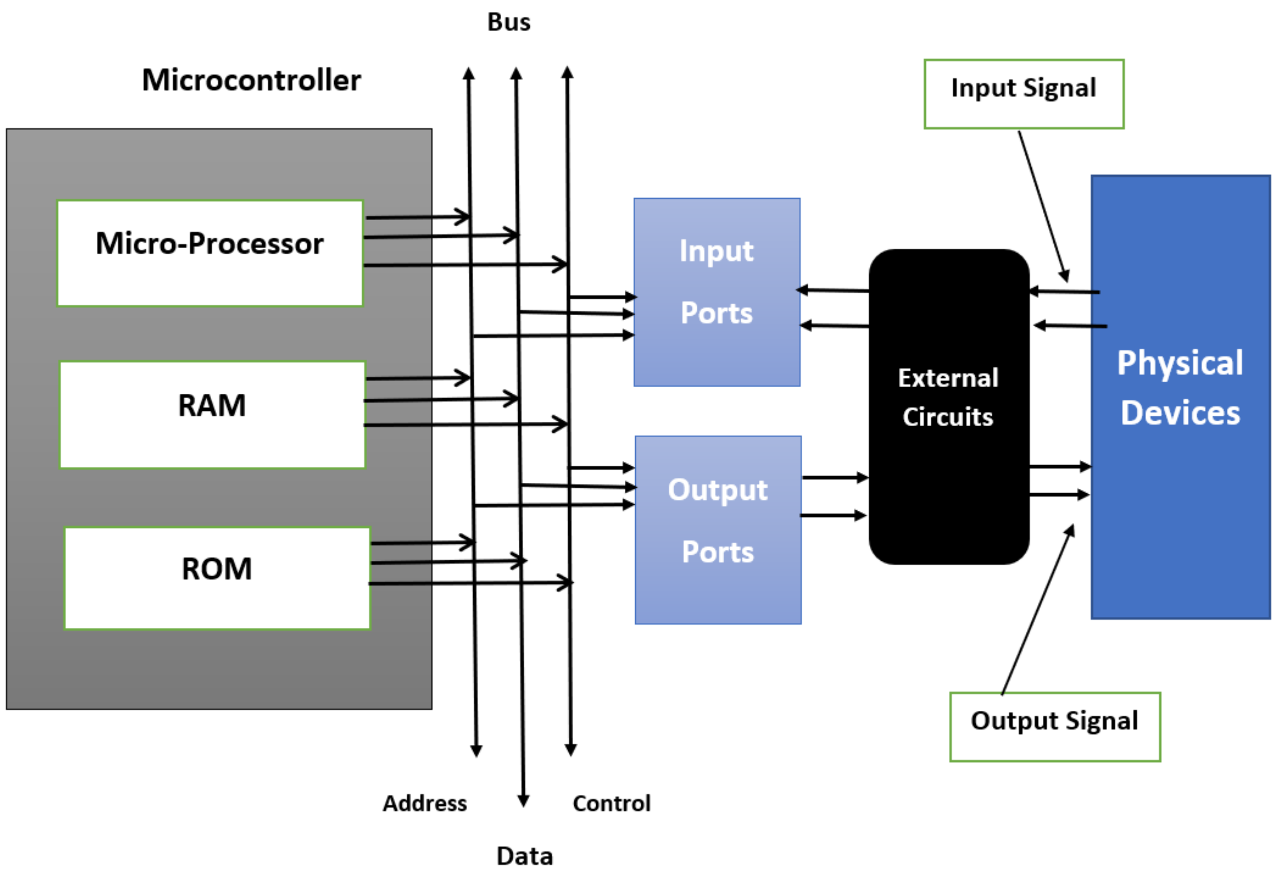
EMBEDDED SYSTEMS

An **embedded system** is a dedicated computing system designed for a specific function within a larger system or device. Unlike general-purpose computers (such as desktops or laptops), embedded systems are **task-specific**, optimized for efficiency, and often operate under real-time constraints.
They integrate **hardware** (microcontrollers, sensors, actuators) and **software** (firmware, real-time operating systems) to perform specialized tasks reliably.



//A **bus** is a communication system that transfers data between different components of a computer or embedded system. It serves as a pathway for data flow between the CPU, memory, input/output (I/O) devices, and other peripherals.

These embedded systems can work without human intervention or with little human intervention.
Dedicated Functionality – Designed for a specific task (e.g., controlling a washing machine)
Resource-Constrained – Limited memory, processing power, and storage compared to general-purpose computers
Real-Time Operation – Often required to respond within strict time limits (e.g., airbags in a car must deploy instantly)
Power Efficiency – Many embedded systems run on batteries and need to consume minimal power (e.g., IoT sensors)
Reliability & Stability – Often designed to operate 24/7 with minimal failures (e.g., industrial automation systems)
Compact & Embedded in Hardware – Usually integrated into the device rather than being separate

Advantages of Embedded System

- Small size.
- Enhanced real-time performance.
- Easily customisable for a specific application.

Disadvantages of Embedded System

- High development cost.
- Time-consuming design process.
- As it is application-specific less market available.

Hardware Layer: Some of the hardware elements that are incorporated in an embedded system include the sensor, actuator, memory, current I/O interfaces as well as power supply. These components are interfaced with the micro controller or micro processor depending up on the input signals accepted.
Input/Output (I/O) Interfaces: They to give the system input in form of data from sensors or inputs made by the users and the microcontroller processes the data received. The processed data is then utilized to coordinate the output devices such as displays, motors or communication modules.
Firmware: Firmware which is integrated within a system's hardware comprises of certain instructions to accomplish a task. Such software is often used for real time processing and is tuned to work in the most optimal manner on the system hardware.
Processing: Depending on the given software and the input data received from the system's inputs the microcontroller calculates the appropriate output or response and manages the system's components.
Real-time Operation: Some of the most common systems are real time, this implies that they have the ability to process events or inputs at given time. This real time capability makes sure that the system accomplishes its intended function within stated time demands.

For instance therein an embedded system in a washing machine, the microcontroller would interface with the buttons (selections made by a user), sensors, for instance water levels, temperature and timers; it would control outputs such as motors, heaters and displays among others based on the program intended for washing cycles.

An **actuator** is a device that converts electrical, hydraulic, or pneumatic energy into mechanical motion. It is commonly used in embedded systems, robotics, industrial automation, and IoT applications to control physical movements such as rotation, linear motion, or force application.

Peripheral devices are external or built-in hardware components that connect to a computer, microcontroller, or embedded system to expand its functionality. These devices can be used for **input, output, storage, or communication like keyboard mouse printers hard drives**

MICROCONTROLLERS

A **microcontroller (MCU)** is a compact integrated circuit designed to control a specific function within an embedded system. It consists of a **central processing unit (CPU), memory (RAM, ROM/Flash), and input/output (I/O) peripherals** all combined on a single chip. Unlike general-purpose computers, microcontrollers are optimized for real-time operations and power efficiency, making them ideal for automation, sensing, and control tasks. They are commonly used in consumer electronics, industrial machines, medical devices, automotive systems, and IoT applications.

Microcontrollers consist of several key components that enable them to function efficiently in real-time control applications.

1. Central Processing Unit (CPU)
   - The CPU is responsible for executing program instructions stored in memory.

  - It follows a cycle of fetching, decoding, and executing instructions.
  - Most microcontrollers use Reduced Instruction Set Computing (RISC) for faster execution.
2. Memory Units
  - Flash Memory: Stores program code (firmware) permanently.
  - Random Access Memory (RAM): Holds temporary data required during execution.
  - Electrically Erasable Programmable Read-Only Memory (EEPROM): Stores small amounts of data that need to be preserved even after power loss.

//built-in non-volatile memory module that serves several important purposes:

- **Persistent Data Storage:**
  EEPROM retains data even when the microcontroller is powered off. This is ideal for storing configuration settings, calibration parameters, or any small data that must be preserved across power cycles.
- **Reprogrammable Memory:**
  Unlike ROM, EEPROM can be erased and rewritten electrically, which allows the MCU to update stored data during operation. However, it has a limited number of write cycles (typically around 100,000 to 1,000,000 cycles), so it's best used for data that doesn't change too frequently.
- **Dedicated or Emulated:**
  Some microcontrollers, such as those in the AVR family (e.g., used in many Arduino boards), have a dedicated EEPROM section. Other MCUs may use a portion of their flash memory to emulate EEPROM functionality, often with additional management routines to handle wear leveling and data integrity.
- **Usage in Applications:**
  EEPROM is commonly used to store user settings, error logs, device calibration data, or other parameters that the system might need to recall after a reset or power loss.

In summary, within an MCU, EEPROM plays a critical role by providing a reliable method for storing essential data persistently, even when the power is turned off.

3. Input/Output (I/O) Ports
  - General-Purpose Input/Output (GPIO) pins allow the microcontroller to interact with external components.
  - Digital I/O: Reads HIGH (1) or LOW (0) signals.
  - Analog Input: Uses an Analog-to-Digital Converter (ADC) to read sensor values.
  - Pulse Width Modulation (PWM): Controls analog-like signals for motors and LEDs.
4. Timers and Counters
  - Used to generate delays, schedule events, or measure external signals.
  - Some timers are dedicated for PWM signal generation to control devices like motors and LEDs.
5. Communication Interfaces
  - UART (Universal Asynchronous Receiver/Transmitter): Used for serial communication. //**Serial communication** is a method of transmitting data **one bit at a time** over a communication channel. It is commonly used for data exchange between computers, microcontrollers, and peripheral devices.

  - SPI (Serial Peripheral Interface): Enables fast communication with peripherals like memory chips and sensors.
  - I2C (Inter-Integrated Circuit): Allows multiple devices to share a common communication line.

  - CAN (Controller Area Network): Used in automotive systems for communication between components.

Microcontroller Execution Process

Microcontrollers execute tasks through a structured process that ensures real-time control.

1. Power-On and Initialization
   - When powered on, the microcontroller executes the bootloader and initializes peripherals.
   - //A bootloader is a small program that runs when a computer or microcontroller is powered on. It is responsible for initializing hardware and loading the operating system or firmware into memory.

2. Instruction Fetch and Decode
   - The CPU fetches an instruction from memory and decodes it to determine the required operation.
3. Execution and Data Processing
   - The microcontroller performs computations, interacts with I/O devices, and modifies memory contents.
4. Sensor Reading and Actuator Control
   - If a sensor is connected, the ADC converts the analog signal to a digital value.
   - Based on the processed data, the microcontroller controls output devices like motors or LEDs.
5. Loop Execution for Continuous Processing
   - The microcontroller runs the program in a continuous loop, ensuring real-time response to inputs.

Microcontroller (MCU)

Advantages:

- **Highly integrated**: Includes CPU, RAM, ROM, and I/O peripherals in one chip.
- **Power-efficient**: Designed for low-power applications (IoT, wearables, battery-powered systems).
- **Cost-effective**: Cheaper than a microprocessor-based system due to fewer external components.
- **Real-time operation**: Suitable for real-time applications requiring precise timing and control.

Limitations:

- **Lower processing power**: Cannot handle complex computations or multitasking efficiently.
- **Limited memory**: Typically has **KBs to MBs** of RAM and storage, insufficient for large applications.
- **No advanced OS support**: Runs firmware or RTOS but not full operating systems like Windows or Linux.

Example of Microcontroller Operation

Consider a microcontroller used in an automatic fan control system:

1. A temperature sensor provides an analog voltage based on the surrounding temperature.
2. The microcontroller's ADC converts this analog voltage into a digital value.
3. The CPU processes the data and compares it to a predefined threshold.

4. If the temperature exceeds the threshold, the microcontroller turns on a cooling fan using a PWM signal.
5. The system continuously monitors the temperature and adjusts the fan speed accordingly.

MICROPROCESSORS

It is a programmable device that takes in input performs some arithmetic and logical operations over it and produces the desired output. In simple words, a Microprocessor is a digital device on a chip that can fetch instructions from memory, decode and execute them, and give results.
A Microprocessor takes a bunch of instructions in machine language and executes them, telling the processor what it has to do. The microprocessor performs three basic things while executing the instruction:

- It performs some basic operations like addition, subtraction, multiplication, division, and some logical operations using its Arithmetic and Logical Unit (ALU). New Microprocessors also perform operations on floating-point numbers.
- Data in microprocessors can move from one location to another.
- It has a Program Counter (PC) register that stores the address of the next instruction based on the value of the PC, Microprocessor jumps from one location to another and makes decisions.

Key Components of a Microprocessor

1. **Arithmetic Logic Unit (ALU)**
   - Performs mathematical and logical operations like addition, subtraction, AND, OR, etc.
2. **Control Unit (CU)**
   - Directs data flow and manages execution of instructions.
3. **Registers**
   - Small memory units inside the CPU that store temporary data and instructions.
4. **Bus Interface**
   - Connects the processor to external memory (RAM, ROM) and I/O devices using address, data, and control buses.
5. **Clock Generator**
   - Controls the speed of operation (measured in MHz or GHz).

2. Microprocessor Execution Cycle (Fetch-Decode-Execute)

The microprocessor operates in cycles to execute instructions:

2.1. Fetch

- The **program counter (PC)** holds the memory address of the next instruction.
- The instruction is fetched from memory and loaded into the **instruction register (IR)**.

2.2. Decode

- The **control unit (CU)** interprets the instruction and determines the necessary operations.

2.3. Execute

- The **ALU** performs the required computation or logic operation.
- Results are stored in a register or written back to memory.
- The **PC** is updated to point to the next instruction.

This cycle repeats continuously as the processor executes programs

Advantages:

- **High computational power**: Suitable for multitasking, AI, gaming, and other intensive applications.
- **Supports full operating systems**: Can run Windows, Linux, macOS, and Android.
- **Scalability**: Can be combined with GPUs, RAM, SSDs, and other high-speed components for advanced performance.

Limitations:

- **Higher power consumption**: Requires cooling systems and consumes more energy.
- **More expensive**: Needs additional components (RAM, storage, power management), increasing cost.
- **Larger physical footprint**: Requires multiple chips and PCBs, making it unsuitable for compact embedded applications.

Microcontrollers are **not used in laptops, desktops, or high-performance systems** because they are designed for **embedded control applications**, not for general-purpose computing. Here's why:

1. Limited Processing Power

- Microcontrollers (MCUs) typically operate at clock speeds in the **MHz range**, while microprocessors (MPUs) in laptops and PCs run at **GHz speeds** (thousands of times faster).
- They have **simpler instruction sets** optimized for control tasks, whereas processors in laptops are optimized for multitasking and high-speed operations.

2. Low Memory and Storage

- MCUs have **limited built-in RAM and ROM** (often in KBs or MBs), while laptops require **GBs of RAM and TBs of storage** to run operating systems and applications.
- A microcontroller cannot efficiently handle the memory-intensive tasks required in a laptop.

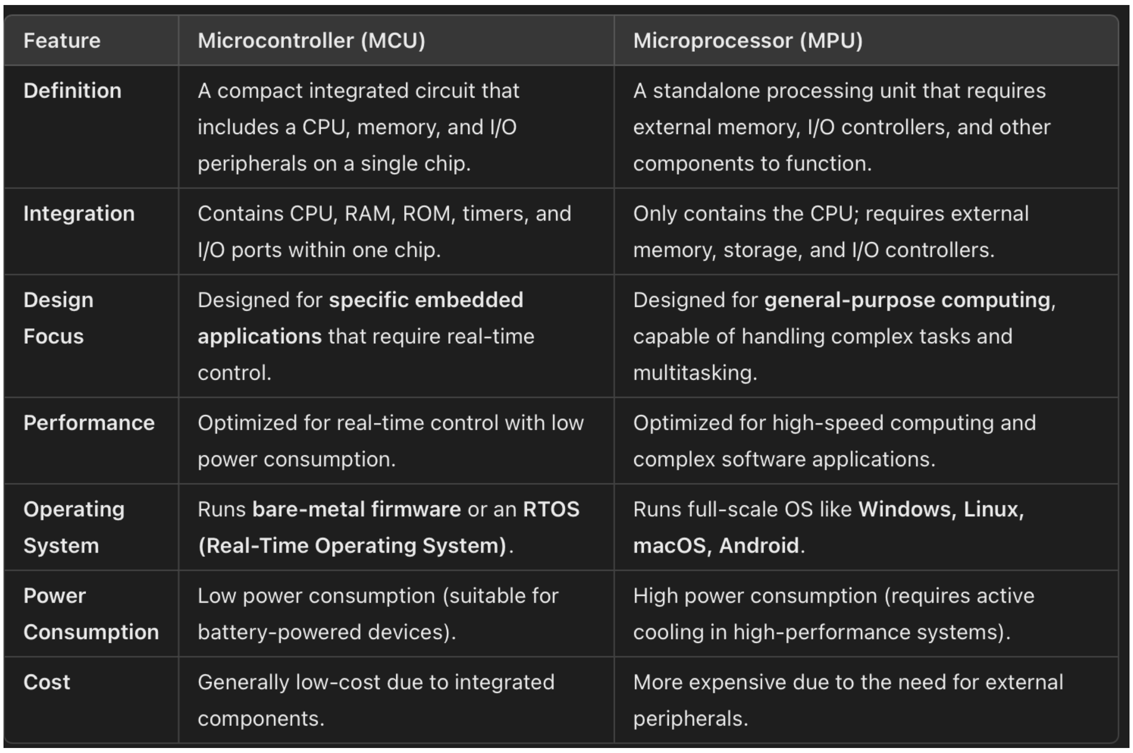
3. No Support for Operating Systems

- Laptops run complex operating systems like **Windows, macOS, and Linux**, which require large memory and powerful CPUs.
- Microcontrollers usually run **bare-metal firmware or real-time operating systems (RTOS)**, which are too simple for multitasking and user applications.

4. Lack of Advanced Peripherals

- Laptops require **dedicated graphics processing (GPUs), high-speed networking (Wi-Fi, Ethernet), and storage controllers**, which MCUs lack.
- MCUs are designed for **real-time control tasks** like reading sensors and controlling motors rather than running complex software.

Architecture refers to the **design and structure** of a computer system, defining how its components interact and function together. It includes the organisation of hardware, data flow, and the set of rules that govern processing and communication.



Feature	Microcontroller (MCU)	Microprocessor (MPU)
Definition	A compact integrated circuit that includes a CPU, memory, and I/O peripherals on a single chip.	A standalone processing unit that requires external memory, I/O controllers, and other components to function.
Integration	Contains CPU, RAM, ROM, timers, and I/O ports within one chip.	Only contains the CPU; requires external memory, storage, and I/O controllers.
Design Focus	Designed for **specific embedded applications** that require real-time control.	Designed for **general-purpose computing**, capable of handling complex tasks and multitasking.
Performance	Optimized for real-time control with low power consumption.	Optimized for high-speed computing and complex software applications.
Operating System	Runs **bare-metal firmware** or an **RTOS (Real-Time Operating System)**.	Runs full-scale OS like **Windows, Linux, macOS, Android**.
Power Consumption	Low power consumption (suitable for battery-powered devices).	High power consumption (requires active cooling in high-performance systems).
Cost	Generally low-cost due to integrated components.	More expensive due to the need for external peripherals.