# 8086 Microprocessor

- It is a 16-bit µp.
- 8086 has a 20 bit address bus can access up to 220 memory locations (1 MB).
- It can support up to 64K I/O ports.
- It provides 14, 16 -bit registers.
- It has multiplexed address and data bus AD0- AD15 and A16 – A19.
- It requires single phase clock with 33% duty cycle to provide internal timing.
- 8086 is designed to operate in two modes, Minimum and Maximum.
- It can prefetches up to 6 instruction bytes from memory and put them in instr queue in order to speed up instruction execution.
- It requires +5V power supply.
- A 40 pin dual in line package

# Architectural Diagram of 8086:

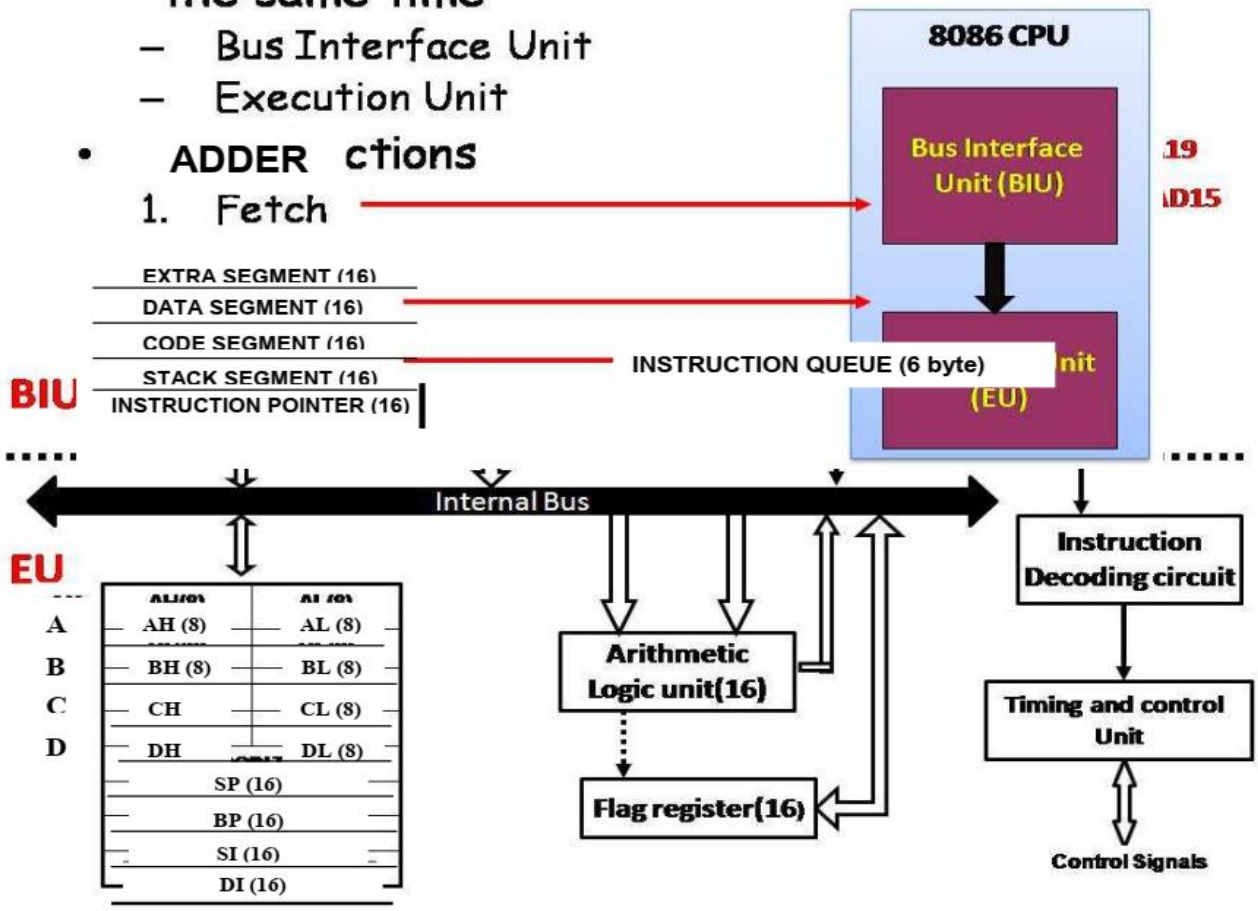
The 8086 has two parts, the Bus Interface Unit (BIU) and the Execution Unit (EU).

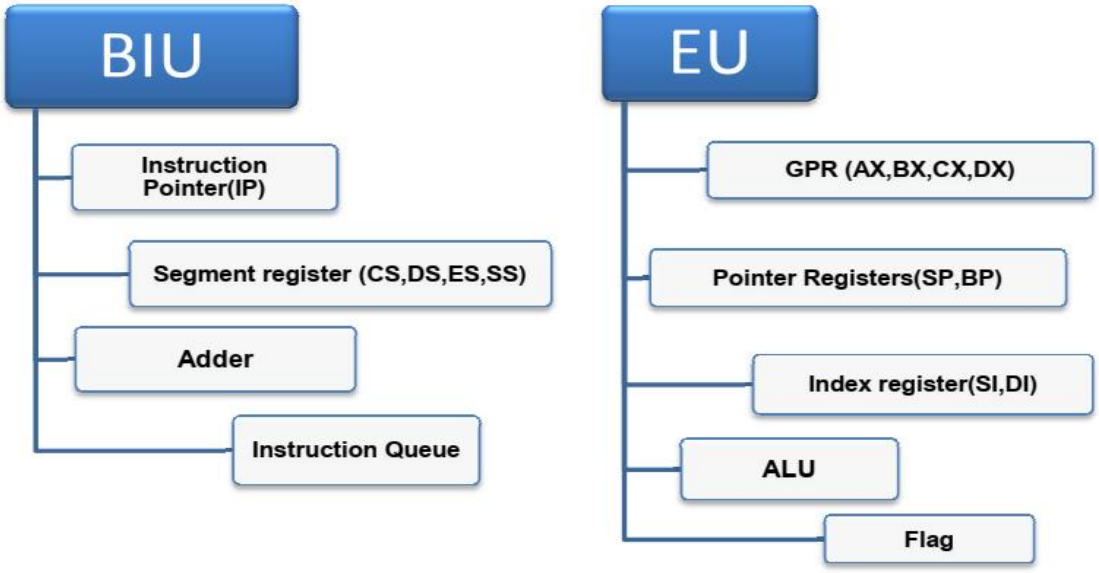
- The BIU fetches instructions, reads and writes data, and computes the 20-bit address.
- The EU decodes and executes the instructions using the 16-bit ALU.
- The two units functions independently.

## Minimum and Maximum Modes:

- The minimum mode is selected by applying logic 1 to the MN / $\overline{MX}$ input pin. This is a single microprocessor configuration.
- The maximum mode is selected by applying logic 0 to the MN / $\overline{MX}$ input pin. This is a multi micro processors configuration.

- 8086 employs parallel processing
- 8086 CPU has two parts which operate at the same time
  - Bus Interface Unit
  - Execution Unit
- ADDER ctions
  1. Fetch



# Bus Interface Unit (BIU):

- The BIU performs all bus operations such as instruction fetching, reading and writing operands for memory and calculating the addresses of the memory operands.
- The instruction bytes are transferred to the instruction queue.
- It provides a full 16 bit bidirectional data bus and 20 bit address bus.
- The bus interface unit is responsible for performing all external bus operations.

***Specifically it has the following functions***:

- Instruction fetch , Instruction queuing, Operand fetch and storage, Address calculation relocation and Bus control.
- The BIU uses a mechanism known as an instruction queue to implement a ***pipeline architecture.***
- This queue permits prefetch of up to six bytes of instruction code. Whenever the queue of the BIU is not full and it has room for at least two more bytes and at the same time EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by prefetching the next sequential instruction.
- These prefetching instructions are held in its FIFO queue. With its 16 bit data bus, the BIU fetches two instruction bytes in a single memory cycle.
- After a byte is loaded at the input end of the queue, it automatically shifts up through the FIFO to the empty location nearest the output.
- The EU accesses the queue from the output end. It reads one instruction byte after the other from the output of the queue. If the queue is full and the EU is not requesting access to operand in memory.
- These intervals of no bus activity, which may occur between bus cycles, are known as ***Idle state***.
- If the BIU is already in the process of fetching an instruction when the EU request it to read or write operands from memory or I/O, the BIU first completes the instruction fetch bus cycle before initiating the operand read / write cycle.
- The BIU also contains a dedicated **adder** which is used to generate the 20bit physical address that is output on the address bus. This address is formed by adding an appended 16 bit segment address and a 16 bit offset address.
- For example: The physical address of the next instruction to be fetched is formed by combining the current contents of the code segment CS register and the current contents of the instruction pointer IP register.

# EXECUTION UNIT (EU)

- The Execution unit is responsible for decoding and executing all instructions.

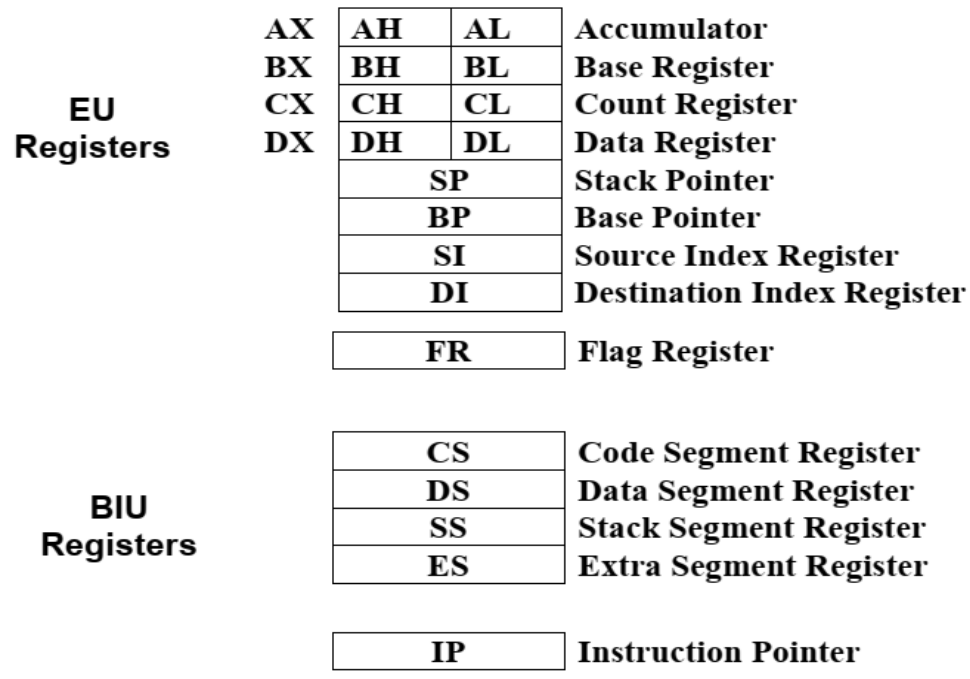
- The EU extracts instructions from the top of the queue in the BIU, decodes them, generates operands if necessary, passes them to the BIU and requests it to perform the read or write bys cycles to memory or I/O and perform the operation specified by the instruction on the operands.
- During the execution of the instruction, the EU tests the status and control flags and updates them based on the results of executing the instruction.
- If the queue is empty, the EU waits for the next instruction byte to be fetched and shifted to top of the queue.
- When the EU executes a branch or jump instruction, it transfers control to a location corresponding to another set of sequential instructions.
- Whenever this happens, the BIU automatically resets the queue and then begins to fetch instructions from this new location to refill the queue

**The BIU contains the following registers**:

IP - the Instruction Pointer
CS - the Code Segment Register
DS - the Data Segment Register
SS - the Stack Segment Register
ES - the Extra Segment Register

The BIU fetches instructions using the CS and IP, written CS:IP, to contract the 20-bit address. Data is fetched using a segment register (usually the DS) and an effective address (EA) computed by the EU depending on the addressing mode.

# Internal Registers of 8086



- The 8086 has four groups of the user accessible internal registers.
- These are
    - Instruction pointer(IP)
    - Four General purpose registers(AX,BX,CX,DX)
    - Four pointer (SP,BP,SI,DI)
    - Four segment registers (CS,DS,SS,ES)
    - Flag Register(FR)

- The 8086 has a total of fourteen 16-bit registers including a 16 bit register called the ***status register (flag register)***, with 9 of bits implemented for status and control flags.

- Most of the registers contain data/instruction offsets within 64 KB memory segment.

- There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor addressable memory these 4 segments are located the processor uses four segment registers:

# Segment Registers

1) **Code segment** (CS) is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register.

2) **Stack segment** (SS) is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction.

3) **Data and Extra segment** (DS and ES) is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, and DX) and index register (SI, DI) is located in the data and Extra segment.

# Data Registers

1) **AX (Accumulator)**
    - It is consists of two 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX. AL in this case contains the low-order byte of the word, and AH contains the high-order byte. Accumulator can be used for I/O operations and string manipulation.
2) **BX (Base register)**

- It is consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX. BL in this case contains the low-order byte of the word, and BH contains the high-order byte.
- BX register usually contains a offset for data segment.

**3) CX (Count register)**

- It is consists of two 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX. When combined, CL register contains the low-order byte of the word, and CH contains the high-order byte.
- Count register can be used in Loop, shift/rotate instructions and as a counter in string manipulation.
- 8086 has the LOOP instruction which is used for conuter purpose when it is executed CX/CL is automatically decremented by 1.

*EX*

```
        MOV CL, 05H
START   NOP
        LOOP START   (here CL is automatically decremented by 1without
                      DCR instruction.
```

**4) DX (Data register)**

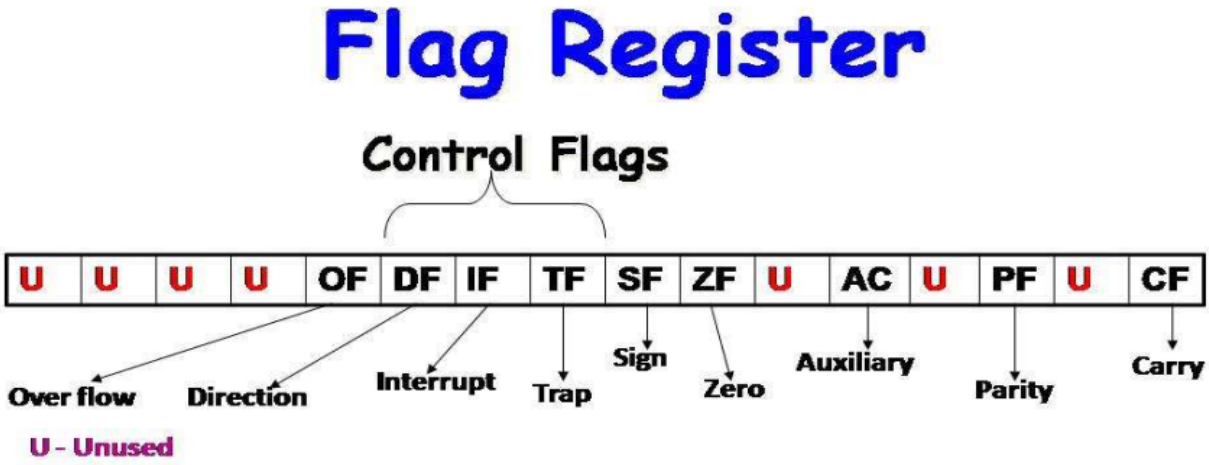
- It is consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX. When combined, DL register contains the low-order byte of the word, and DH contains the high-order byte.
- DX can be used as a port number in I/O operations.
- In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.

# Pointer register

1. **Stack Pointer** (SP) is a 16-bit register is used to hold the offset address for stack segment.
2. **Base Pointer** (BP) is a 16-bit register is used to hold the offset address for stack segment.
   i. BP register is usually used for based, based indexed or register indirect addressing.
   ii. The difference between SP and BP is that the SP is used internally to store the address in case of interrupt and the CALL instrn.
3. **Source Index** (SI) and **Destination Index** (DI )
   These two 16-bit register is used to hold the offset address for DS and ES in case of string manipulation instrn.
   i. SI is used for indexed, based indexed and register indirect addressing, as well as a source data addresses in string manipulation instructions.
   ii. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data addresses in string manipulation instructions.

**Instruction Pointer** (IP)

It is a 16-bit register. It acts as a program counter and is used to hold the offset address for CS.




**A flag is** a 16-bit register containing 9 one bit flags.

i. **Overflow Flag** (OF)
   - This flag is set if an overflow occurs. i.e. if the result of a signed operation is large enough to be accommodated in a destination register.

ii. **Direction Flag** (DF) –
   - 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.

iii. **Interrupt-enable Flag** (IF) –
   - If this flag is set, the maskable interrupts are recognized by the CPU. Otherwise they are ignored. Setting this bit enables maskable interrupts.

iv. **Single-step Flag** (TF) –
   - If this flag is set, the processor enters the single step execution mode. In other words, a trap interrupt is generated after execution of each instruction. The processor executes the current instruction and the control is transferred to the Trap interrupt service routine.

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

vi. **Zero Flag** (ZF) - set if the result is zero.

vii. **Auxiliary carry Flag** (AF) –
   - set if there was a carry from or borrow to bits 0-3 in the AL register.

viii. **Parity Flag** (PF) –
   - set if parity (the number of "1" bits) in the low-order byte of the result is even.

ix. **Carry Flag** (CF) –
   - This flag is set when there is a carry out of MSB in case of addition or a borrow in case of subtraction. For example. When two numbers are added, a carry may

be generated out of the most significant bit position. The carry flag, in this case, will be set to 1'. In case, no carry is generated, it will be '0.