EMBEDDED SYSTEMS

What is Embedded System?

is a computational system that is developed based on an integration of both hardware and software in order to perform a given task. It can be said as a dedicated computer system has been developed for some particular reason. But it is not our traditional computer system or general-purpose computers, these are the Embedded systems that may work independently or attached to a larger system to work on a few specific functions. These embedded systems can work without human intervention or with little human intervention.

Components of Embedded Systems:

1. Hardware **2.** Software **3.** Firmware

How does an Embedded System Work?

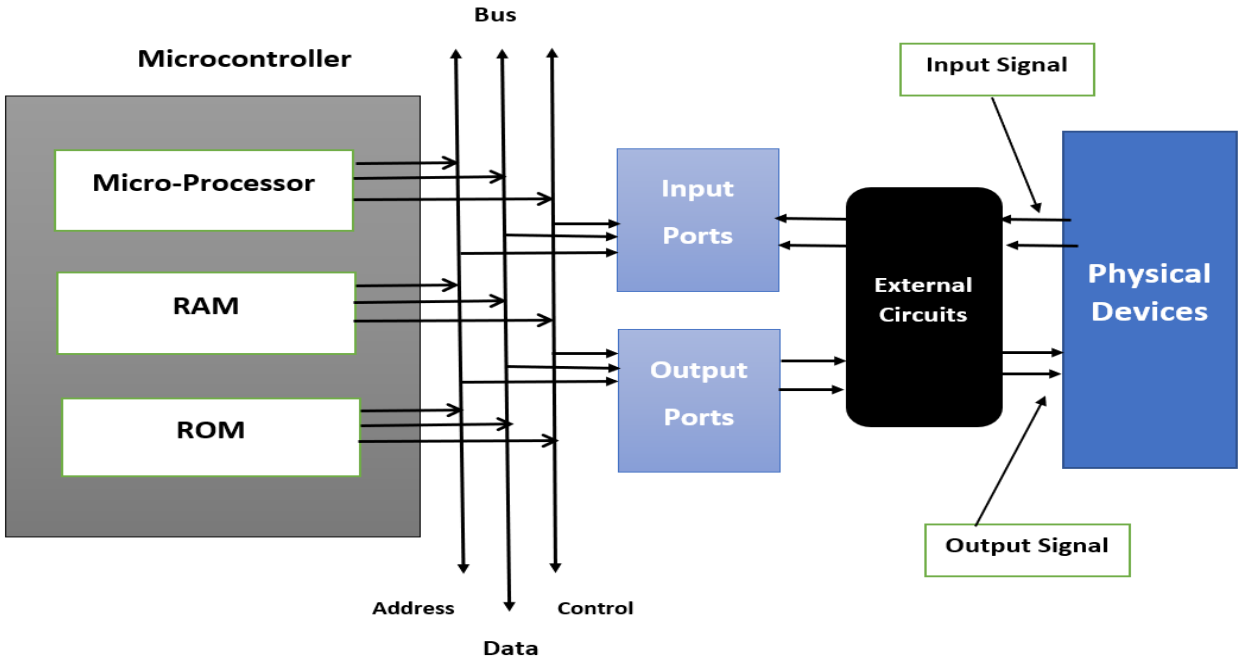
Embedded systems operate from the combination of hardware and software that focuses on certain operations. An embedded system at its heart has microcontroller or microprocessor hardware on which user writes the code in form of software for control of the system. Here is how it generally works:

- **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 :** 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.

Embedded Systems are classified based on the two factors :

1. Performance and Functional Requirements
2. Performance of Micro-controllers



Processor building blocks are the fundamental components that make up a central processing unit (CPU). These components include the Arithmetic logic unit (ALU), control unit ,register. These elements work together to fetch, decode, and execute instructions, performing arithmetic and logical operations, and managing the flow of data within the processor.

An Arithmetic Logic Unit (ALU) is a fundamental digital circuit within a computer's CPU that performs arithmetic and logic operations. It's the core component responsible for calculations and logical comparisons. Essentially, the ALU takes in data and instructions, performs the specified operations, and then outputs the result.

Control Unit (CU):

The control unit acts as the "conductor" of the processor. It fetches instructions from memory, decodes them, and generates control signals that coordinate the activities of other components, ensuring the correct sequence of operations.

Registers in computer systems are small, high-speed storage locations within the CPU used to temporarily hold data and instructions during processing. They are categorized based on their function and purpose. Key register types include Accumulator, Program Counter, Instruction Register, Memory Address Register, Memory Buffer Register, and General Purpose Registers.

Register Types:

- **An accumulator register** is a special-purpose register within a CPU or microprocessor that stores intermediate results of arithmetic and logical operations. It acts as a central hub for calculations, allowing for efficient data manipulation without constantly accessing main memory.
- **A Program Counter (PC) register** is a crucial component within a computer's CPU that stores the memory address of the next instruction to be executed. It essentially acts as a pointer, guiding the CPU through the sequence of instructions that make up a program. Also known as an instruction pointer or instruction address register, it ensures the CPU processes instructions in the correct order.
- **The instruction register (IR)** is a crucial component of a CPU's control unit, primarily responsible for holding the current instruction being executed. It acts as a temporary storage space for the instruction fetched from memory, allowing the CPU to decode and execute it.
  Function:
- **Storage:** The IR holds the instruction that is currently being executed by the CPU.
- **Decoding:** It stores the instruction's opcode and operands, which are then decoded to determine the operation to be performed and the data involved.
- **Control:** The decoded instruction from the IR is passed to the control unit, which directs other components of the CPU to carry out the operation.
- Use:
- **1. Fetch Cycle:**
- The CPU fetches the next instruction from memory and places it in the instruction register.

- **2. Decode Cycle:**
- The control unit reads the instruction from the IR, interprets the opcode and operands, and prepares the necessary control signals.
- **3. Execute Cycle:**
- The control unit activates the appropriate components of the CPU (e.g., ALU, registers) to execute the instruction.

A Memory Address Register (MAR) is a register within a computer's CPU that stores the memory address of the data that needs to be accessed. It acts as a pointer, indicating where the CPU should read data from or write data to in the computer's memory. Essentially, it's the address book for the CPU's memory operations.

Function:

The MAR holds the memory address that the CPU wants to use for either reading data from or writing data to the computer's memory.

A Memory Buffer Register (MBR) sometimes called a Memory Data Register (MDR), is a CPU register that temporarily holds data being transferred to or from main memory. It acts as a buffer, facilitating efficient data transfer between the CPU and memory during read and write operations.

Here's a more detailed explanation:

Function:

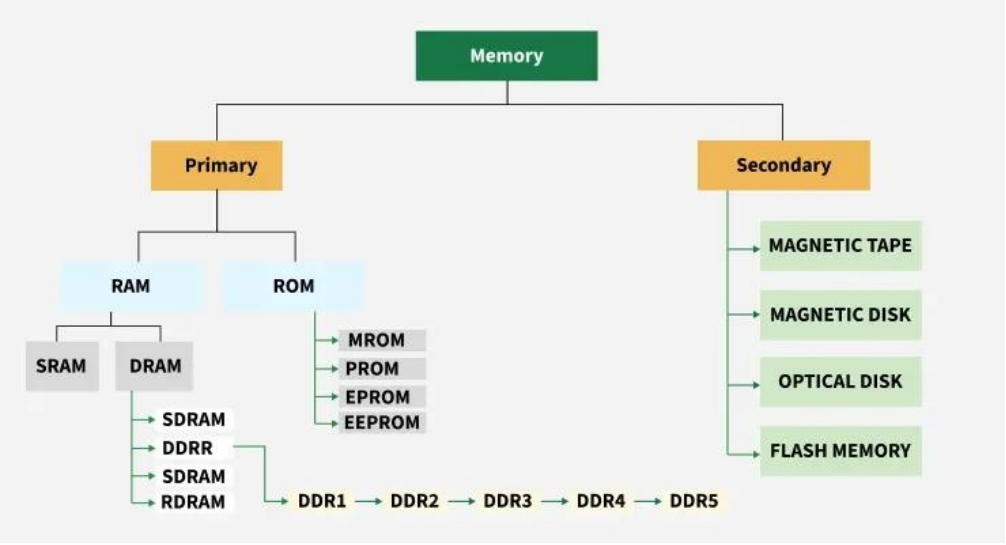
The MBR stores the data that is either being read from or written to memory.

An Instruction Set Architecture (ISA) is a blueprint that defines how software interacts with a computer's hardware, specifically the CPU. It outlines the instructions a processor can understand and execute, essentially acting as the interface between the software (programs, applications) and the hardware (CPU). ISAs define what operations the processor can perform (like arithmetic, logical operations, data movement) and how those operations are accessed.

How Memory Works?

When a computer runs programs or processes data, it uses different types of memory to work smoothly and quickly. These memories help the CPU access data and instructions fast. Let’s see how each type of memory works.

- **Primary memory**, or RAM, is where the computer stores data and instructions that are actively being used by the CPU. It's fast and allows the CPU to access information quickly, but it is **volatile**, meaning it loses all its data when the computer is turned off. When you open a program or file, it is loaded from secondary memory into **RAM** so the CPU can access it more efficiently during use.
- **Secondary memory** is where all your files, programs, and data are stored permanently. This includes storage devices like **SSDs** or **hard drives**. It is **non-volatile**, meaning the data is saved even when the computer is turned off. Secondary memory provides a large amount of storage space, but it is slower than **RAM**. When you open a program, it is transferred from secondary memory to **RAM** for faster access by the CPU.
- **Cache memory** is a small, very fast type of memory located close to the CPU. It stores frequently accessed data and instructions to speed up the computer's performance. The CPU first checks the **cache** for any data it needs. If it's there, the CPU can use it instantly, saving time compared to fetching data from **RAM** or **secondary memory**. If the data isn't in the cache, it's fetched from **RAM** or secondary memory, and then stored in the cache for quicker access next time.



System architecture refers to the overall structure and organization of a system, including its components, their relationships, and the principles governing their design and evolution. It provides a high-level view of the system, defining its major parts and how they interact. When considering timing, system architecture must account for how components respond within specific time constraints and how those constraints impact overall system behavior.