# Basic Structure of Computers

A slightly deeper look inside a computer

# What We'll Cover

- What a computer really is
- Different kinds of computers
- What's inside a computer
- How programs actually run

# 1: What Is a Computer?

- We will look at the core idea of a computer
- Why computers look different
- Trade-offs, not “best” devices

# Question

Before we begin, raise your hand if you have:

- Installed an app
- Copied files to a pen drive
- Restarted a phone to fix a problem
- Used a CLI
- Written a program

# What is a Computer?

A computer is a machine that:

- Accepts **input**
- **Processes** it
- **Stores** it
- Produces **output**

This happens repeatedly.

# Think Deeper

Which of these are computers?

- Calculator
- Smartphone
- ATM
- Traffic signal
- Washing machine

Why?

# Types of Computers (Why So Many?)

Computers differ by:

- Size
- Cost
- Speed
- Purpose

There is **no single best computer**.

# Embedded Computers

Designed to:

- Do **one task**
- Do it **reliably**
- Do it **efficiently**

Often run the same program forever.

# Question

Why not put a laptop inside a washing machine?

- Cost
- Power use
- Reliability
- All of the above

# Personal Computers

Designed for:

- Flexibility
- Many applications
- Human interaction

Trade-off:

- Less efficient for single tasks

# Servers

Optimized for:

- Many users
- Continuous operation
- Network access

Trade-off:

- Often not user-friendly
- Expensive

# 1: Summary

- Computers are defined by behavior and design
- Programmability matters
- Different needs → different designs

# Any questions?

# 2: Inside a Computer

- We will look at the building blocks of a computer
- What they do and how they work together

# Main Functional Units

Every computer still has:

- Input
- Output
- Memory
- Processing
- Control

Even supercomputers.

# Input Unit

Input devices:

- Convert human actions
- Into **digital signals**
- Understood by the computer

# Think About This

When you press a key:

What actually enters the computer?

- The letter
- A number
- A binary code
- You don't need to know the answer at this point

# Output Unit

Output devices:

- Convert digital data
- Into human-accessible form

Examples:

- Screen → light
- Speaker → sound

# Question

- What are the other types of output computers create?
- Can computers create output for other computers?

# Memory: Types?

Where everything gets stored.

Memory differs by:

- Speed
- Cost
- Permanence

Faster = more expensive.

# Primary Memory (RAM)

Used for:

- Running programs
- Temporary data

Key idea:

- CPU can only work on data in RAM

# Question

Why do you need more RAM in your phone?

# Secondary Storage

Used for:

- Long-term storage
- Large data

CPU cannot use it directly.

# Question

Why not make RAM permanent and very large?

- Cost
- Technology limits
- Speed trade-offs

# Any questions?

# Processor (CPU)

CPU does **not**:

- Store files
- Remember much

CPU **only**:

- Executes instructions

# Inside the CPU

CPU contains:

- ALU (calculations)
- Registers (fast storage)
- Control logic

# Registers (Why So Small?)

Registers are:

- Extremely fast
- Very few

Used for:

- Immediate work

# ALU

Handles:

- Arithmetic (+ − × ÷)
- Logic (>, <, =)

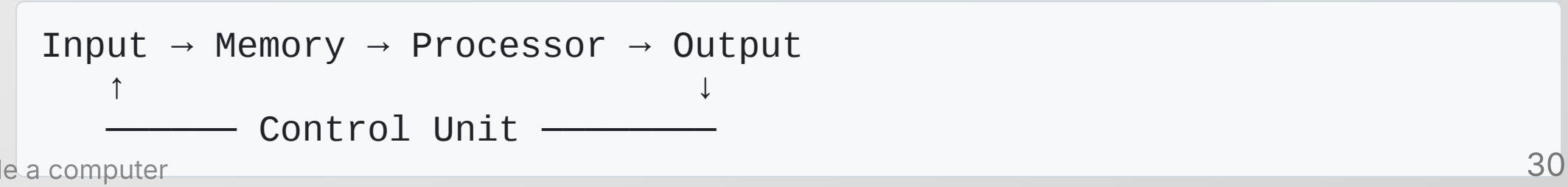
Everything else supports ALU work.

# Control Unit

Control unit:

- Decides **sequence**
- Sends **signals** and directs all activities
- Synchronizes actions (keeps things organized)
- Makes sure everything works together

Like a traffic controller for the computer. Without it, nothing works together.



# Any questions?

# CPU and GPU

Your computer has:

- A **CPU**
- Often a **GPU**

Why two processors?

# What is a GPU?

GPU = **Graphics Processing Unit**

Originally built to:

- Draw images
- Draw video
- Draw games fast

Now, it is used a lot for AI & ML

# Simple Analogy

CPU is like:

- One very smart worker

GPU is like:

- Thousands of simple workers

Both are useful — for different jobs.

# CPU Strengths

CPU is good at:

- Decision making
- Branching (if / else)
- Running programs step by step
- Managing the system

# GPU Strengths

GPU is good at:

- Doing the **same operation**
- On **lots of data**
- At the **same time**

# Question

Which task suits a GPU better?

- Adding 2 numbers
- Adding 1 million pairs of numbers
- Checking file permissions

# Why GPUs Exist

Drawing an image means:

- Millions of pixels
- Same math for each pixel
- Needs to happen very fast

AI/ML inference means:

- To perform thousands of simple calculations simultaneously

Perfect job for a GPU.

# Without GPU

- The computer struggles to draw the screen fast
- Games run with visible lag
- Videos stutter instead of playing smoothly
- AI/ML tasks take a very long time

# GPU Is Not a Replacement for CPU

GPU:

- Cannot run the system
- Cannot control input/output
- Cannot manage programs

CPU is still in charge.

# CPU + GPU Together

CPU:

- Decides *what* to do

GPU:

- Does *a lot of the work*

They cooperate.

# Question

True or False:

GPU is faster than CPU for all tasks.

# Question

If a task needs:

- Many decisions
- Different steps each time

CPU or GPU?

# Any questions?

# Summary

- All computers have same functional units
- CPU works only on RAM
- Fast memory is costly
- CPU controls, GPU accelerates
- GPU is not a CPU replacement
- Task decides processor
- Input is not just keyboard and Output is not just a screen

# Any questions?

# 3: Programs and Execution

- What is a program?
- How it works

# Program = Instructions + Order

A program specifies:

- What to do
- When to do it
- In what order

Order matters.

# Think Like a Computer

If instructions are:

1. Add numbers
2. Take input

What goes wrong?

# Simple Example: 2 + 3 = 5

- Input: You type "2 + 3"
- Memory: Computer stores the numbers
- Processor: Does the math (2 + 3)
- Memory: Stores the answer (5)
- Output: Displays "5" on screen
- The Control unit makes all this happen!

# Data vs Instructions

Both are stored in memory.

Difference:

- Instructions = actions
- Data = values

Computer treats both as bits.

# How Execution Really Works

1. Fetch instruction
2. Decode instruction
3. Execute instruction
4. Repeat

This happens millions of times per second.

# Reflection Question

Which part is the bottleneck when something is not working?

- CPU
- Memory
- Input/Output

# 3: Summary

- Programs drive everything
- Order matters
- Execution is a loop

# TODO

- Check what is your computer hardware configuration
- Come up with the best hardware configuration for your ideal PC and answer why against each

# What’s Next?

- Instruction execution in detail
- Memory addressing
- Binary arithmetic

# Thank you!

Any questions?

Thejesh GN