Basic Architecture of 8086 Microprocessor

Overview

The Intel 8086 microprocessor is a 16-bit processor with two main functional units:

- **Bus Interface Unit (BIU)**
- **Execution Unit (EU)**

It employs **parallel processing**, where BIU handles external bus operations while EU executes instructions.

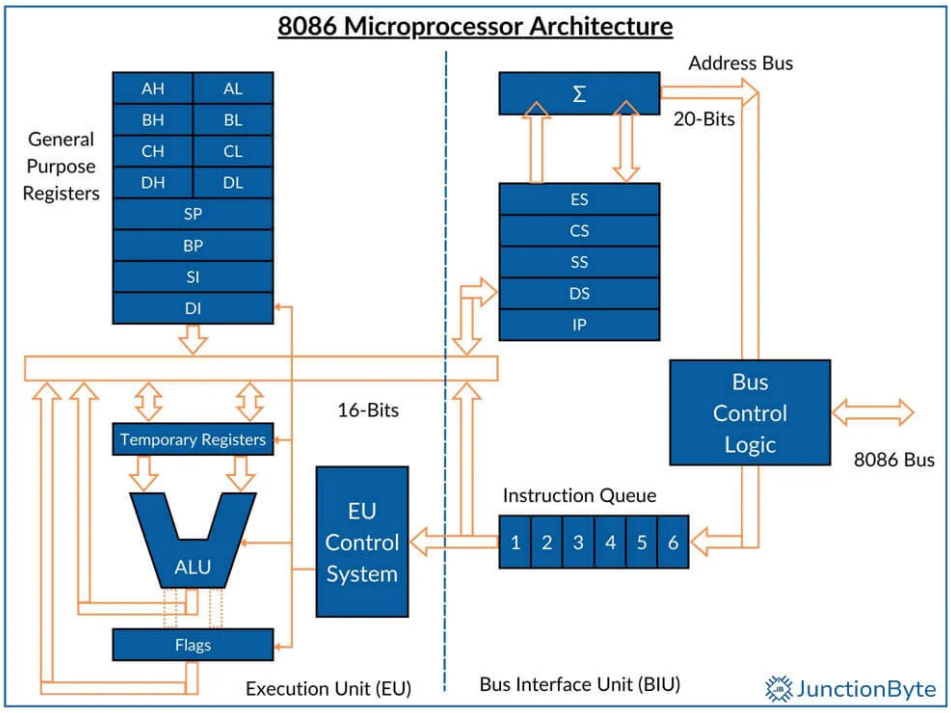

Figure 1: Basic Architecture of Intel 8086 Microprocessor

Bus Interface Unit (BIU)

The BIU is responsible for:

- Address generation using **segment registers** and **instruction pointer (IP)**.
- **Instruction queue** management (fetching instructions in advance).
- Bus control logic (interfacing with memory and I/O).
- Performing **external bus operations**:
  - Instruction fetching.
  - Reading/writing operands from memory.
  - Input/output operations for peripherals.

Execution Unit (EU)

The EU is responsible for:

- **Instruction decoding and execution**.
- Accepting instructions from the **instruction queue**.
- Handling **general-purpose registers** (AX, BX, CX, DX).
- Performing arithmetic and logic using the **ALU**.
- Using the **Control Unit** to manage instruction execution.
- Updating the **Flag Register (status register)** after execution.

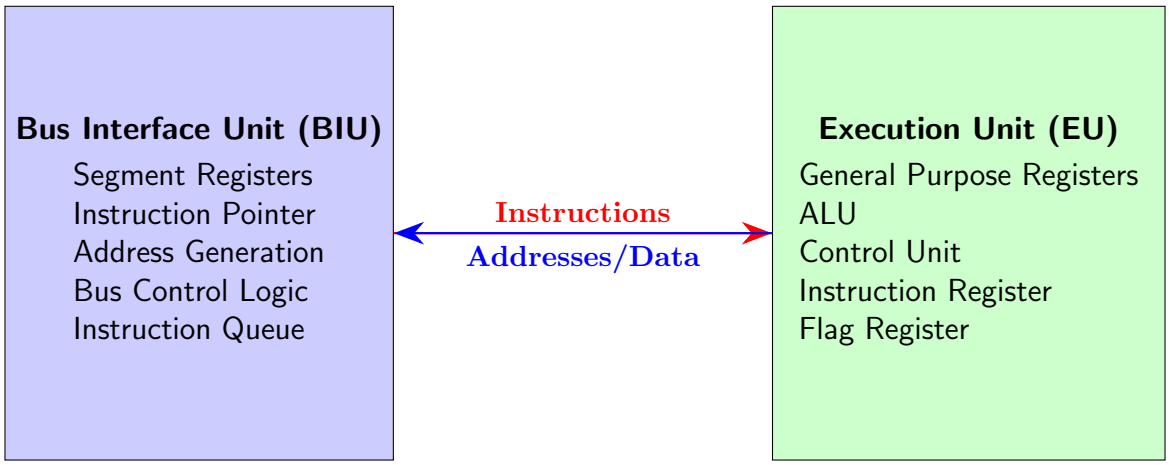

Figure: Basic Architecture of 8086 (BIU–EU model)

Instruction Queue and Pipelining

- BIU fetches instructions from memory and stores them in the **6-byte instruction queue**.
- EU reads instructions from the queue for decoding and execution.
- This allows **pipelining**: while EU executes one instruction, BIU prefetches the next.
- Saves CPU time by overlapping fetch and execution cycles.

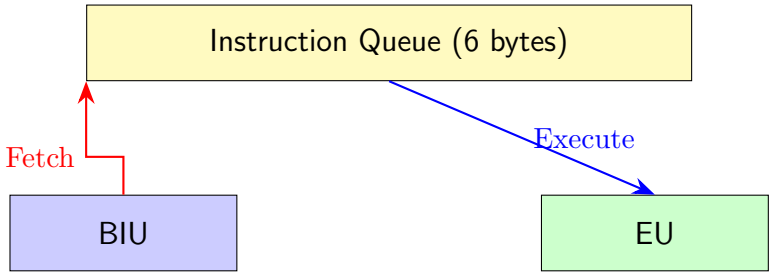

Figure: Instruction Queue Operation in 8086

Queue Operation Details

- Queue is filled by BIU and emptied by EU simultaneously.
- **Odd CS:IP** ⇒ 1-byte instruction fetched.
- **Even CS:IP** ⇒ 2-byte instruction fetched.
- Improves performance by overlapping memory access and execution.

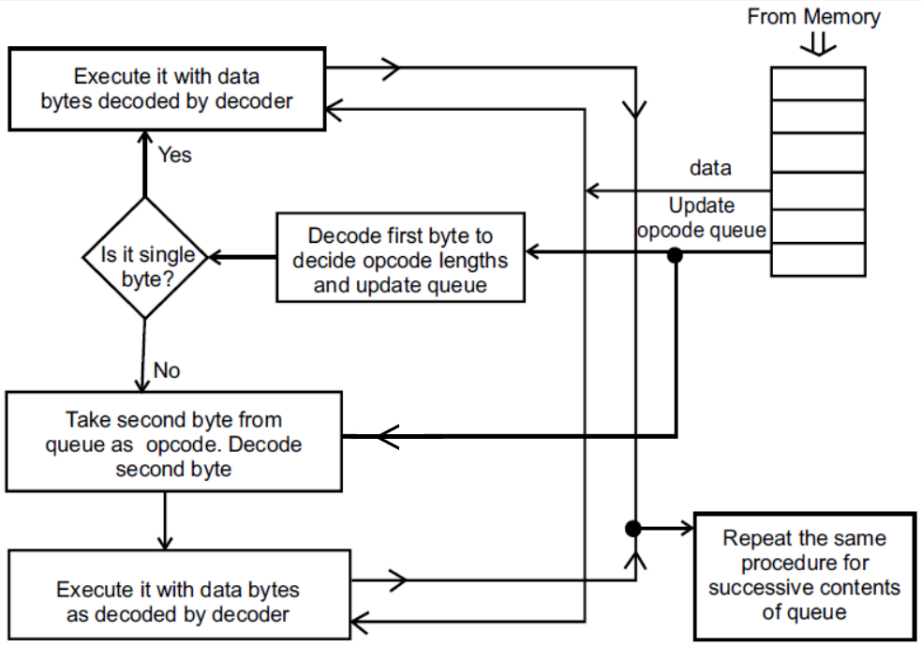

Figure 2: Queue operation

Why 8086 Has a 20-bit Address Bus

Key Idea

Although the Intel 8086 is a **16-bit processor**, it uses a **20-bit address bus** to access up to **1 MB of memory**. This is achieved through **segmented addressing**.

1. Word Size vs. Address Bus

- 8086 has 16-bit registers and ALU, so its native word size is 16 bits.
- A 16-bit address bus would only allow access to $2^{16} = 65{,}536$ bytes $= 64\,\text{KB}$.
- To overcome this limitation, Intel introduced the **Segment:Offset addressing scheme**.

2. Segment:Offset Addressing and Physical Address Computation

- The CPU provides two 16-bit values:
  - **Segment register** – base of a 64 KB memory block.
  - **Offset register** – location within that block.
- These are combined using the formula:

$$\text{Physical Address} = (\text{Segment} \times 16) + \text{Offset}$$

- Multiplying the segment by 16 (i.e., shifting left by 4 bits) ensures that all segment bases start at a **16-byte boundary**.
- Adding the offset then gives a **20-bit physical address**.

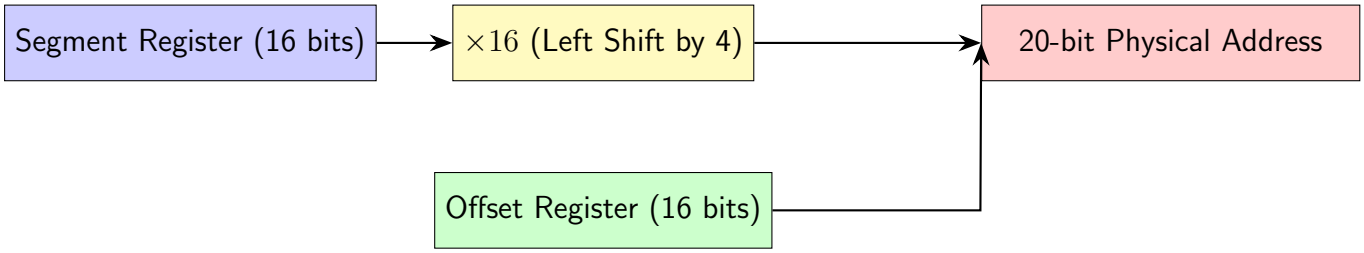

Figure: Segment + Offset = 20-bit Physical Address in 8086

3. Memory Capacity

- Since the physical address is 20 bits wide:

$$2^{20} = 1,048,576 \text{ bytes} = 1 \text{ MB}$$

- Therefore, the 8086 can directly address up to **1 MB of memory**.

Summary

The 8086 employs a **Segment:Offset addressing scheme**, where **(Segment × 16) + Offset** generates a 20-bit physical address. This clever trick overcomes the 64 KB limitation of 16-bit addressing and enables the processor to access **1 MB of memory space**.