# x86 Processor Architecture

Learning Assembly Language

## Topics:

A. How the instruction execution cycle works

B. The Modes of operation and how they are used

C. How registers and flags manage memory

D. How the floating point unit performs mathematical functions

## A. How the instruction execution cycle works

- Every CPU follows a cycle to execute instructions:
    - **Fetch** – Get instruction from memory
    - **Decode** – Figure out what it means
    - **Execute** – Perform the action
    - **Store (optional)** – Save the result

Repeats billions of times per second in modern CPUs.

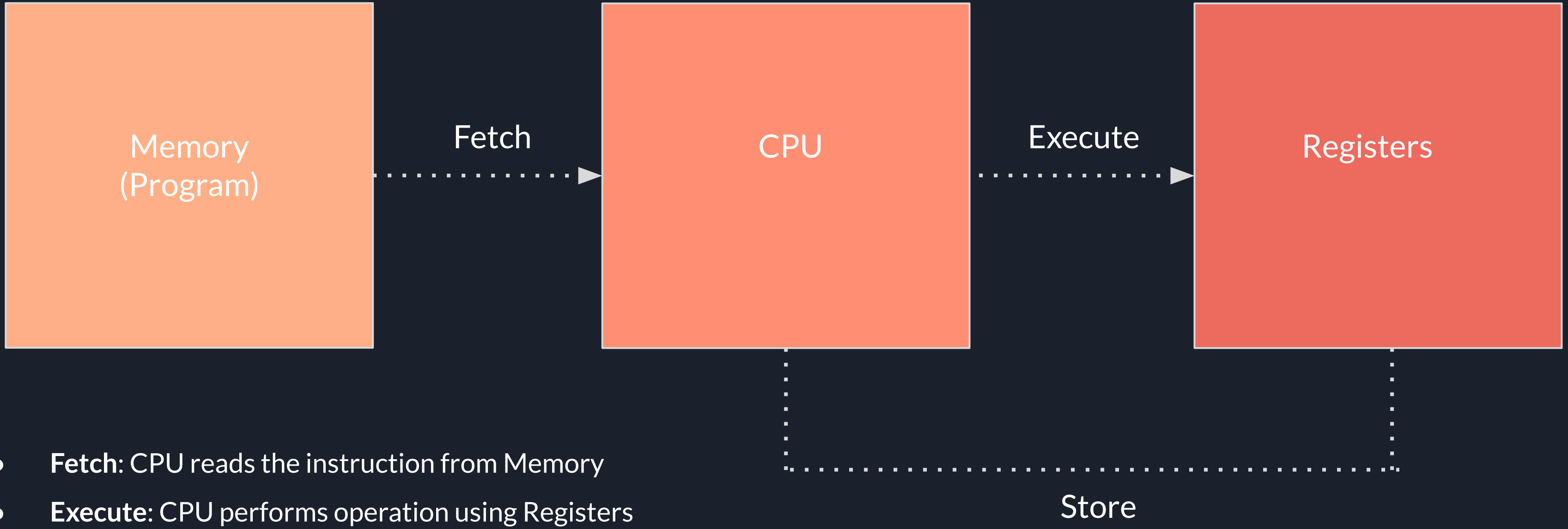
## Example – Simulating an ADD Instruction

- Example instruction: ADD R1, R2

  → Adds the value in register R2 to R1

- Simulated with registers:
  - R1 = 4, R2 = 6
  - After execution: R1 = 10

## Instruction Cycle Flow Diagram



- **Fetch**: CPU reads the instruction from Memory
- **Execute**: CPU performs operation using Registers
- **Store**: Result (if any) is written back to memory or registers

## B. The Modes of Operation and How They Are Used

**CPU Modes of Operation**

- CPUs run in different modes based on privilege level and control:
    - User Mode: Restricted. No direct hardware access. Used by applications.
    - Kernel Mode: Full access. Used by the OS.
    - Protected Mode: Supports multitasking, memory protection (used in modern OS).
    - Real Mode: Oldest mode, no memory protection, used by early PCs.

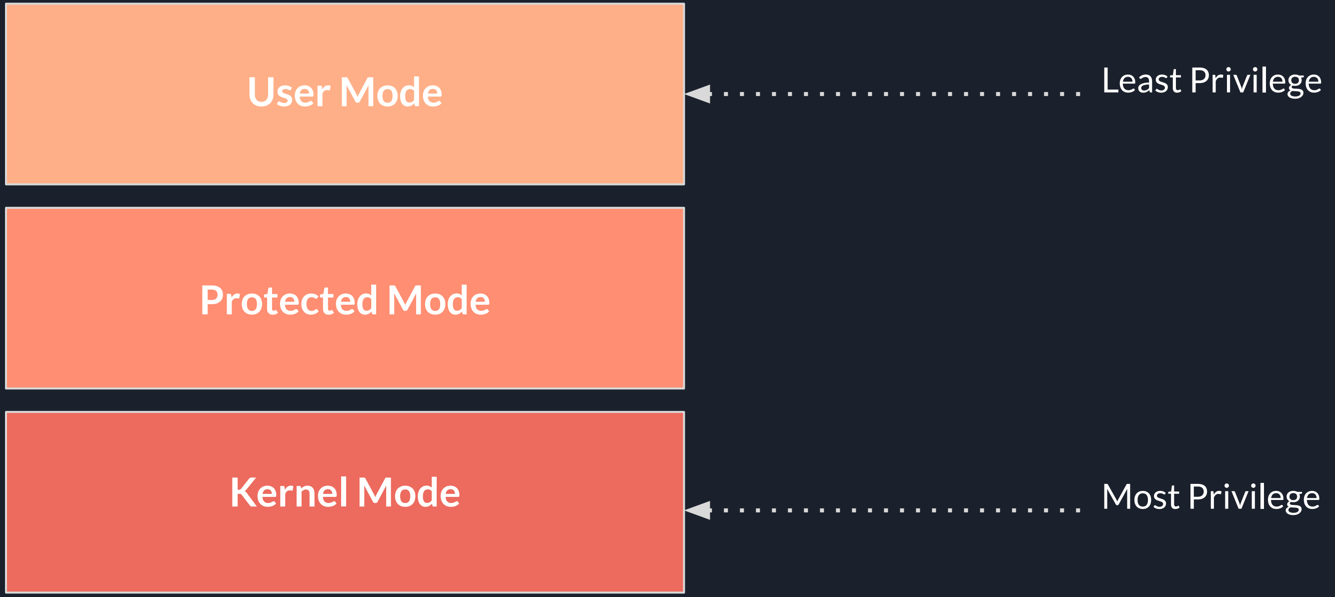
## B. The Modes of Operation and How They Are Used

**Why Modes Matter**

- In User Mode, programs can’t crash your system , they’re sandboxed.
- In Kernel Mode, system calls, drivers, and OS components operate.
- Modes help protect memory and prevent crashing the system.

## B. The Modes of Operation and How They Are Used

**Vertical Privilege Ring Diagram:**



## C. How Registers and Flags Manage Memory

### What Are CPU Registers?

- Registers are small, high-speed storage areas inside the CPU
- Used to:
    - Store temporary data
    - Point to memory addresses
    - Control instruction flow

- Examples:
    - EAX – Arithmetic
    - ESP – Stack pointer
    - EIP – Instruction pointer (During a procedure call (CALL): The current value of EIP is pushed onto the stack to serve as the return address)

## What Are Flags?

- Flags are single-bit values that represent CPU status after operations
- Help the CPU make decisions (e.g., branching, comparisons)
- Common Flags:
  - ZF – Result was zero
  - SF – Result was negative
  - CF – Carry occurred
  - OF – Overflow occurred

## Example – Zero Flag (ZF)

- Operation: 5 + (-5) → Result: 0
- Zero Flag: Set to 1 (True)
- Operation: 2 + 3 → Result: 5
- Zero Flag: Set to 0 (False)

Flags change based on the outcome of an instruction.

They help guide conditional jumps and branching in assembly code.

## D. How the floating point unit performs mathematical functions

What is the Floating Point Unit (FPU)?

- A dedicated part of the CPU that handles decimal (non-integer) operations
- Important for scientific, graphics, financial, and AI computations
- Uses IEEE 754 standard to represent floating-point numbers in binary

## Examples of FPU Operations

- Arithmetic with decimals:
  - 1.25 + 2.5 = 3.75
  - 4.0 ÷ 2.0 = 2.0
- Advanced math:
  - Square root: √9 = 3.0
  - Trig: sin(θ), cos(θ), etc.
- All operations follow binary floating-point format

FPU allows CPUs to perform real-number calculations efficiently and accurately.