## **What is a CPU?**

The **CPU (Central Processing Unit)** is the **"brain"** of the computer. It is responsible for executing instructions from programs and controlling other system components.

## **⚙️ Main Components of a CPU**

### **1. Control Unit (CU)**

* Directs the operation of the processor.
* Fetches instructions from memory.
* Decodes them and coordinates execution.
* Controls data flow between the CPU and other devices.

### **2. Arithmetic Logic Unit (ALU)**

* Performs **arithmetic operations** (add, subtract, multiply, divide).
* Performs **logical operations** (AND, OR, NOT, comparisons).

### **3. Registers**

* Small, fast memory units inside the CPU.
* Temporarily store instructions, data, and addresses.
* Common types:  
  + **Accumulator (ACC)** – stores intermediate results.
  + **Program Counter (PC)** – holds the address of the next instruction.
  + **Instruction Register (IR)** – holds the current instruction.
  + **General Purpose Registers (R0, R1, …)**.

## **🔄 Basic CPU Operations: The Instruction Cycle**

1. **Fetch** – Get the instruction from memory.
2. **Decode** – Understand what the instruction means.
3. **Execute** – Perform the operation.
4. **Store** – Save the result if necessary.

## **⛓️ CPU Performance Factors**

* **Clock Speed** (measured in GHz): Higher = faster.
* **Instruction Set Architecture (ISA)**: Defines how instructions are formatted and executed.
* **Pipelining**: Breaks execution into stages for faster processing.
* **Cache Memory**: Small memory near the CPU for faster access to frequently used data.
* **Number of Cores**: More cores = better parallelism.

## **💡 Fun Fact**

Modern CPUS can have **billions of transistors** and multiple **cores** (quad-core, octa-core) to perform many tasks simultaneously.

## **🔄 Instruction Cycle Steps**

### **1. Fetch**

* The **Program Counter (PC)** holds the memory address of the next instruction.
* This address is sent to **memory**, and the instruction is **fetched**.
* The fetched instruction is stored in the **Instruction Register (IR)**.
* PC is updated to point to the next instruction (usually PC = PC + 1).

### **2. Decode**

* The **Control Unit (CU)** decodes the instruction in the IR.
* It interprets:  
  + **Opcode** (operation to perform)
  + **Operands** (data or addresses involved)

### **3. Execute**

* Based on the decoded instruction, the ALU or other functional units perform the operation.  
  + E.g., if it’s an ADD instruction, the ALU adds two operands.

### **4. Memory Access (if needed)**

* If the instruction involves data stored in memory (e.g., LOAD/STORE), memory is accessed.
* The result is stored in a register or written back to memory.

### **5. Write-back (optional)**

* Final results (e.g., from ALU) are written back to a register or memory location.

## **🔁 Cycle Repeats**

After the instruction is executed, the CPU goes back to the **fetch** step for the next instruction, creating a continuous loop.

## **📘 Example: ADD R1, R2, R3**

(Assume a RISC-style instruction: R1 = R2 + R3)

| **Step** | **Action** |
| --- | --- |
| **Fetch** | CPU fetches ADD R1, R2, R3 from memory using PC |
| **Decode** | CU decodes it: opcode = ADD, operands = R2, R3, R1 |
| **Execute** | ALU performs R2 + R3 |
| **Write-back** | Result is stored in R1 |
| **PC updated** | PC = PC + 1 to fetch the next instruction |