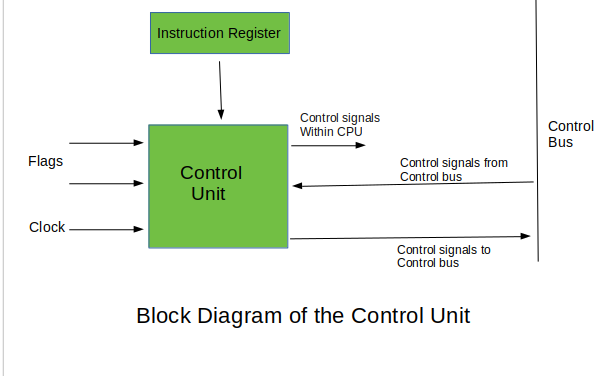
|  |  |  |
| --- | --- | --- |
| **Functions** | **8085** | **8086** |
| **size** | It is 8-bit microprocessor | It is 16-bit micro processor |
| **Address bus** | It has 16-bit address bus | It has 20-bit address bus |
| **Memory** | It can access up to 64kb | It can access up to 1 Mb of memory |
| **Instruction** | It doesn’t have an instruction queue | It has an Instruction queue |
| **Pipelining** | It doesn’t support a pipelined architecture | It supports a pipelined architecture |
| **I/O operations** | It can access 256 I/O’s operations | It can access 65,536 I/O’s operations |
| **cost** | The cost is low | The cost is high |

**CPU control unit design:** Hardwired and micro-programmed design approaches, design of a

Simple hypothetical CPU.

**Control Unit**

* The Control Unit is the part of the computer’s central processing unit (CPU), which directs the operation of the processor.
* Control unit generates timing and control signals for the operations of the computer. The control unit communicates with ALU and main memory. It also controls the transmission between processor, memory and the various peripherals. It also instructs the ALU which operation has to be performed on data.

****

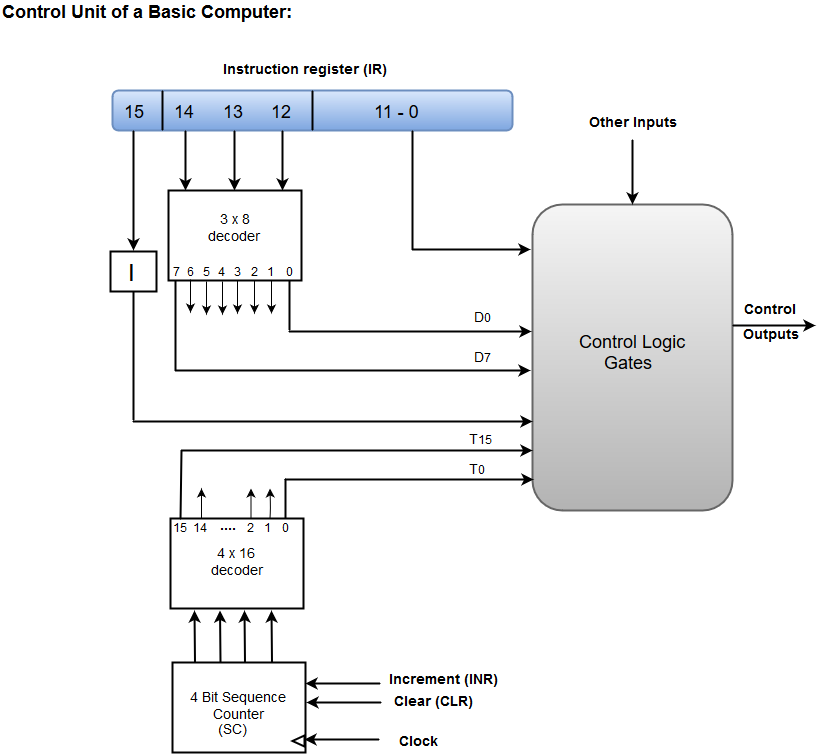
**Functions of the Control Unit:**

* It coordinates the sequence of data movements into, out of, and between a processor’s many sub-units.
* It interprets instructions.
* It controls data flow inside the processor.
* It receives external instructions or commands to which it converts to sequence of control signals.
* It controls many execution units (i.e. [ALU](https://www.geeksforgeeks.org/introduction-of-alu-and-data-path/), data buffers and[registers](https://www.geeksforgeeks.org/different-classes-of-cpu-registers/)) contained within a CPU.
* It also handles multiple tasks, such as fetching, decoding, execution handling and storing results.

**The Control Unit is classified into two major categories:**

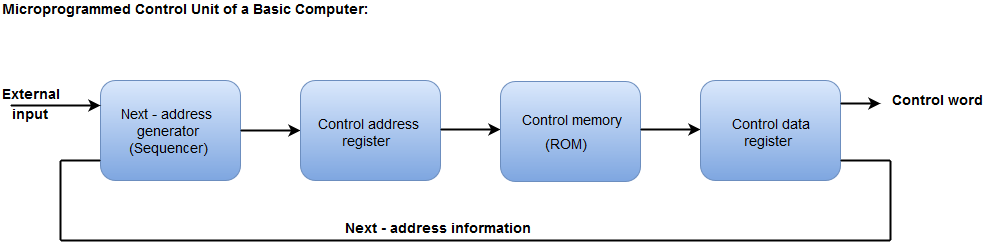
* Hardwired Control
* Micro programmed Control

**1) Hardwired Control unit:** *The Hardwired Control organization involves the control logic to be implemented with gates, flip-flops, decoders, and other digital circuits.*



* A Hard-wired Control consists of two decoders, a sequence counter, and a number of logic gates.
* An instruction fetched from the memory unit is placed in the instruction register (IR).
* The component of an instruction register includes; I bit, the operation code, and bits 0 through 11.
* The operation code in bits 12 through 14 are coded with a 3 x 8 decoder.
* The outputs of the decoder are designated by the symbols D0 through D7.
* The operation code at bit 15 is transferred to a flip-flop designated by the symbol I.
* The operation codes from Bits 0 through 11 are applied to the control logic gates.
* The Sequence counter (SC) can count in binary from 0 through 15.

**2) Micro programmed Control unit:**

****

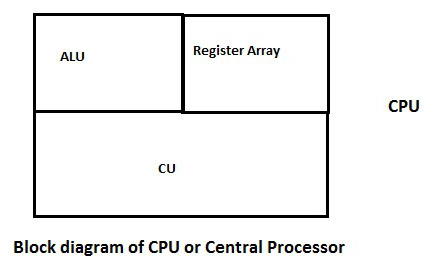
* The Micro programmed Control organization is implemented by using the programming approach.
* In Micro programmed Control, the micro-operations are performed by executing a program consisting of micro-instructions.
* The Control memory address register specifies the address of the micro-instruction.
* The Control memory is assumed to be a ROM, within which all control information is permanently stored.
* The control register holds the microinstruction fetched from the memory.
* The micro-instruction contains a control word that specifies one or more micro-operations for the data processor.
* While the micro-operations are being executed, the next address is computed in the next address generator circuit and then transferred into the control address register to read the next microinstruction.
* The next address generator is often referred to as a micro-program sequencer, as it determines the address sequence that is read from control memory.

**Difference between Hardwired Control and Micro programmed Control Unit:**

|  |  |
| --- | --- |
| **Hardwired Control Unit** | **Micro programmed Control Unit** |
| Technology is circuit based. | Technology is software based |
| It is implemented through flip-flops, gates, decoders etc. | Microinstructions generate signals to control the execution of instructions. |
| Fixed instruction format. | Variable instruction format (16-64 bits per instruction). |
| Instructions are register based. | Instructions are not register based |
| ROM is not used. | ROM is used. |
| It is used in RISC. | It is used in CISC. |
| Faster decoding. | Slower decoding |
| Difficult to modify. | Easily modified. |
| Chip area is less. | Chip area is large. |

**Design of a simple Hypothetical CPU:**

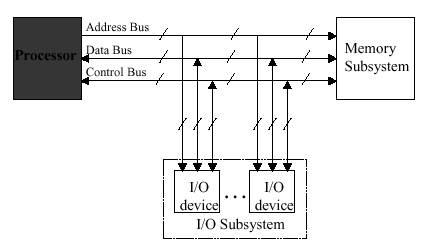
* **A computer has three main components**  
  1) Central Processing Unit (CPU) or Central Processor  
  2) Memory  
  3) I/O Devices  
  **The CPU in turn has three parts**  
  1) Arithmetic and Logic Unit (ALU)  
  2) Registers  
  3) Control Unit (CU)
* **Arithmetic and Logic Unit (ALU)**  
  The ALU performs the following operations: Addition, Subtraction, Logical AND, Logical OR, Logical XOR, Complement, Increment, Decrement, Left Shift, Clear.
* **Registers**  
  This is a small memory unit. Registers are used by the processor for temporary storage and manipulation of data and instructions. A register is a set of flip-flop. A flip-flop is an electronic circuit, which at any point of time stores either 0 or 1, which is any of the two states of a switch ON or OFF.  
  A register is mostly of different sizes and capacities: 8 bit, 16 bit, 32 bit, etc. Each register has a specific function in the CPU.



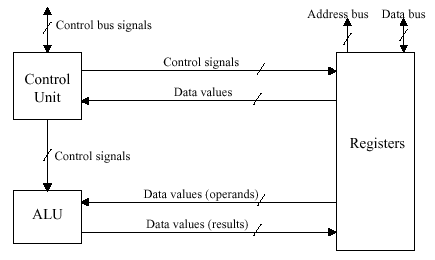
* **Given below are few commonly known registers:**  
  **Accumulator (AC)**  
  the ALU requires temporary registers or memory locations for all its operations. An accumulator is one of the main registers of the ALU, used to store data and perform arithmetic and logic operations. The results of the operations are stores automatically in this register.  
  **Program Counter (PC)**  
  A PC is used as a memory pointer. It stores the address of the next instruction to be executed. This register is used to sequence the execution of instructions.  
  **Instruction Register (IR)**  
  An IR holds the instruction until it is decoded.  
  **Stack Pointer (SP)**  
  the address of a stack top is held in the stack pointer. A stack is a sequence of memory locations. It is used to save the contents of a register during the execution of a program. The memory location of an occupied potion is known as stack top.
* **Given below are some of the registers for a basic computer and their functions:**Symbol     Name                                    Function  
  DR            Data Register                         Holds memory operand  
  AR            Address Register                   Holds address for memory  
  AC            Accumulator                         Processor Register  
  IR              Instruction Register               Holds instruction code  
  PC             Program Counter                 Holds address of next instruction  
  TR             Temporary Register              Holds Temporary data  
  INPR         Input Register                      Holds Input Character  
  OUTPR     Output Register                    Holds Output Character  
  **Control Unit (CU)**  
  This circuit is responsible for the entire amount of functions of the ALU. It receives instructions from memory and executes them after decoding them. Timing and control signals are generated by this circuit and sent to other circuits for the execution of the any program. It also transfers data between memory and I/O devices.
* **The Main purpose of Central Processing Unit**

1. It control the system
2. It performs data movements
3. It have the data processing capability
4. It have the data storage functionality.

**Generic Computer Organization:**

****

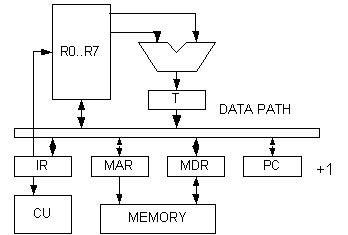
**CPU Internal Organization:**

****

* The design of a hypothetical processor can consists of an ALU to operate on the Operands, accumulator used to store one of the operands to be processed by ALU and also store the partial result and a temporary register, register file containing four general purpose registers.
* The processor can perform processing, Calculation and decisions.
* The CPU should be capable of performing basic arithmetic and logical operations, as well as accessing and manipulating data in memory. The system should have a minimal Instruction Set Architecture (ISA) and be optimized for power efficiency.

**S1 simple CPU:**

The design of a simple hypothetical CPU called S1.  S1 contains all the important elements of a real processor



**Instruction set design:**

* 16 bit word, fixed length
* address space 1024 words
* load, store architecture
* 8 registers, R7 as stack pointer
* condition code Z,S  zero, sign

Hypothetical system uses a simple Reduced Instruction Set computer (RISC).

**ISA with the following instructions:**

1. Load word (LW): load a 32 bit word from memory in to a register.

2. Store Word (SW): store a 32 bit word from register to memory.

3. ADD (add): Add two registers and store the result in a third register.

4. Subtract (sub): Subtract one register from another and store the result in a third register.

5. AND (and): perform a bitwise AND operation on two registers and store the result in a third register.

* OR (or): Perform a bitwise OR operation two registers and store the result in a third register.
* Jump (j): jump to a target address.
* Data path: It is the part of the CPU that performs ALU operations.

Simple CPU use a 3 stages pipeline.

1. Instruction Fetch (IF)

2. Instruction Decode (ID)

3. Execute (EX)

Data path Components: Program counter (PC), Instruction Memory (IM), Register file, ALU, Data memory.