COMPUTERS AND INFORMATION

Computer Practicum EN - WIS - LES

Lecturer: Niki Hrovatin

What Is a Computer?

A **computer** is a **programmable electronic device** that executes instructions to process data and perform tasks.

A computer performs four fundamental functions:

- **Input** - receives data from users, or other devices.
- **Processing** - performs calculations -- work done by hardware and software to produce results.
- **Output** - presents results or controls other devices.
- **Storage** - saves data and state for later use.



Why do we like computers? A computer can perform different tasks using the same hardware simply by running different programs.

Computers Are Everywhere

Personal computers - desktops and laptops used for everyday work.

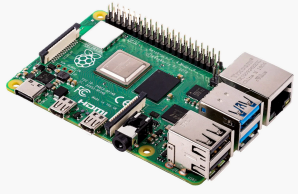


Mobile computers - smartphones and tablets.



Servers - computers that provide data and services to other devices.

Embedded computers - built into machines, vehicles, and measuring instruments.



What is Inside a Computer?

Hardware — the physical components of a computer: processor, memory, storage, and input/output devices.

Software - programs that tell the hardware what to do.

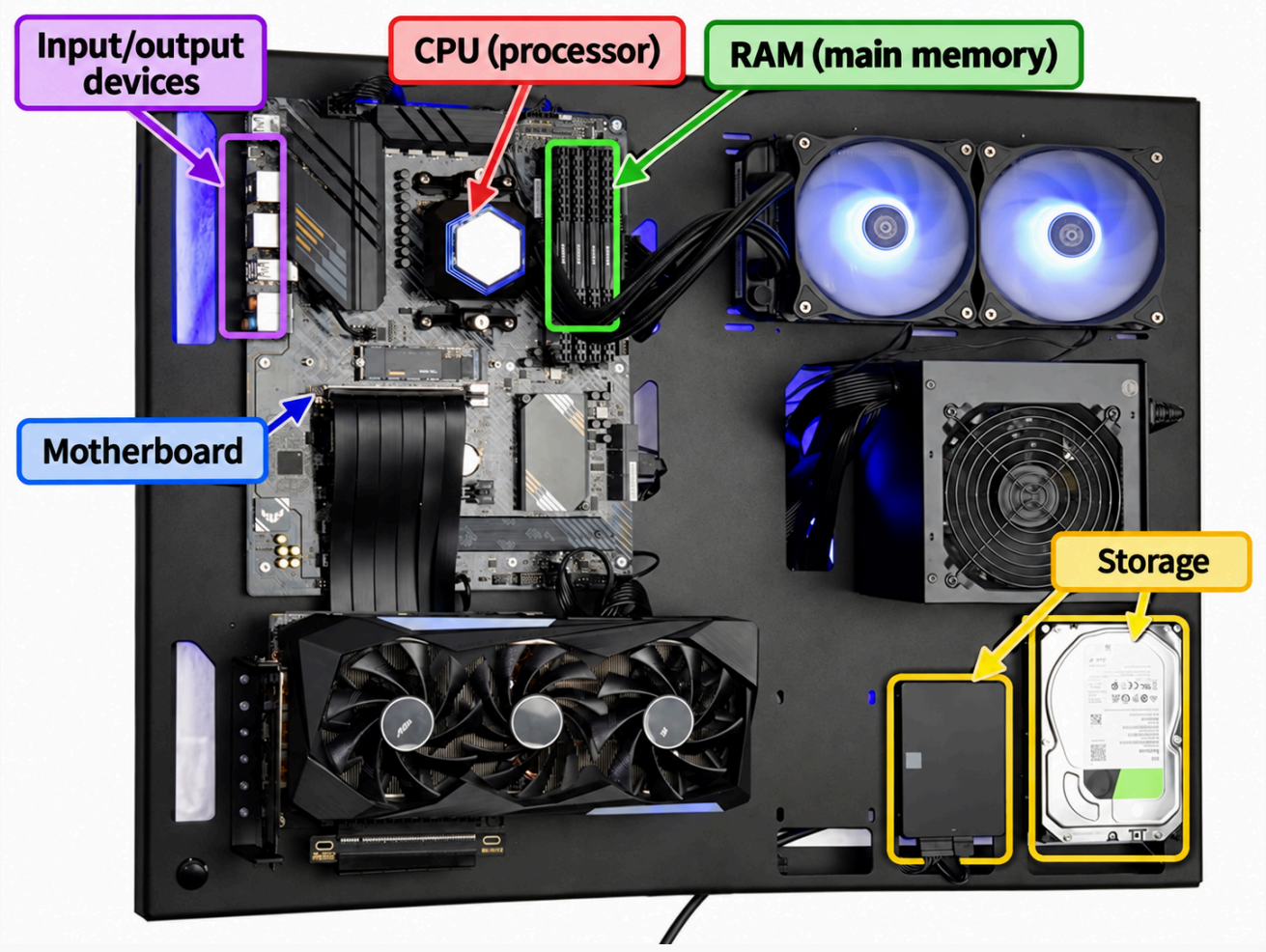
- **Operating system** - manages the computer and its resources.
- **Applications** - help us perform specific tasks (Word, Excel, GTA6, etc.).

Data - the values and instructions that computers store, transfer, and process.

Main Hardware Components

- **CPU (processor)** - executes program instructions.
- **RAM (main memory)** - holds data and programs currently in use.
- **Storage** - retains files and software when the power is off.
- **Input/output devices** - external interaction with other systems and users.
- **Motherboard** - connects the internal components.
- **Supporting components** - provide electrical power, cooling, and basic controls (e.g., power and reset buttons).

Main Hardware Components



Input and Output Devices

Input devices provide data to the computer:

- Keyboard, mouse, microphone, camera, scanner, sensors



Output devices present results or act on the environment:

- Monitor, printer, speakers
- LEDs, motors, relays, and other actuators

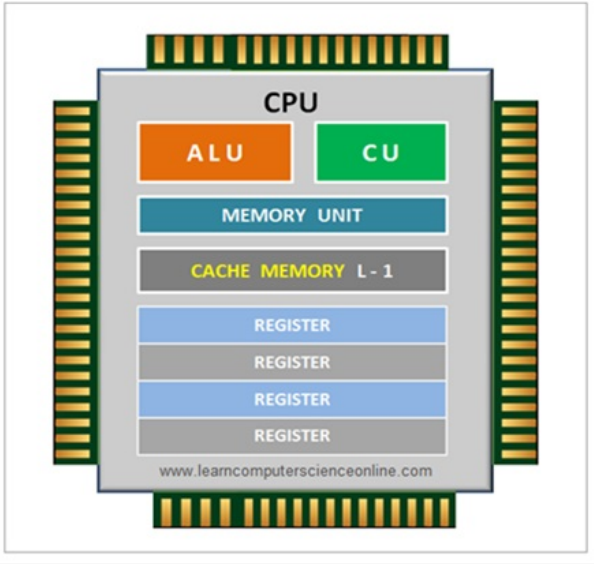


The Central Processing Unit (CPU)

The **CPU** executes the instructions of a running program.

Its main building blocks include:

- **Control unit (CU)** - coordinates instruction execution.
- **Arithmetic logic unit (ALU)** - performs arithmetic and logical operations.
- **Registers** - very small, extremely fast storage inside the CPU.
- **Cache memory** - small, high-speed memory that stores frequently accessed data and instructions, reducing the need to access RAM.



The CPU works closely with **main memory (RAM)**, where active programs and data are held.

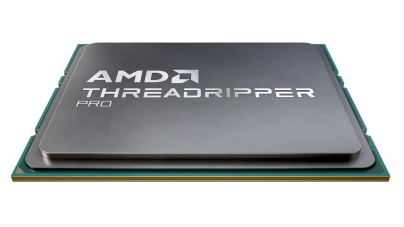
Modern CPUs

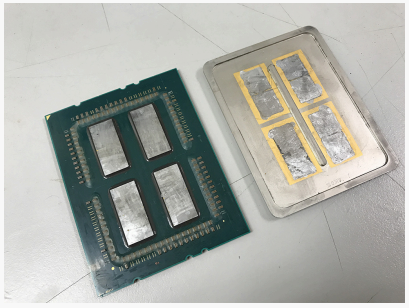
Modern CPUs contain **multiple processing cores**, each capable of executing instructions independently.

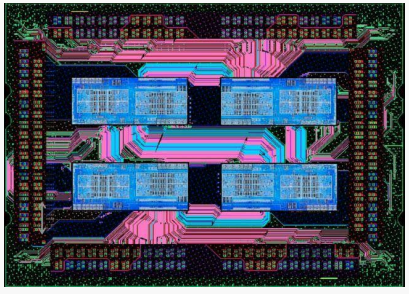
This allows parallel execution of programs

Modern CPU specs:

- **Multiple cores** — CPUs can have 16, 32, 64, or even more cores.
- **Multithreading** — some CPU cores can execute multiple threads (sequences of instructions) concurrently.
- **Clock frequency** — typically ~2–5+ GHz (billions of clock cycles per second).
- **64-bit architecture** — supports processing 64-bit values and addressing large amounts of memory.
- **Manufacturing process** — e.g., 3 nm or 5 nm technology, indicating the semiconductor fabrication generation.
- **Cache hierarchy** — multiple levels of high-speed memory:
  - **L1 cache** ~32–128 KB per core (fastest).
  - **L2 cache** ~0.5–4 MB per core.
  - **L3 cache** ~8–256 MB or more, typically shared among cores.





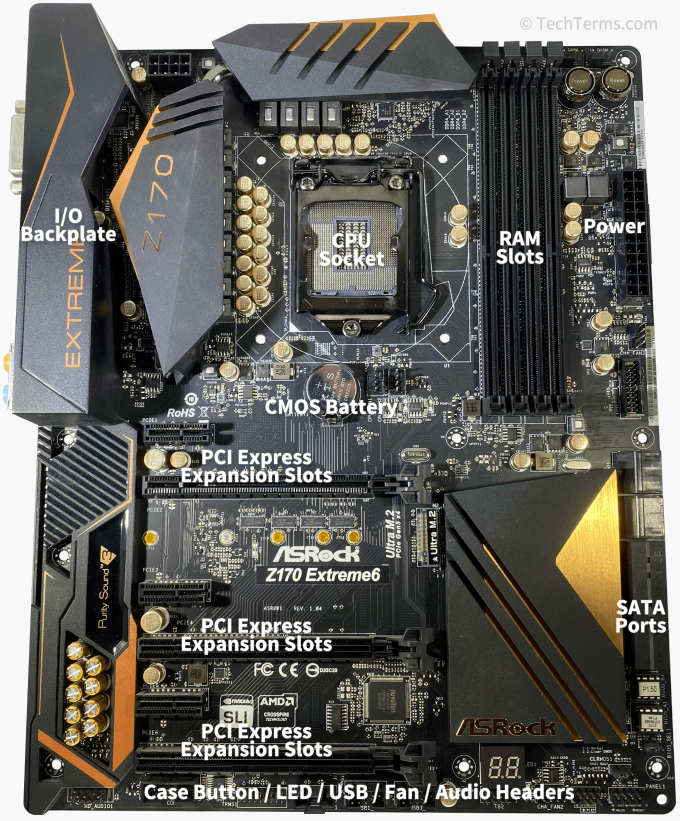


Motherboard and Connections

The **motherboard** is the main circuit board connecting the computer's components.

- Connects the **CPU**, **RAM**, and other hardware.
- Provides connections for storage and external devices.
- Allows components to exchange data through electrical connections and buses.

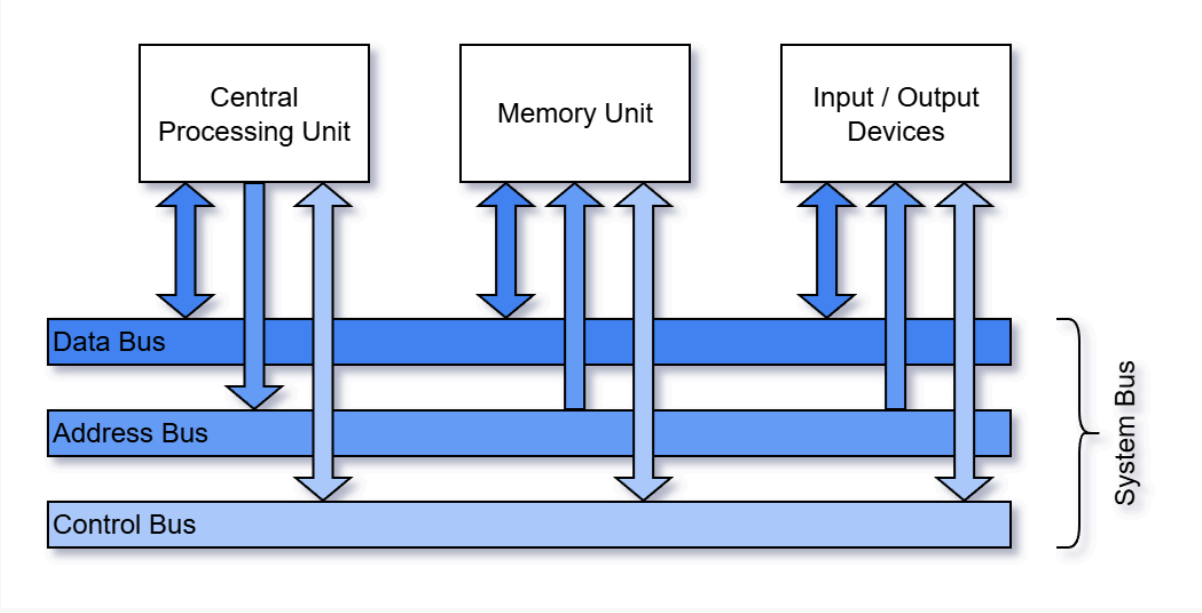
A **power supply** converts electrical power into the voltages needed by the hardware.



Von Neumann Architecture

A basic model explaining how most general-purpose computers work.

- **Memory** stores both program instructions and data.
- The **CPU** fetches instructions from memory and executes them.
- **Input/output devices** exchange information with the outside world.
- **Buses** carry data, addresses, and control signals between components.

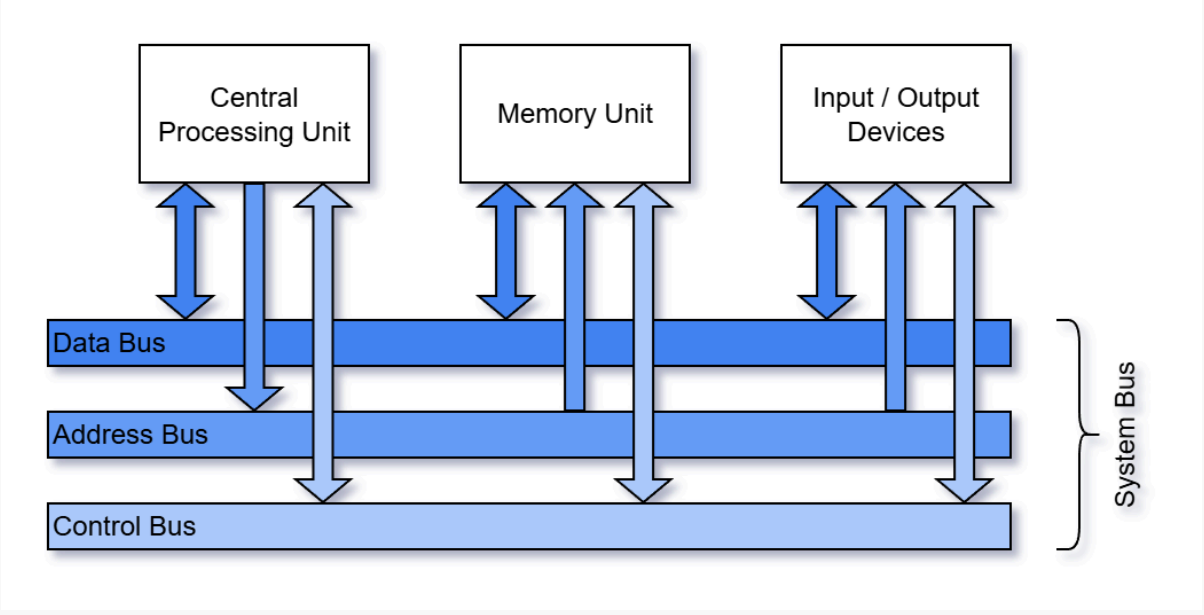


Von Neumann Architecture

Programs are stored as data in memory rather than being physically wired into the machine.

Again: **Why do we like computers?**

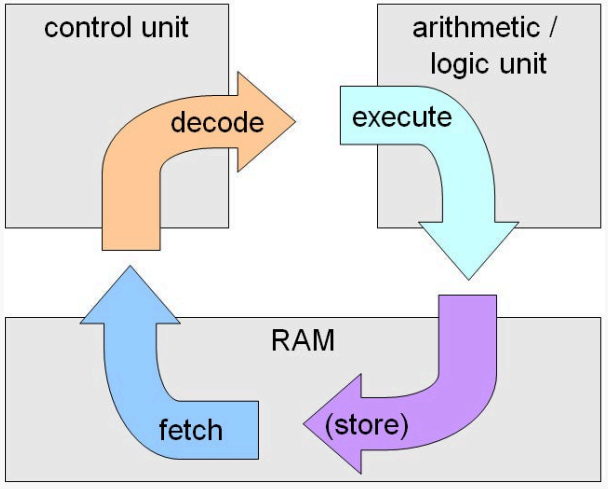
- **Genral purpose machines** A computer can perform different tasks using the same hardware simply by running different programs.



How Does the CPU Execute a Program?

The simplified **fetch-decode-execute cycle**:

1. **Fetch** - read the next instruction from memory.
2. **Decode** - determine what the instruction means.
3. **Execute** - carry out the operation.
4. **Update** - store results and continue with the next instruction.



This cycle repeats extremely rapidly while a program is running.

How fast? A CPU running at **5 GHz** performs **5 billion clock cycles per second**, allowing billions of instructions to be executed every second.

Memory vs. Storage

RAM (Random Access Memory)

- Holds the programs and data currently being used.
- Fast, but **volatile**: its contents are lost when power is removed.

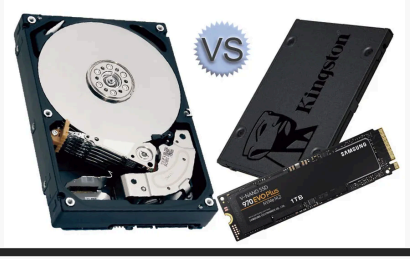


Storage (SSD, HDD, SD card, USB drive)

- Holds files, applications, and datasets for later use.
- **Nonvolatile**: normally retains data without power.



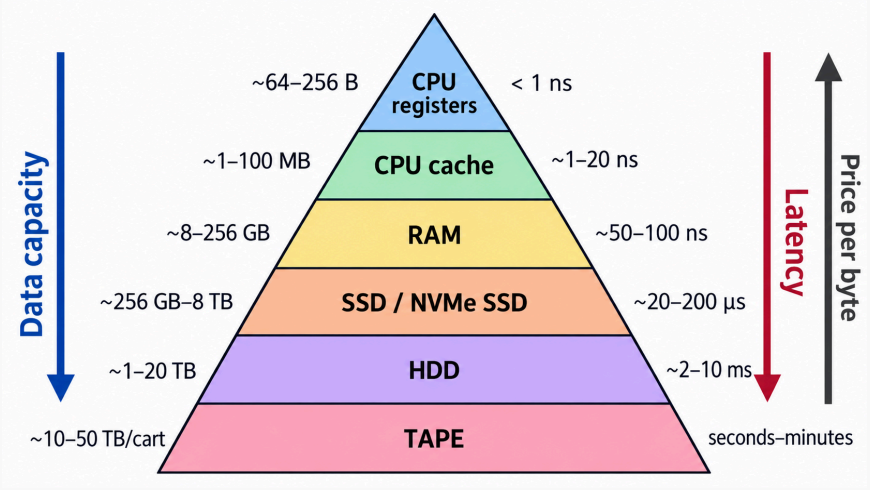
Example: Excel works with data in RAM, but your saved `.xlsx` file resides on storage.



The Memory Hierarchy

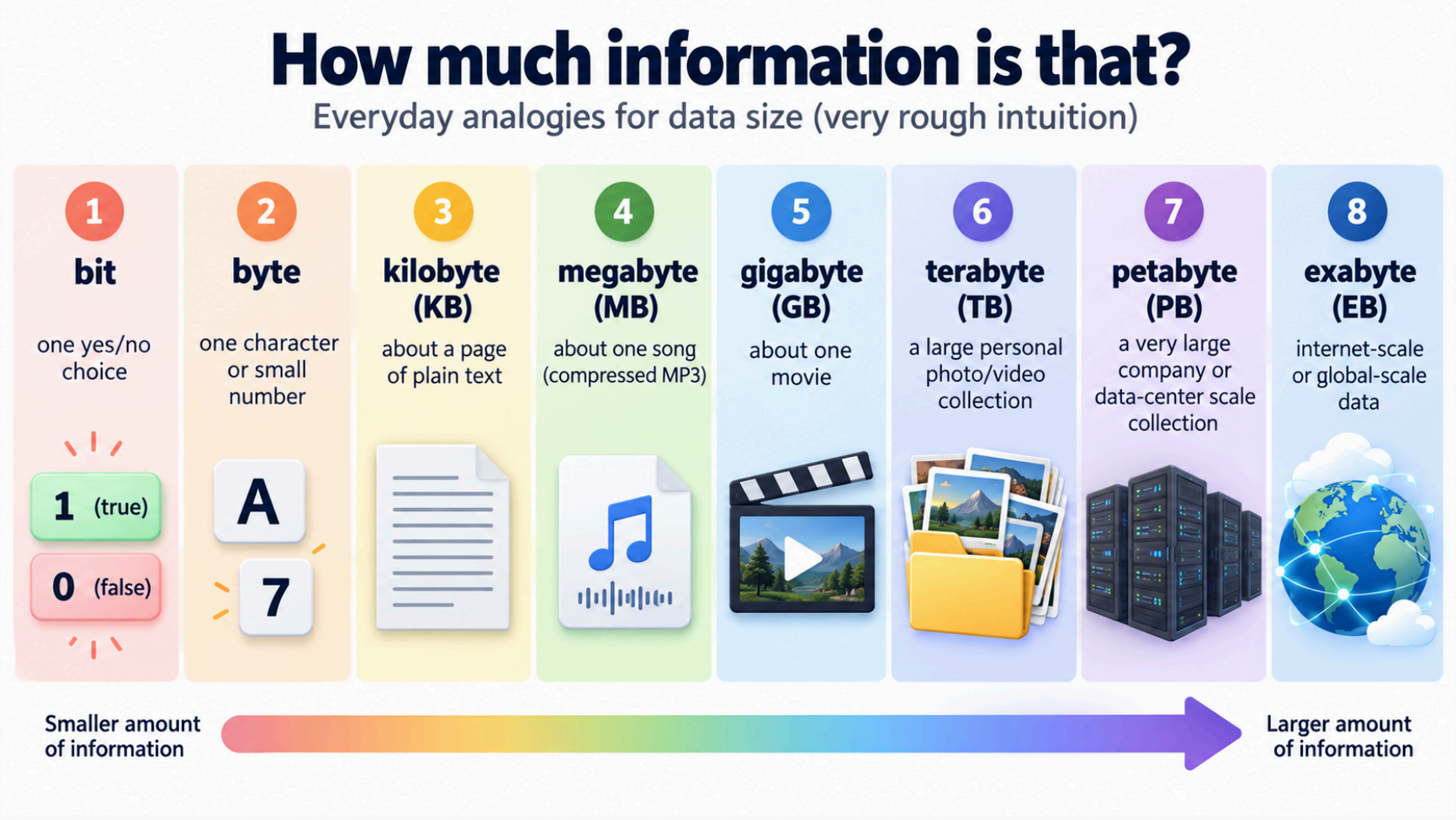
Computers combine different kinds of memory because **speed, capacity, and cost** involve trade-offs.

1. **CPU registers** - extremely fast, very small capacity.
2. **CPU cache** - fast memory for frequently accessed data.
3. **RAM** - larger working memory for active programs.
4. **SSD / HDD** - much larger, persistent storage.



Closer to the CPU generally means faster access, smaller capacity, and higher cost per byte.

Data size


How much information is that?
Everyday analogies for data size (very rough intuition)
1
bit
one yes/no choice
1 (true)
0 (false)
2
byte
one character or small number
A
7
3
kilobyte (KB)
about a page of plain text
4
megabyte (MB)
about one song (compressed MP3)
5
gigabyte (GB)
about one movie
6
terabyte (TB)
a large personal photo/video collection
7
petabyte (PB)
a very large company or data-center scale collection
8
exabyte (EB)
internet-scale or global-scale data
Smaller amount of information
Larger amount of information

What Is an Operating System?

An **operating system (OS)** is software that manages computer hardware and provides services to applications.

- Makes it possible to run programs without controlling each hardware component directly.
- Coordinates the use of the CPU, memory, storage, and devices.
- Provides ways for people to interact with the computer.

Examples: Windows, Linux, macOS, Android, and iOS.

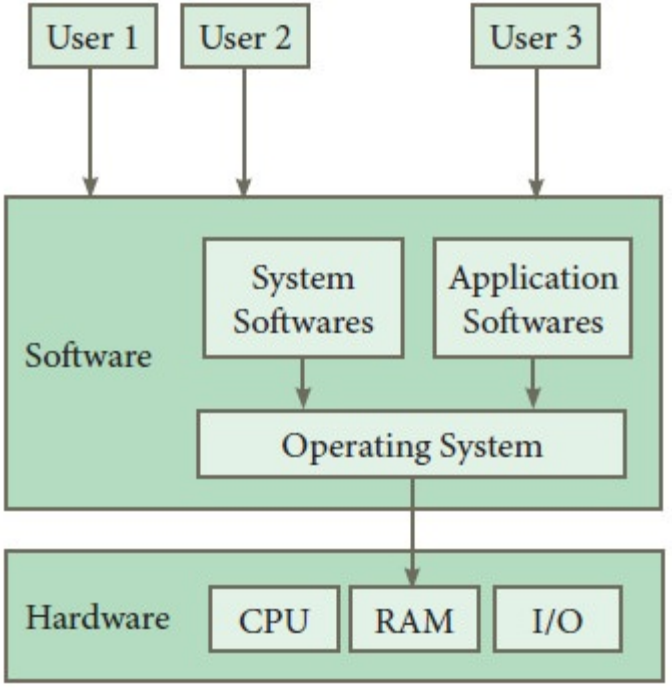


Figure 5.1. Overview of an Operating System

Main Functions of an Operating System

- **Process management** - starts programs and schedules CPU time.
- **Memory management** - allocates RAM to running programs.
- **File management** - organizes files, folders, and access permissions.
- **Device management** - communicates with hardware through drivers.
- **Security** - manages users, permissions, and access to resources.
- **User interface** - provides graphical or command-based interaction.

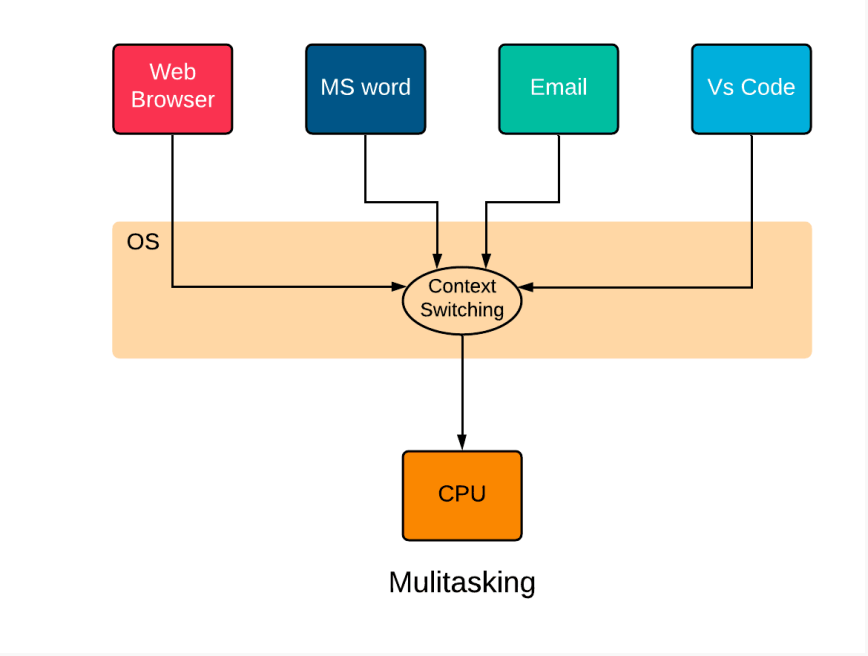
Multitasking and Resource Management

An operating system allows us to work with **several applications at once**.

- It shares processor time among running programs.
- It gives programs space in RAM and keeps their memory separate.
- It handles requests to read files and use devices.
- **Resource allocation** — running programs need sufficient CPU time and memory to remain responsive.

Example: You can edit a report, analyze measurements in Excel, and browse the Web at the same time.

If available resources are insufficient, applications may slow down or become unresponsive.

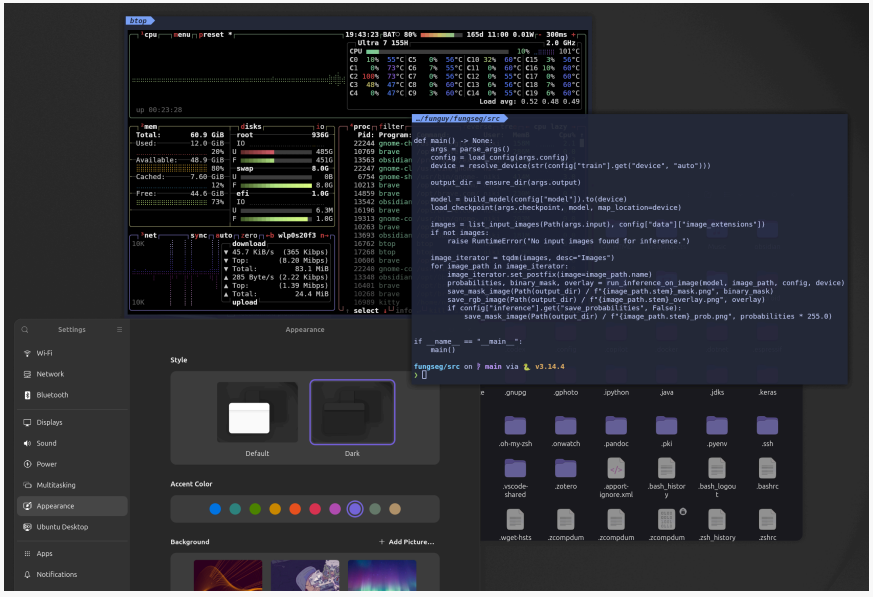


Mulitasking

How Do We Interact with an OS?

Graphical User Interface (GUI)

- Interact through windows, icons, menus, and buttons.
- Example: Opening a CSV file in a file manager.



Command Line Interface (CLI)

- Interact by entering text commands in a terminal.
- Useful for automation, scripting, and managing remote computers.

Both provide access to the same underlying operating system.

Operating Systems Across Devices

Desktop / laptop - Windows, macOS, Linux

Smartphone / tablet - Android, iOS

Server - often Linux or Windows Server

Embedded system (smart devices) - may use an embedded OS or run firmware without a full OS

Computers: From Data to Information

We have seen the **CPU**, **RAM**, **storage**, and the **operating system**. But what do we use them for?

A computer **receives, stores, processes, and presents data**, often to help us obtain **useful information**.

Example: Monitoring temperature in a workshop

- **Input:** a sensor records `20 °C`, `22 °C`, and `24 °C` at 5-minute intervals.
- **Storage:** the measurements are saved in a file.
- **Processing:** a program compares the measurements.
- **Output:** a graph reveals that the temperature is **increasing**.

Two questions: What is the difference between *data* and *information*? How does a computer represent them?

Data and Information: What Is the Difference?

Data are recorded representations of observations, measurements, or facts that can be stored and processed.

- Example: the values `20`, `22`, `24`.

Information is the **meaning we obtain** by interpreting or processing data **in context**.

- **Context:** these are air temperatures in °C, measured 5 minutes apart.
- **Information:** the workshop temperature increased by **4 °C in 10 minutes**.

Without context, the value **24** might mean **24 °C**, **24% wood moisture content**, or **24 mm board thickness**.

Key idea: Data are the *representation*; information is *what that representation tells us*.

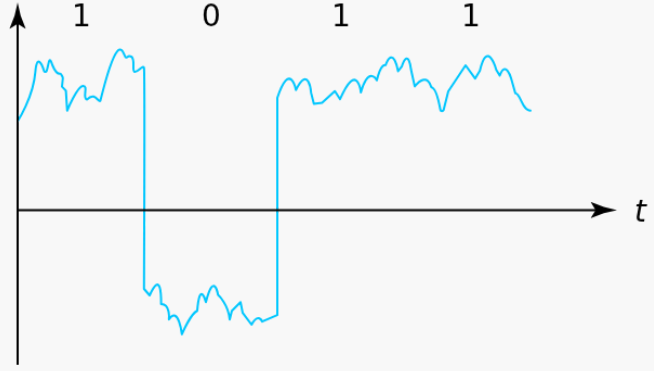
How Do Computers Represent Data?

RAM and storage devices do not hold temperatures, letters, or photographs directly. They hold **encoded representations** of data.

Digital computers use **binary**: combinations of only two symbols, **`0` and `1`**.

A **bit** (short for *binary digit*, symbol **b**) is one binary digit: either **`0`** or **`1`**.

- Electronic circuits can distinguish two logical states reliably.
- Many bits together can represent numbers, text, images, and other data.
- The **CPU executes machine instructions** that are also encoded as bit patterns.



Question: How can we represent a number such as 24 using only zeroes and ones?

How Much Can We Represent with Bits?

A single bit has **2 possible values**. Adding another bit **doubles** the number of possible patterns.

Number of bits	Example patterns	Possible patterns
1	`0`, `1`	2
2	`00`, `01`, `10`, `11`	4
3	`000` through `111`	8
8	`00000000` through `11111111`	256

Rule: With **n bits**, we can represent 2^n **different patterns**.

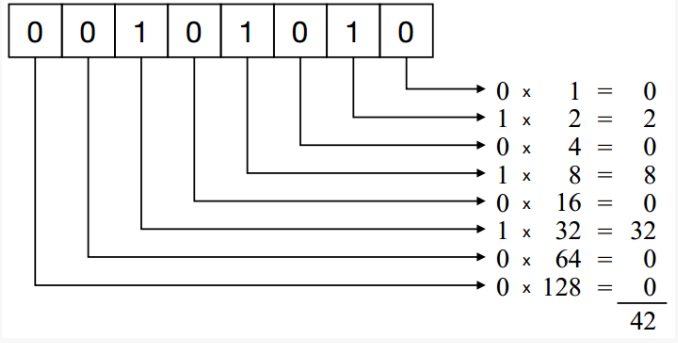
Reading Binary Numbers

We usually write numbers in **decimal (base 10)**, using digits `0` to `9`.

Binary (base 2) uses only `0` and `1`. Its place values, from right to left, are **1, 2, 4, 8, 16, ...**

Example: The decimal number 13

Place value	**8**	**4**	**2**	**1**
Binary digit	`1`	`1`	`0`	`1`



Therefore, **`1101` (binary) = 8 + 4 + 1 = 13 (decimal)**.

Written using 8 bits, this is **`00001101`**.

With 8 bits, an *unsigned integer* can have a value from **0 to 255**.

Binary Patterns Need an Interpretation

The same bits can have **different meanings**, depending on the rules used to interpret them. These rules are called **encodings** or **data formats**.

Binary pattern	Example interpretation
`00000101`	Number **5** (8-bit unsigned integer)
`00110101`	Character **"5"** (ASCII / UTF-8 text)
`01000001`	Letter **"A"** (ASCII / UTF-8), or number **65**
`11111111 00000000 00000000`	**Red pixel** in an 8-bit-per-channel RGB image

The CPU also reads **machine instructions encoded in binary**, according to its instruction set. It does not directly execute English, Java, or Python source code.

Remember: Bits encode data; **encoding rules and context** tell us what the data mean.

Measuring Data: Bits and Bytes

1 bit (b) represents one binary digit. **1 byte (B) = 8 bits**.

We usually measure the **capacity of RAM and storage**, and the **size of files**, in bytes.

Unit	Number of bytes	Number of bits
1 byte (B)	1	8
1 kilobyte (kB)	1,000	8,000
1 megabyte (MB)	1,000,000	8,000,000
1 gigabyte (GB)	1,000,000,000	8,000,000,000
1 terabyte (TB)	1,000,000,000,000	8,000,000,000,000

Careful: B means *bytes* and **b** means *bits* (e.g., **1 MB = 8 Mb**).

You may also see **binary units**: **1 KiB = 1,024 B**, **1 MiB = 1,024 KiB**, **1 GiB = 1,024 MiB**.

Putting It All Together

Imagine a temperature sensor reports `23 °C`:

1. **Input:** the computer receives the measurement.
2. **Representation:** for example, the number `23` can be encoded as `00010111` (an 8-bit unsigned integer).
3. **Storage:** the encoded data can be held in **RAM** or saved to an **SD card**.
4. **Processing:** the **CPU** executes binary-encoded instructions to perform calculations on the data.
5. **Output:** software displays **23 °C**, draws a graph, or raises an alert.

The big picture: Computers **store and process data as bits**. We obtain **information** when those data are interpreted using the right encoding, units, and context.