Embedded Systems Concepts

What's an embedded system?

An embedded system is a specialized computer system that is part of a larger system or device, performing specific dedicated functions. Unlike general-purpose computers, embedded systems are designed to execute pre-defined tasks or functions with specific requirements, often in real-time, and are typically optimized for performance, power consumption, size, and cost.

Embedded systems can be found in a wide range of devices and applications, including consumer electronics (such as smartphones, digital cameras, and smart appliances), industrial machines, medical devices, automotive systems (such as engine control units and navigation systems), aerospace systems, and more.

These systems typically consist of a microcontroller or microprocessor, memory (RAM and ROM), input/output interfaces, and often include specialized hardware components like sensors, actuators, or communication modules. They may run simple real-time operating systems (RTOS) or operate without an operating system, depending on the complexity and requirements of the application.

Difference between general purpose and special purpose systems.

***General-purpose systems** possess the versatility to execute a multitude of functions, a capability determined by both the available hardware configuration and the repertoire of installed programs.*

***Special-purpose systems** are tailored to perform specific tasks with a predetermined focus, typically designed to excel in a particular application or domain, thereby offering optimized performance and efficiency for their intended function.*

Difference between system on board and system on chip.

System on Board and **System on Chip** are both approaches to integrating multiple electronic components into a single system, but they differ in their level of integration and functionality.

System on Board (SoB):

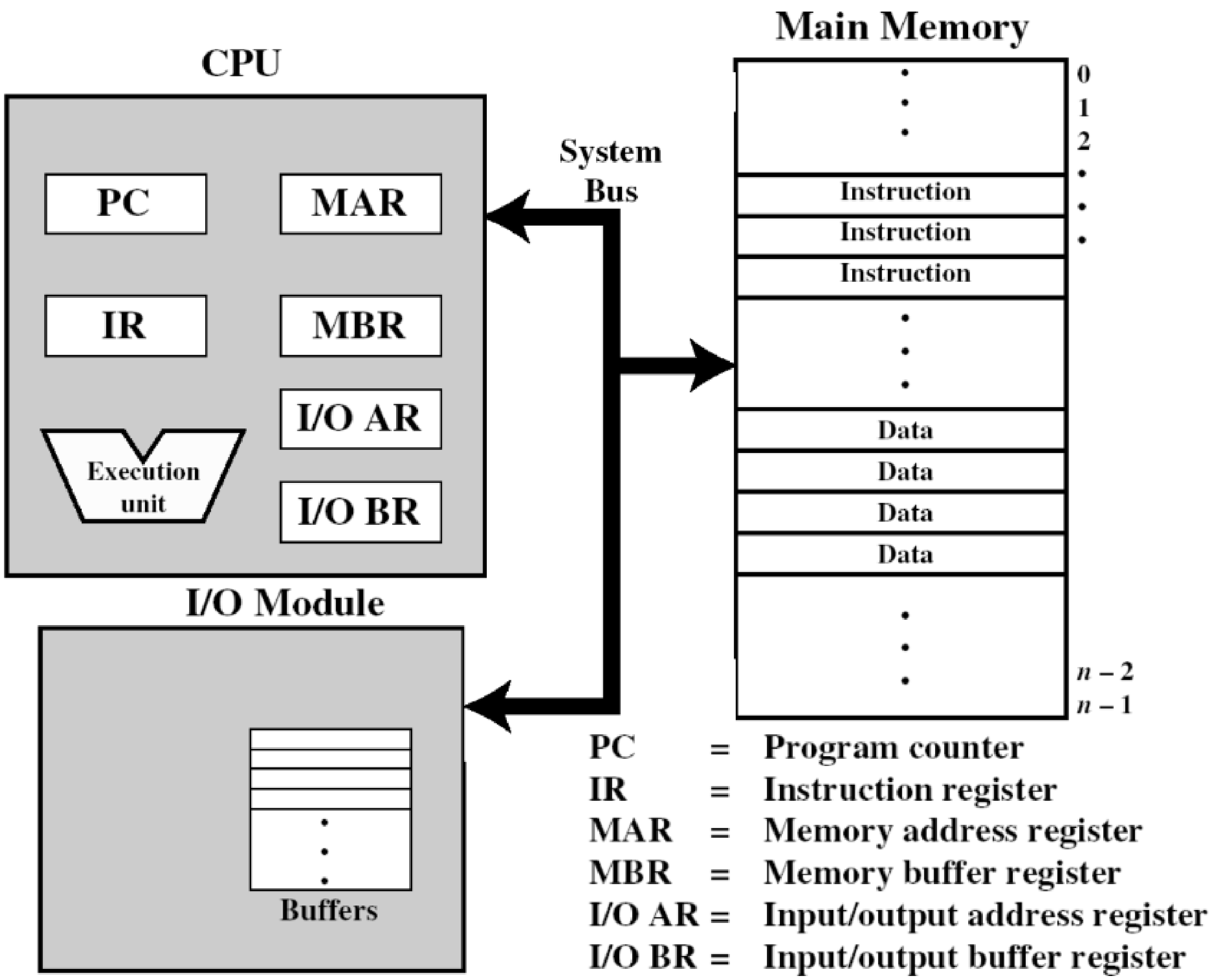
- *In SoB, various electronic components, such as processors, memory modules, input/output interfaces, and other peripherals, are mounted onto a printed circuit board (PCB).*
- *Each component on the board serves a specific function, and communication between components typically occurs via interfaces like buses or communication protocols.*
- *SoB designs offer flexibility in component selection and replacement, as individual components can be upgraded or swapped out if needed.*
- *They are commonly used in applications where modularity and flexibility are essential, such as in desktop computers, servers, and some industrial systems.*

System on Chip (SoC):

- *SoC integrates all or most of the components of a computer or electronic system into a single integrated circuit (IC) or chip.*
- *This integration typically includes a microprocessor or microcontroller, memory, input/output interfaces, and other specialized hardware components like analog-to-digital converters, digital signal processors, or graphics processing units.*
- *SoCs offer significant space and power savings compared to SoB designs because of their high level of integration.*

- *They are commonly used in applications where size, power efficiency, and performance are critical factors, such as smartphones, tablets, IoT devices, and embedded systems.*

Processor structure



Processor Internal register

PC Register

Program counter register, which is used to hold the address of the next instruction to be executed.

IR Register

Instruction Register, which is used to hold the instruction being executed.

MAR Register

Memory Address Register, which is used to hold the address of the data to be written from main memory or the location for the data to be written at. (Connected to the address bus).

MBR Register

Memory Buffer Register, which is used to hold the data to be written on the data bus or receive the data from data bus.

Arithmetic and Logical unit (ALU):

The Arithmetic Logic Unit (ALU) is a fundamental component of a computer's central processing unit (CPU). It is responsible for performing arithmetic and logical operations on input data, producing output results based on the instructions provided by the CPU.

*The **ALU** operates based on control signals provided by the CPU, which dictate the specific operation to be performed and the input data sources. These control signals coming from the CPU are generated by the **Control Unit.***

Control Unit:

*The Control Unit (CU) is another essential component of a CPU. While **ALU** handles arithmetic and logical operations, the **Control Unit** manages the execution of instructions and coordinates the activities of the entire CPU.*

Instruction Fetch: The Control Unit retrieves program instructions from memory. It uses the Program Counter (PC) to determine the address of the next instruction to be fetched.

Here's a breakdown of its functions:

- ***Instruction Decoding****: Once an instruction is fetched, the Control Unit decodes it to determine the operation to be performed and the operands involved.*
- ***Execution Control****: The Control Unit coordinates the execution of instructions by generating control signals to activate different parts of the CPU, such as the ALU, registers, and memory.*
- ***Operand Fetch****: For instructions that require data from memory or registers, the Control Unit ensures that the operands are fetched from the appropriate location.*
- ***Execution****: The Control Unit manages the execution of arithmetic, logical, and other operations specified by the instructions. It ensures that the ALU operates on the correct data with the appropriate operation.*
- ***Memory Access****: If an instruction involves accessing memory (e.g., loading or storing data), the Control Unit manages the data transfer between the CPU and memory.*
- ***Branching and Control Flow****: The Control Unit handles branching instructions, such as conditional and unconditional jumps, which alter the sequence of instruction execution based on certain conditions or program logic.*
- ***Exception Handling:*** *In case of errors or exceptional conditions (e.g., division by zero, invalid memory access), the Control Unit initiates appropriate actions, such as raising exceptions, interrupting the current program flow, and transferring control to an exception handler.*

Memory:

In computer architecture, various types of memory play crucial roles in storing and accessing data and instructions. Here are the different types of memories commonly found in computer systems:

Main Memory:

- ***Random Access Memory (RAM):*** *Volatile memory used for temporary storage of data and instructions that the CPU needs to access quickly. RAM is fast but loses its contents when the power is turned off.*
- *Read-Only Memory (ROM): Non-volatile memory that stores firmware or software that is essential for booting up the computer and initializing hardware components. ROM retains its contents even when the power is turned off.*

Secondary Memory (Storage):

- ***Hard Disk Drive (HDD):*** *Magnetic storage device used for long-term storage of data and programs. HDDs offer large storage capacities but are slower than solid-state drives.*
- ***Solid-State Drive (SSD):*** *Storage device that uses flash memory for data storage. SSDs are faster, more durable, and consume less power than HDDs, making them popular for both personal and enterprise use.*
- ***Flash Memory****: Non-volatile memory commonly used in USB flash drives, memory cards, and solid-state drives. Flash memory allows for fast read and write operations and is widely used for portable storage and in embedded systems.*
- ***Optical Discs:*** *Storage media like CDs, DVDs, and Blu-ray discs, which use optical technology to store data. Optical discs are typically used for distributing software, multimedia content, and backups.*

Cache Memory:

- ***Level 1 (L1) Cache:*** *Small, high-speed memory located directly on the CPU chip. L1 cache stores frequently accessed data and instructions to speed up processing.*
- ***Level 2 (L2) Cache:*** *Larger cache memory located between the CPU and main memory. L2 cache helps reduce the time it takes to access data from main memory.*
- ***Level 3 (L3) Cache****: Additional cache memory found in some CPUs, serving as a larger buffer between the CPU and main memory.*

Difference between Von- Neumann and Harvard architecture

Von Neumann Architecture:

- ***Unified Memory:*** *In the Von Neumann architecture, instructions and data are stored in the same memory space. This means that the CPU accesses both instructions and data from the same memory pool.*
- ***Single Bus Structure:*** *Von Neumann architecture typically employs a single bus for both data and instructions, which means that data transfers and instruction fetches share the same pathway.*
- ***Stored Program Concept****: One of the defining features of the Von Neumann architecture is the stored-program concept, where instructions are treated as data and stored in memory. This allows programs to be easily modified and manipulated by the CPU.*
- ***Sequential Execution:*** *Instructions are executed sequentially, meaning that each instruction is fetched, decoded, executed, and then the next instruction is fetched.*

Harvard Architecture:

- ***Separate Memory Spaces:*** *In the Harvard architecture, instructions and data are stored in separate memory spaces. There are distinct memory units for program instructions (instruction memory) and data (data memory).*
- ***Dual Bus Structure:*** *Harvard architecture typically employs separate buses for data and instructions. This separation allows for simultaneous access to instructions and data, which can improve performance.*
- ***Instruction and Data Pipelining:*** *Harvard architecture often allows for instruction and data pipelining, where multiple instructions and data can be fetched simultaneously, further enhancing performance.*
- ***Parallel Execution****: Because of the separate memory spaces and dual bus structure, Harvard architecture can potentially execute instructions and access data in parallel, leading to improved performance compared to Von Neumann architecture.*