8086 Microprocessor

- A **Microprocessor** is an Integrated Circuit with all the functions of a CPU. it has no memory or peripherals.
- 8086 does not have a RAM or ROM inside it. However, it has *internal registers* for storing intermediate and final results and interfaces with memory located outside it through the System Bus.
- In the case of 8086, it is a 16-bit **Integer processor** in a 40-pin.
- The size of the internal registers (present within the chip) indicates how much information the processor can operate on at a time (*in this case 16-bit registers*) and how it moves data around internally within the chip, sometimes also referred to as the internal data bus.

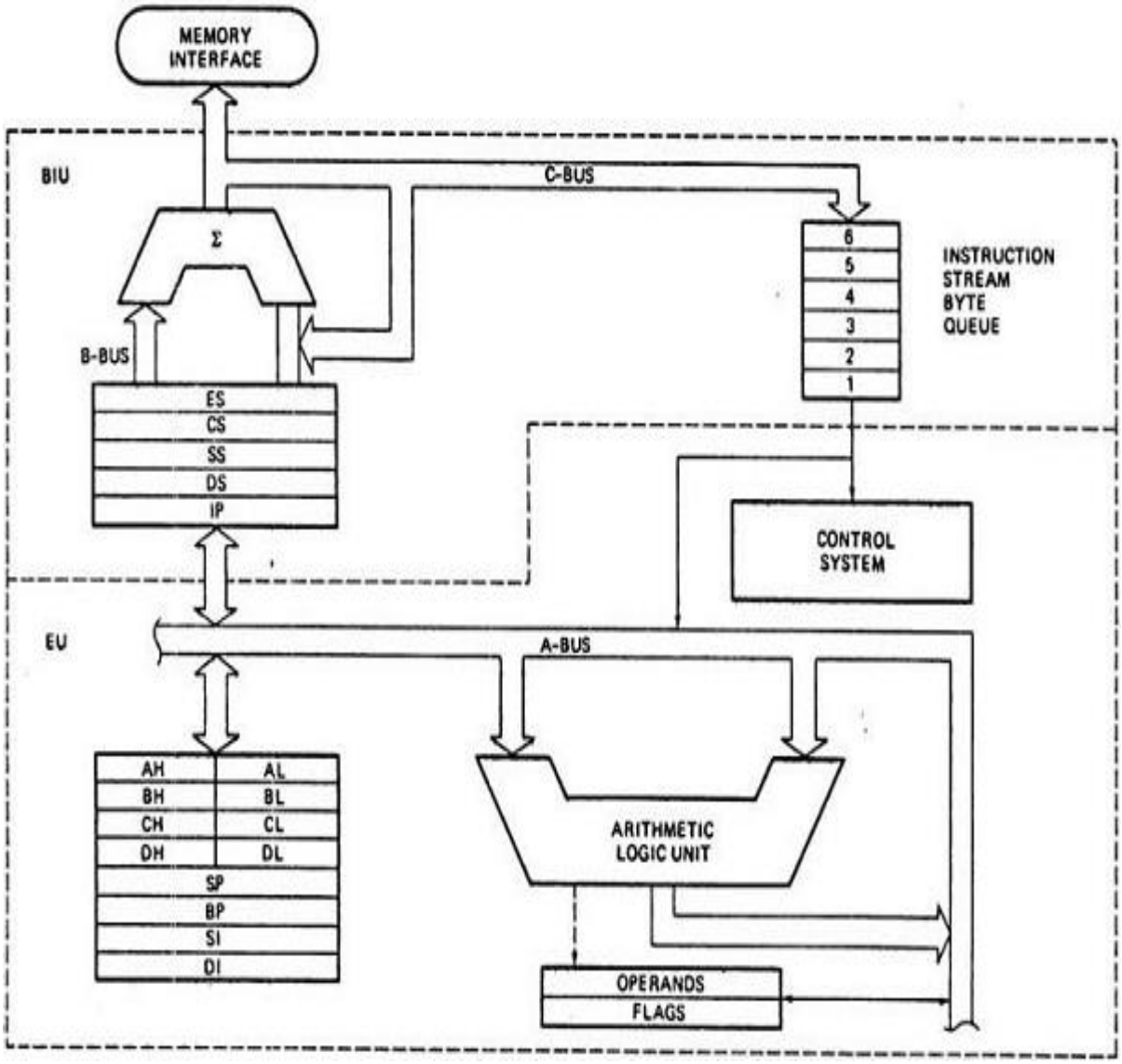


Fig1: 8086 Architecture

- 8086 provides the programmer with 14 internal registers, each of 16 bits or 2 bytes wide.
- **The main advantage of the 8086 microprocessor is that it supports Pipelining.**

The internal architecture of Intel 8086 is divided into 2 units: **The Bus Interface Unit (BIU)**, and **The Execution Unit (EU)**. These are explained as following below.

1. The Bus Interface Unit (BIU):

It provides the interface of 8086 to external memory and I/O devices via the System Bus. It performs various machine cycles such as memory read, I/O read, etc. to transfer data between memory and I/O devices.

BIU performs the following functions are as follows:

- It generates the 20-bit physical address for memory access.
- It fetches instructions from the memory.
- It transfers data to and from the memory and I/O.
- Maintains the 6-byte pre-fetch instruction queue(**supports pipelining**).

BIU mainly contains the **4 Segment registers**, the **Instruction Pointer**, a pre-fetch queue, and an **Address Generation Circuit**.

- **Instruction Pointer (IP):**

- It is a *16-bit register*. It holds offset of the next instructions in the *Code Segment.*
- IP is incremented after every instruction byte is fetched.
- IP gets a new value whenever a branch instruction occurs.

- **4 Segment registers**,
  - **Code Segment register: (16 Bit 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: (16 Bit register):** DS holds the base address for the Data Segment.
  - **Stack Segment register: (16 Bit register):** SS holds the base address for the Stack Segment.
  - **Extra Segment register: (16 Bit register):** ES holds the base address for the Extra Segment.

Address Generation Circuit:

- The BIU has a Physical Address Generation Circuit.
- It generates the 20-bit physical address using Segment and Offset addresses using the formula:
- In Bus Interface Unit (BIU) the circuit shown by the Σ symbol is responsible for the calculation unit which is used to calculate the physical address of an instruction in memory.

6 Byte Pre-fetch Queue:

- It is a 6-byte queue (FIFO).
- Fetching the next instruction (by BIU from CS) while executing the current instruction is called pipelining.
- Gets flushed whenever a branch instruction occurs.
- The pre-Fetch queue is of 6-Bytes only because the maximum size of instruction that can have in 8086 is 6 bytes. Hence to cover up all operands and data fields of maximum size instruction in 8086 Microprocessor there is a Pre-Fetch queue is 6 Bytes.
- The pre-Fetch queue is connected with the control unit which is responsible for decoding op-code and operands and telling the execution unit what to do with the help of timing and control signals.
- The pre-Fetch queue is responsible for pipelining and because of that 8086 microprocessor is called **fetch, decode, execute type microprocessor.** Since there are always instructions present for decoding and execution in this queue the speed of execution in the microprocessor is gradually increased.
- **When there is a 2-byte space in the instruction pre-fetch queue then only the next instruction will be pushed into the queue** otherwise if only a 1-byte space is vacant then there will not be any allocation in the queue. It will wait for a spacing of 2 bytes in subsequent queue decoding operations.
- Instruction pre-fetch queue works in a sequential manner so if there is any branch condition then in that situation pre-fetch queue fails. Hence to avoid chaos instruction queue is flushed out when any branch or conditional jumps occur.

2. The Execution Unit (EU):

The main components of the EU are General purpose registers, the ALU, Special purpose registers, the Instruction Register and Instruction Decoder, and the Flag/Status Register.

1. Fetches instructions from the Queue in BIU, decodes, and executes arithmetic and logic operations using the ALU.
2. Sends control signals for internal data transfer operations within the microprocessor.(Control Unit)
3. Sends request signals to the BIU to access the external module.
4. It operates with respect to T-states (clock cycles) and not machine cycles.

8086 has four 16-bit general purpose registers AX, BX, CX, and DX which store intermediate values during execution. Each of these has two 8-bit parts (higher and lower).

- **AX register: (Combination of A_L and A_H Registers)** It holds operands and results during multiplication and division operations. Also an accumulator during String operations.

- **BX register: (Combination of B_L and B_H Registers)** It holds the memory address (offset address) in indirect addressing modes.

- **CX register: (Combination of C_L and C_H Registers)** It holds the count for instructions like a loop, rotates, shifts and string operations.

- **DX register: (Combination of D_L and D_H Registers)** It is used with AX to hold 32-bit values during multiplication and division.

Arithmetic Logic Unit : Performs arithmetic and logic operations.
Special purpose registers: Special purpose registers are called Offset registers also. Which points to specific memory locations under each segment.

- **Stack Pointer:** Points to Stack top. Stack is in Stack Segment, used during instructions like PUSH, POP, CALL, RET etc.
- **Base Pointer:** BP can hold the offset addresses of any location in the stack segment. It is used to access random locations of the stack.
- **Source Index:** It holds offset address in Data Segment during string operations.
- **Destination Index:** It holds offset address in Extra Segment during string operations.

Instruction Register and Instruction Decoder:
The EU fetches an opcode from the queue into the instruction register. The instruction decoder decodes it and sends the information to the control circuit for execution.

Flag/Status register (16 bits): It has 9 flags that help change or recognize the state of the microprocessor.
6 Status flags:
1. Carry flag(CF)
2. Parity flag(PF)
3. Auxiliary carry flag(AF)
4. Zero flag(Z)
5. Sign flag(S)
6. Overflow flag (O)

Status flags are updated after every arithmetic and logic operation.

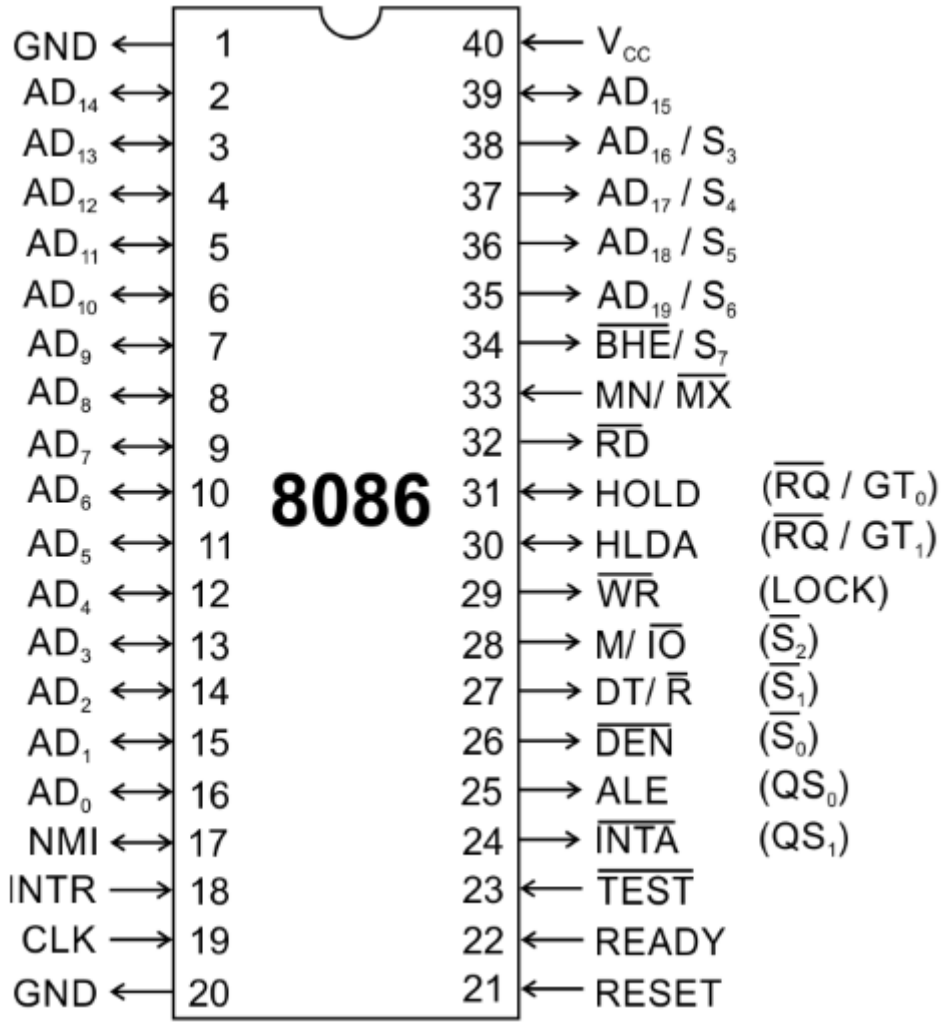
3 Control flags:
1. Trap flag(TF)
2. Interrupt flag(IF)
3. Direction flag(DF)

These flags can be set or reset using control instructions like CLC, STC, CLD, STD, CLI, STI, etc. The Control flags are used to control certain operations.

Execution of whole 8086 Architecture:

1. All instructions are stored in memory hence to fetch any instruction first task is to obtain the Physical address of the instruction is to be fetched. Hence this task is done by Bus Interface Unit (BIU) and by Segment Registers. Suppose the Code segment has a Segment address and the Instruction pointer has some offset address then the physical address calculator circuit calculates the physical address in which our instruction is to be fetched.
2. After the address calculation instruction is fetched from memory and it passes through C-Bus (Databus) as shown in the figure, and according to the size of the instruction, the instruction pre-fetch queue fills up. For example *MOV AX, BX* is 1 Byte instruction so it will take only the 1st block of the queue, and *MOV BX,4050H* is 3 Byte instruction so it will take 3 blocks of the pre-fetch queue.
3. When our instruction is ready for execution, according to the FIFO property of the queue instruction comes into the control system or control circuit which resides in the Execution unit. ***Here instruction decoding takes place***. The decoding control system generates an opcode that tells the microprocessor unit which operation is to be performed. So the control system sends signals all over the microprocessor about what to perform and what to extract from General and Special Purpose Registers.
4. Hence after decoding microprocessor fetches data from GPR and according to instructions like ADD, SUB, MUL, and DIV data residing in GPRs are fetched and put as ALU's input. and after that addition, multiplication, division, or subtraction whichever calculation is to be carried out.
5. According to arithmetic, flag register values change dynamically.
6. ***While Instruction was decoding and executing from step-3 of our algorithm, the Bus interface Unit doesn't remain idle. it continuously fetches an instruction from memory and put it in a pre-fetch queue and gets ready for execution in a FIFO manner whenever the time arrives.***
7. So in this way, unlike the 8085 microprocessor, here the fetch, decode, and execution process happens in parallel and not sequentially. This is called *pipelining,* and because of the instruction pre-fetch queue, all fetching, decoding, and execution process happen side-by-side. Hence there is partitioning in 8086 architecture like Bus Interface Unit and Execution Unit to support Pipelining phenomena.

8086 Pin Diagram



Intel 8086 is microprocessor. It is available in 40 pin DIP chip. It uses a 5V DC supply for its operation. The 8086 uses 20-line address bus. It has a 16-line data bus. The 20 lines of the address bus operate in multiplexed mode. The 16-low order address bus lines have been multiplexed with data and 4 high-order address bus lines have been multiplexed with status signals.

S2	S1	S0	Characteristics
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port

S2	S1	S0	Characteristics
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive state

A16/S3, A17/S4, A18/S5, A19/S6 : The specified address lines are multiplexed with corresponding status signals.

A17/S4	A16/S3	Function
0	0	Extra segment access
0	1	Stack segment access
1	0	Code segment access
1	1	Data segment access

BHE'/S7 : Bus High Enable/Status. During T1 it is low. It is used to enable data onto the most significant half of data bus, D8-D15. 8-bit device connected to upper half of the data bus use BHE (Active Low) signal. It is multiplexed with status signal S7. S7 signal is available during T2, T3 and T4.

RD': This is used for read operation. It is an output signal. It is active when low.

READY : This is the acknowledgement from the memory or slow device that they have completed the data transfer. The signal made available by the devices is synchronized by the 8284A clock generator to provide ready input to the microprocessor. The signal is active high(1).

INTR : Interrupt Request. This is triggered input. This is sampled during the last clock cycles of each instruction for determining the availability of the request. If any interrupt request is found pending, the processor enters the interrupt acknowledge cycle. This can be internally masked after resulting the interrupt enable flag. This signal is active high(1) and has been synchronized internally.

NMI : Non maskable interrupt. This is an edge triggered input which results in a type II interrupt. A subroutine is then vectored through an interrupt vector lookup table which is located in the system memory. NMI is non-maskable internally by software. A transition made from low(0) to high(1) initiates the interrupt at the end of the current instruction. This input has been synchronized internally.
INTA : Interrupt acknowledge. It is active low(0) during T2, T3 and Tw of each interrupt acknowledge cycle.
MN/MX' : Minimum/Maximum. This pin signal indicates what mode the processor will operate in.
RQ'/GT1′, RQ'/GT0′ : Request/Grant. These pins are used by local bus masters used to force the microprocessor to release the local bus at the end of the microprocessor's current bus cycle. Each of the pin is bi-directional. RQ'/GT0′ have higher priority than RQ'/GT1′.
LOCK' : Its an active low pin. It indicates that other system bus masters have not been allowed to gain control of the system bus while LOCK' is active low(0). The LOCK signal will be active until the completion of the next instruction.
TEST' : This examined by a 'WAIT' instruction. If the TEST pin goes low(0), execution will continue, else the processor remains in an idle state. The input is internally synchronized during each of the clock cycle on leading edge of the clock.
CLK : Clock Input. The clock input provides the basic timing for processing operation and bus control activity. Its an asymmetric square wave with a 33% duty cycle.
RESET : This pin requires the microprocessor to terminate its present activity immediately. The signal must be active high(1) for at least four clock cycles.
Vcc : Power Supply(+5V D.C.)
GND : Ground
QS1,QS0 : Queue Status. These signals indicate the status of the internal 8086 instruction queue according to the table shown below

QS1	QS0	Status
0	0	No operation
0	1	First byte of op code from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

DT/R : Data Transmit/Receive. This pin is required in minimum systems, that want to use an 8286 or 8287 data bus transceiver. The direction of data flow is controlled through the transceiver.
DEN : Data enable. This pin is provided as an output enable for the 8286/8287 in a minimum system which uses transceiver. DEN is active low(0) during each memory and input-output access and for INTA cycles.
HOLD/HOLDA : HOLD indicates that another master has been requesting a local bus .This is an active high(1). The microprocessor receiving the HOLD request will issue HLDA (high) as an acknowledgement in the middle of a T4 or T1 clock cycle.
ALE : Address Latch Enable. ALE is provided by the microprocessor to latch the address into the 8282 or 8283 address latch. It is an active high(1) pulse during T1 of any bus cycle. ALE signal is never floated, is always integer.