Volume 9, Issue 3, March – 2024 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAR917

Embedded Systems Basics

Konduru Sravani
Department of E.C.E, Bapatla Women's Engineering College, Andhra Pradesh, India

Abstract:- Embedded systems play a pivotal role in modern technological advancements, integrating computing capabilities into various devices and systems. This paper presents an abstract of an embedded system designed for real-time monitoring and control applications in industrial automation. The system utilizes a combination of microcontroller-based hardware and software components to achieve reliable and efficient operation.

Keywords:- *Embedded System, Harvard and Von Neumann, RISC and CISC.*

I. INTRODUCTION

Embedded systems represent the technological backbone of our modern world, quietly powering an extensive array of devices and systems that we interact with daily. From the smartphones in our pockets to the complex machinery in factories, embedded systems play a pivotal role in enabling functionality, automation, and connectivity. In this introduction to embedded systems, we will explore their fundamental characteristics, applications, design considerations, and the evolving landscape of embedded technology. By delving into these aspects, we aim to gain a comprehensive understanding of the role embedded systems play in shaping our interconnected world and driving innovation across myriad domains.

II. WHAT IS EMBEDDED SYSTEM

An embedded system is a specialized computing system designed to perform specific tasks or functions within a larger mechanical or electrical system. Unlike general-purpose computers, which are designed for a wide range of applications and user interactions, embedded systems are dedicated to executing predefined tasks efficiently and reliably. These systems are typically embedded into devices, equipment, or machinery to control, monitor, or interface with the physical world.

➢ *Examples:*

- Washing Machine
- Digital Camera
- Air Conditioner
- Navigation Systems
- GPS Receivers

➢ *An Embedded System Major Component:*

Embedded systems consist of several major components that work together to perform specific tasks or functions efficiently. Here are the key components of an embedded system along with brief explanations for each:

➢ *Microcontroller or Microprocessor:*

- The microcontroller or microprocessor is the central processing unit (CPU) of the embedded system.
- Microcontrollers are typically used in embedded systems because they integrate the CPU, memory, input/output (I/O) ports, and other peripherals on a single chip.
- Microprocessors are more powerful but may require additional external components for complete functionality.
- They execute the software instructions and control the operation of the embedded system.

➢ *Memory:*

- Embedded systems require memory to store program code, data, variables, and temporary information during operation.
- There are two main types of memory in embedded systems:

- *ROM (Read-Only Memory):*

Stores permanent program code and data that should not change during normal operation.

- *RAM (Random Access Memory):*

Stores temporary data, variables, stack information, and dynamically allocated memory during runtime.

➢ *Input/Output (I/O) Interfaces:*

- I/O interfaces allow the embedded system to communicate with external devices, sensors, actuators, and human interfaces.
- Common I/O interfaces include digital input/output pins, analog – to -digital converters (ADCs), digital – to -analog converters (DACs), serial communication interfaces (UART, SPI, I2C), Ethernet ports, USB ports, and display interfaces (LCD, LED, touchscreen).

➢ *Sensors and Actuators:*

- Sensors detect physical parameters such as temperature, pressure, light, motion, proximity, and more. They convert these analog signals into digital data that the microcontroller can process.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAR917

- Actuators, on the other hand, are used to control physical processes or devices based on the input from the microcontroller. Examples include motors, relays, solenoids, valves, and LEDs.

➢ *Power Supply:*

- Embedded systems require a power supply to operate. The power supply may be from batteries, external power sources, or power management circuits.
- Power management circuits ensure that the embedded system operates within specified voltage and current limits, and they may include voltage regulators, battery charging circuits, and power-saving features.

➢ *Communication Interfaces:*

- Communication interfaces enable the embedded system to communicate with other devices, systems, or networks. This includes wired and wireless communication protocols.
- Common communication interfaces include UART (Universal Asynchronous Receiver-Transmitter), SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit), Ethernet, Wi-Fi, Bluetooth, Zigbee, and CAN (Controller Area Network).

➢ *Operating System (OS) or Firmware:*

- Embedded systems may run on a real-time operating system (RTOS), a lightweight operating system, or custom firmware.
- The OS or firmware manages system resources, scheduling tasks, handling interrupts, providing drivers for peripherals, and supporting application software development.

➢ *User Interface (UI):*

- Some embedded systems have user interfaces for interaction with users or operators. UI components may include buttons, switches, displays (LCD, LED), touchscreens, keypads, and indicators (LEDs, buzzers).

➢ *Applications of Embedded Systems:*

- Networking and Communication.
- Aerospace and Defense.
- Home Automation and IoT
- Medical Devices**.**

➢ *Embedded Systems Development environment:*

An Embedded Systems Development Environment refers to the tools, software, and hardware used to develop software for embedded systems. Embedded systems are specialized computing devices designed to perform specific functions within a larger system. These systems can range from simple microcontrollers in household appliances to complex systems in automotive, aerospace, and industrial applications.

➢ *Types of Embedded Processor Architectures:*

- Based on Program Memory and Data Memory (Harvard and Von Neumann).
- Based on Mechanism (RISC and CISC).
- Based on Data transfer to the memory (Little Endian and Big Endian).
- Based on Types of Address (I/O mapped I/O and Memory Mapped I/O).

➢ *Based on Program Memory and Data Memory (Harvard and Von Neumann):*

- *Memory Block:*

Von Neumann Architecture uses a single memory unit to store both data and instructions. This unified memory structure is often referred to as the "Von Neumann bottleneck" because it can lead to performance limitations when the CPU must access instructions and data from the same memory sequentially.

III. WHAT IS VON NEUMANN ARCHITECTURE

Von Neumann Architecture is a computer architecture model proposed by John von Neumann in the late 1940s. It is named after the mathematician and computer scientist John von Neumann, who played a significant role in the development of modern computing concepts.

➢ *The Von Neumann Architecture is Characterized by the Following key Components and Principles:*

- Central Processing Unit.
- Memory
- Control Unit
- Data Bus
- Stored Program Concept

Von Neumann Architecture revolutionized computer design by introducing a unified memory structure and the concept of stored programs, laying the foundation for modern computer systems. However, it also has limitations, such as the potential for bottlenecks due to the shared memory bus and the inability to perform parallel processing efficiently. These limitations have led to alternative architectures, such as Harvard Architecture and modified versions of Von Neumann Architecture with improved performance characteristics.

➢ *What is Harvard Architecture?*

Harvard Architecture is a computer architecture design that separates memory into separate address spaces for instructions and data. It is named after the Harvard Mark I computer, developed at Harvard University in the 1940s, which first implemented this architecture. Unlike the Von Neumann Architecture, which uses a single memory space for both instructions and data, Harvard Architecture has distinct memory spaces for program instructions and data.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAR917

➢ *Key Features of Harvard Architecture Include:*

- Separate Memory Spaces.
- Dual Memory Buses.
- Instruction and Data Pipelining.
- Program Memory Protection.

Overall, Harvard Architecture offers a structured and efficient approach to organizing memory and processing elements, particularly suitable for systems that require high performance and real-time responsiveness, such as embedded systems, digital signal processors (DSPs), and microcontrollers.

➢ *Difference between Von Neumann and Harvard Architecture?*

The Von Neumann Architecture and Harvard Architecture are two distinct computer architecture designs that differ primarily in how they handle memory organization and access. Here are the key differences between Von Neumann and Harvard Architecture:

Table 1 Difference between Von Neumann and Harvard Architecture

Name	Von Neumann Architecture	Harvard Architecture
Memory Organization	It uses a single memory space for both instructions and data.	It employs separate memory spaces for instructions and data.
Memory Access	Instructions and data share the same bus for the accessing the memory.	Sperate buses for instruction fetch and data access.
Instruction and Data access Timing	Due to the shared memory space, the timing of instruction and data access may be affected by each other, potentially leading to performance limitations.	Since instructions and data have dedicated memory spaces and buses, their access timing is independent, allowing for simultaneous access and faster execution in certain scenarios.
Program Memory Protection	Program memory protection can be challenging to implement in Von Neumann Architecture due to the unified memory space.	Harvard Architecture makes program memory protection more feasible by separating instruction and data memory spaces.
Usages and Applications	It is commonly used in general-purpose computers, servers and desktop systems	Microcontrollers, digital signal processing, etc.

In summary, the main difference between Von Neumann and Harvard Architecture lies in their memory organization and access mechanisms. Von Neumann Architecture uses a unified memory space for instructions and data, while Harvard Architecture separates them into distinct memory areas with separate buses, allowing for parallel access and improved performance.

➢ *Based on Mechanism (RISC and CISC):*

RISC and CISC are two different types of computer architectures that are designed the microprocessors that are found in computers.

- RISC: Reduced Instruction Set Computer.
- CSIC: Complex Instruction Set Computer.

➢ *What is RISC Architecture?*

RISC (Reduced Instruction Set Computing) Architecture is a computer design philosophy that emphasizes simplicity and efficiency in instruction execution. It contrasts with Complex Instruction Set Computing (CISC) Architecture, which focuses on providing a wide variety of complex instructions that can perform multiple operations in a single instruction. RISC Architecture became popular in the 1980s as a response to the complexity and inefficiency of CISC architectures.

- **Ex:** ARM, MIPS (microprocessor without interlocked pipeline stages), power Architecture.

➢ *What is CISC Architecture?*

CISC (Complex Instruction Set Computing) Architecture is a computer design philosophy that emphasizes providing a wide variety of complex and powerful instructions that can perform multiple operations in a single instruction. CISC architectures contrast with RISC (Reduced Instruction Set Computing) architectures, which prioritize simplicity and efficiency in instruction execution.

- **Ex**: Motorola 68k, Digital Equipment Corporation Alpha, Virtual Address Extension.

➢ *The Following are some Important Characteristics of a RISC and CISC Processors-*

- *RISC:*

✓ Simplified Instruction Set.
✓ Single – Cycle Execution.
✓ Register – Register Architecture.
✓ Load and Store.
✓ Pipeline Execution.
✓ Efficient for pipelining and parallelism.

- *CISC:*

✓ Rich Instruction Set.
✓ Variable – Length Instructions.
✓ Memory to Memory Operations.
✓ Complex Addressing Modes.
✓ Hardware Support for High – Level Constructs.
✓ Microcode.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAR917

➢ *Difference between RISC and CISC?*

Table 2 Difference between RISC and CISC

Basis for Comparison	RISC Architecture	CISC Architecture
Instruction set	Small, simple, and uniform	Large, diverse, and complex
Execution speed	Faster due to simpler instructions	Maybe slow because of irregular size of instruction set or complex instruction set
Compiler dependency	Relies more on complier optimization	Requires less to no complier intervention.
Code size	Smaller due to simple instruction	Large due to complex instruction set
Pipeline	Often features in-depth pipelines	Pipelines are comparatively shallower
Memory access	Frequently use the load-store architecture for memory access	Supports direct memory access
Example	ARM, MIPS	X86, IBM System

Overall, RISC architectures prioritize simplicity, speed, and efficient use of hardware resources, while CISC architectures offer a rich instruction set with complex operations and support for diverse programming constructs. Each architecture has its strengths and weaknesses, making them suitable for different types of computing tasks and applications.

➢ *Based on Data Transfer to the Memory (Little Endian and Big Endian):*

Data transfer to memory in computing systems can follow two different byte orderings: Little Endian and Big Endian.

➢ *What is Little Endian?*

In Little Endian byte ordering, the Least Significant Byte (LSB) of a multi – byte is stored at the lowest memory address, and the Most Significant Byte (MSB) is stored at the highest memory address.

➢ *What is Big Endian?*

In Big Endian byte ordering, the most significant byte (MSB) of a multi-byte data type is stored at the lowest memory address, and the least significant byte (LSB) is stored at the highest memory address.

➢ *Difference between Little Endian and Big Endian?*

Table 3 Difference between Little Endian and Big Endian

Little Endian	Big Endian
Easier addition and multiplication of multiprecision number.	Easier to determine sign of a number
Requires more number of comparisons for comparing two numbers	Easier to divide two numbers
Not easier to print	Easier to print
Ex: AMD, Intel systems	Ex: IBM systems, TCP/IP

➢ *Based on Types of Address (I/O Mapped I/O and Memory Mapped I/O):*

Address types in computer systems refer to how devices or components are accessed by the CPU. There are two main types of addressing methods: I/O mapped I/O and Memory mapped I/O.

➢ *What is I/O Mapped I/O? (Input – Output Mapped Input – Output)*

- In I/O mapped I/O, a separate address space is used for accessing I/O devices.
- Support IN and OUT Assembly Instructions.
- Separate control lines for I/O.

➢ *What is Memory Mapped I/O?*

- Common address bus for I/O and Memory.
- There are no sperate IN and OUT instructions. What are the instructions are to communicate memory same instructions useful to communicate I/O.
- No sperate lines for I/O. Memory control lines only MEMR, MEMW.

➢ *Difference between I/O Mapped I/O and Memory Mapped I/O?*

Table 4 Difference between I/O Mapped I/O and Memory Mapped I/O

I/O Mapped I/O	Memory Mapped I/O
I/O peripherals are allotted separate or isolated, special address spaces called ports	I/O peripherals share the same instructions and address space
The size of the address space is about 8 bits.	The size of the address space is about 16 bits.
The type of instructions involved in I/O writing and I/O reading.	The type of instructions involved in memory writing and memory reading.
Memory and I/O devices have separate and distinct types of instructions for the transfer of data.	Both memory and I/O devices share the same type of instructions for the transfer of data.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24MAR917

IV. CONCLUSION

In conclusion, embedded systems play a vital role in modern society, powering a wide array of devices and technologies across various industries. Their seamless integration, real-time responsiveness, resource efficiency, and adaptability contribute to their widespread adoption and ongoing innovation. However, addressing challenges such as security vulnerabilities and managing increasing software complexity remains crucial for the continued success and advancement of embedded systems.

REFERENCES

[1]. https://www.heavy.ai/technical-glossary/embedded-systems.
[2]. https://www.refreshnotes.com/2016/02/von-neumann-vs-harvard-architecture.html.
[3]. https://medium.com/@csoham358/a-beginners-guide-to-risc-and-cisc-architectures-fc9af424db3b.
[4]. https://getkt.com/2019/04/06/endianness-little-endian-vs-big-endian/.