CHAPTER 1

Introduction to Embedded System and Microcontroller

Objectives ...

- To understand the Concepts of Embedded Systems and Microcontrollers.
- To explore the Design Metrics.
- To learn various Software Development Tools.
- To understand Microcontroller, their Classifications and Applications.
- To know Harvard and Von-Neumann Architecture and Difference between RISC vs CISC.

INTRODUCTION

Now-a-days everyone is familiar with a term embedded system as it is used in cell phones, remote controllers, washing machines, microwave ovens, fax machines etc. The meaning of **embedded system** is the system which performs some specific tasks. In other words, an embedded products uses microprocessor (or microcontroller) to do one and only one task. These systems are not generalized. They are assigned some specific sequence of operations and requires sort of automation to control the sequence of operations.

In this chapter, characteristics, applications, block diagram of embedded system are discussed. The design metrics including the definitions of various parameters are also included in this chapter. The last part of the chapter consists of different types of processor architectures and the differences between RISC and CISC.

1.1 INTRODUCTION TO EMBEDDED SYSTEM

1.1.1 Embedded System

- An embedded system is a computer system which can perform many tasks such as to access, to process, to store, to control the data in various electronic systems.
- Embedded system is a combination of hardware and software.
- Software is usually known as firmware that is embedded into the hardware.
- Embedded systems are application specific, organized hardware which can be controlled by specific software.
- One of the important features of the embedded systems is that, it gives the output within the specified time limits.

- Embedded systems are used in many appliances which are used in our daily life. It is used in microwave, calculators, TV, remote control, home security systems, traffic control system etc.
- The hardware of embedded system mainly consists of power source, microcontroller/ microprocessor, timers, memory, I/O devises etc.
- The software consists of compliers, integrated development environment (IDE), assembler, simulators etc.
- Usually, embedded C, embedded C++, embedded JAVA, assembly language are used in embedded system programming.
- The block diagram of embedded systems is shown in Fig. 1.1.
- As shown in Fig. 1.1, embedded system consists of processor, program memory, data memory, processor, power supply, parallel ports, serial communication ports, I/O interfacing devices and application specific circuits.

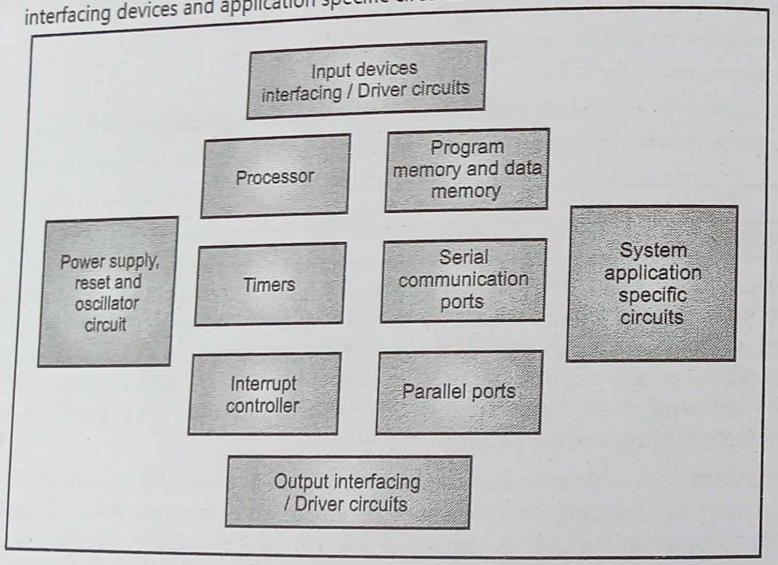

Fig. 1.1: Block diagram of Embedded system

- As mentioned earlier, embedded system consists of hardware and software designed for a specific application.
- The illustration of basic structure of embedded system is shown in Fig. 1.2.

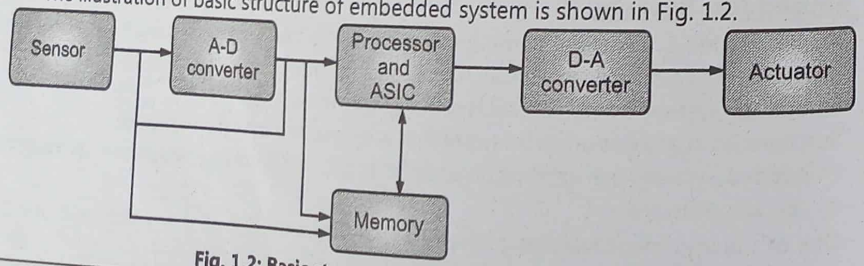

Fig. 1.2: Basic structure of Embedded system

- As shown in Fig. 1.2, embedded system consists of sensor, ADC (Analog to Digital Converter), processor, DAC (Digital to Analog Converter), actuators, memory etc.
- Sensor measures the physical quantity and converts it into an electrical signal. The output of a sensor is usually in analog form.
- ADC converts this analog output received by sensor into digital signal.
- Processor processes the data, calculates the output and stores it in memory.
- As this data has to be transmitted for further process or action, it has to be converted into analog form again.
- DAC converts this digital output into analog form.
- Actuator compares the data with the expected output which is stored in memory and then further action is taken by the actuator.

Elements of Embedded Systems:

- Processor
- Memory
- Input device
- Output device
- Communication interface
- A/D, D/A converters
- Power supply

Characteristics of Embedded Systems:

- Embedded system usually performs a specialized operation
- Embedded system must be of size to fit on a single chip.
- It must perform fast enough to process the data in real time.
- It has to consume less power to extent battery life.
- It monitors the data without any delay to give real time outputs.
- Microprocessor or microcontroller is used as a processor in embedded system.
- Memory of embedded system has to be sufficient to store its software.
- It should contain peripherals to connect input/output devices.
- Software must be flexible and secure with more features.

Advantages of Embedded System:

- Low power consumption.
- Low cost.
- Small size.
- Enhanced real time performance.
- Easily customizable for a specific application.

Disadvantages of Embedded System:

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

Applications of Embedded Systems:

- Smartphones and Tablets.
- Smart TVs.
- Home appliances such as washing machines, microwave ovens, air conditioners.
- Programmable Logic controllers (PLCs).
- Robotic systems.
- Supervisory Control and Data Acquisition (SCADA).
- Pacemakers and Insulin Pumps.
- Patient monitoring systems.
- Mobile base stations, MODEMs.
- Home automation systems.
- Wearable devices.
- Automated irrigation systems.
- Drones.

1.1.2 Design Metrics

There are several key design metrics in the design of an embedded systems which are considered to ensure the product meets the performance, economic and practical requirements.

Major key design metrics are discussed below:

1. **NRE Cost:**
   - Non-Recurring Engineering (NRE) cost is defined as, "the one-time cost incurred for research design, development and testing of the system".
   - It is the cost required for hardware prototyping, software development, certification etc.
   - NRE should be minimum maintaining quality and functionality.
   - For the mass production, high NRE is acceptable.
2. **Unit cost:**
   - Unit cost is defined as, "the cost to produce a single unit of the system (excluding NRE)".
   - It includes hardware components, manufacturing, assembly and testing.
   - Unit cost should be optimal to stay competitive in the market.
3. **Time to market:**
   - Time to market is defined as, "the time taken from concept development to product availability".
   - Time to market should be less so that product can gain competitive advantage and earlier revenue.
   - Late entry into the market may result in loss of opportunities and further reduce profits.
4. **Safety:**
   - Safety is defined as the, "system's ability to operate without causing harm to users or environment".
   - The system should ensure fault tolerance and compliance with the standards.
   - In medical devises, aerospace, industrial control systems safety is a crucial factor.

5. **Maintenance:**
   - Maintenance is the ease of updating, repairing and upgrading the system after deployment.
   - It mainly includes software repairing, hardware servicing.
   - Maintenance should be minimum which can increase the reliability of the system and reduce overall lifecycle cost.
6. **Size:**
   - Size is the physical dimension and footprint of the system.
   - Size should be minimum to achieve compactness and system can be fit into small size.
   - In case of wearable, consumer electronics, automotive system, size plays a vital role.
7. **Total Cost:**
   - Total cost is the combined cost of NRE and unit cost plus operational and maintenance costs.
   - Total cost should balance with performance and quality to maximise return on investment.
8. **Power Dissipation:**
   - Power dissipation is defined as, "the amount of electric power consumed and converted into heat".
   - Power dissipation should be low especially for battery operated systems such as smart phones.

Table 1.1

Metric	Description	Goal / Objective
NRE Cost	One-time development and setup cost	Minimize for faster ROI
Unit Cost	Per-unit production cost	Minimize for profitability
Time to Market	Time from idea to product availability	Shorten for competitive advantage
Safety	Risk of harm or failure	Maximize reliability and fault tolerance
Maintenance	Ease of updates and repairs	Simplify for lower lifecycle cost
Size	Physical dimensions	Minimize for portability/ fit
Cost (Total)	Overall cost including NRE and unit cost	Optimize balance with performance
Power Dissipation	Power consumed and converted to heat	Reduce for efficiency and longer battery life

1.1.3 Software Development Tools

In development of embedded systems, a variety of software tools are used to write, compile, debug and program applications for specific hardware platforms.

There are various software development tools such as editor, assembler, linker, compiler, IDE, ICE, programmer and simulator. The function of each tool is different.

1. **Editor:** The editor is the basic tool used to write and edit source code, typically in high-level languages like C or C++. Editor can be a simple text editor like Notepad or an advanced one integrated into an IDE (Integrated Development Environment).
2. **Assembler:** Once the code is written, it will be translated into a machine-readable format. If the code is written in assembly language, an assembler is used to convert it into object code. It translates ".asm" files to binary.
3. **Linker:** The linker combines individual files into a single executable file after compilation. It produces the file having extension ".hex" or" .bin" that can be loaded into the embedded systems
4. **Complier:** For high-level languages, a compiler is used to convert code into machine readable file. The compiler translates the source code into low-level object code that the microcontroller can execute.

   Compiler converts high-level language (like C/C++) into machine code.
5. **IDE:** All the tasks such as editing, compiling, assembling, linking and even debugging are typically integrated into an IDE or Integrated Development Environment. IDE usually have user friendly interface for entire development process.
6. **ICE:** In-circuit Emulator is a hardware device that emulates the microcontroller and allows real-time debugging by stepping through code while monitoring the hardware behavior. For accurate real-world debugging, developers often use ICE.
7. **Programmer:** Programmer is a hardware tool which uploads the final binary to target device. Programmer vary depending on the microcontroller family. For example, AVR ISP programmer is used for AVR family.
8. **Simulator:** Simulator test the code virtually and check how it would behave in real conditions. Simulator helps to test the software logic without requiring the actual hardware.

The summery of all the softwares is given in below table:

Table 1.2

Tool	Purpose	Details / Examples
Editor	Used to write and modify source code.	Text editors like Notepad ++, VS Code, or editors within IDEs.
Assembler	Converts assembly language code into machine code (object code).	Translates .asm files to binary understood by hardware.
Compiler	Converts high-level language (like C/C++) into machine code.	Examples: GCC, Keil C compiler, IAR compiler.
Linker	Combines object files and libraries into a single executable file.	Resolves symbols and addresses. Produces final .hex or .bin file.
IDE (Integrated Development Environment)	Provides a unified interface for coding, compiling, debugging, and simulating.	Examples: Keil µVision, Eclipse, Code Composer Studio, MPLAB X.
ICE (In-Circuit Emulator)	Hardware device that emulates the target microcontroller for debugging in real time.	Used to test actual hardware interactions. High precision.
Programmer	Transfers the final program (hex/bin file) to the embedded system's memory.	Examples: AVR ISP, PICkit, JTAG tools.
Simulator	Simulates the behavior of the embedded system without real hardware.	Tests logic, I/O, timing, and execution in software. Examples: Proteus, Multisim, Keil simulator.

contd. ...

Typical workflow of these tools is as follows:

1. **Editor:** Write source code (e.g., in C).
2. **Compiler:** Convert source code into object files.
3. **Assembler:** Assemble any assembly code used.
4. **Linker:** Combine all object files into a single executable file.
5. **Simulator:** Test the code virtually before deploying to hardware.
6. **ICE / Debugger:** Debug using actual or emulated hardware.
7. **Programmer:** Burn the final executable to the microcontroller.
8. **IDE:** Often integrates all of the above into a single environment for a particular microcontroller.

1.2 MICROCONTROLLER

1.2.1 History

- The microcontroller is a key component in embedded systems today, but its journey began decades ago.
- In 1971, first microcontroller was invented by Texas Instruments known as the TMS1000. It was developed for use in calculators.
- In 1974, Intel developed Intel 8048 microcontroller which was used in IMB PC keyboard. It has integrated RAM, ROM and I/O which is suitable for standalone embedded systems.
- In 1980, Intel introduced the 8051 microcontroller, which became one of the most popular microcontroller in history. It has 8-bit architecture, 4KB ROM, 128 bytes RAM and I/O ports. It became a benchmark for many embedded applications and is still in use today in modified forms.
- From 1980-1990, many companies entered into the micro-controller market. Microchip developed PIC microcontroller, Atmel developed AVR series, Motorola developed 68HC11. Also, Mitsubishi, Hitachi, Toshiba, and NEC in Japan started developing microcontrollers.
- In 2000, ARM cortex-M microcontrollers came into the market which has high performance, low power consumption and scalable architecture.
- From 2010 till present, microcontrollers were optimised for IoT (Internet of Things) applications which has in-built wireless communication facility (Bluetooth, Wi-fi). It

supports low power consumption and real time control. Open source platform like Arduino were developed which are useful for students and hobbyists. Arduino uses AVR and ARM microcontroller.

- Microcontrollers have evolved from basic control units in calculators to powerful, efficient chips embedded in nearly every modern electronic device from smartphones and appliances to cars and medical devices.

1.2.2 Introduction

- The device which controls and sequences the flow of operation in a specified manner is referred as microcontroller. Microcontrollers are assigned with some specific tasks which they perform in a systematic way.
- Microcontroller is one of the powerful control systems, which can be used for the measurement and control applications in industry as well as in a specific application.
- Microcontroller is sometimes described as a "Computer on Chip" because it contains all the features of full computer including Central Processing Unit (CPU), inbuilt clock circuitry ROM, RAM, input and output ports with special features such as serial communication, analog to digital conversion and more recently signal processing.
- It is a general-purpose device, which is meant to read data, perform limited calculations on the data and control its environment based on those calculations. The prime use of microcontroller is to control the operation of machine using a fixed program that is stored in ROM and that does not change over the lifetime of system.
- A special application of microcontroller is well suited for data logging as well as monitoring and recording the environmental, physical and chemical parameters. In addition to the above home monitoring systems, embedded processors are used in home appliances.
- Microcontrollers have many advantages as compared with the microprocessors especially in the areas where high integration and saving of space is required.

1.2.3 Classification

Microcontrollers can be classified based on several factors such as architecture, data width, memory type, instruction set and application.

(i) Based on bit architecture, there are three types:

1. **8-bit Microcontrollers:** It process 8-bit of data at a time. It is suitable for simple tasks like home appliances, toys and small gadgets. Examples: Intel 8051, Atmel AVR, Microchip PIC16.
2. **16-bit Microcontrollers:** It handle 16-bit data. It offer better performance than 8-bit. It is used in complex applications like motor control and industrial systems. Examples: PIC24 (Microchip).
3. **32-bit Microcontrollers:** It is high-speed and powerful. It can handle more complex operations and real-time systems. It is widely used in automotive, IoT, robotics and multimedia. Examples: ARM Cortex-M series (STM32, NXP, TI).

(ii) Based on the memory Architecture, there are two types:

1. **Harvard Architecture:** It has separate memory spaces for program and data.
2. **Von Neumann Architecture:** It has single memory for both program and data.

(iii) Based on Instruction Set Architecture there are two types:

1. **CISC:** CISC means 'Complex Instruction Set Computer' which has many complex instructions.
2. **RISC:** RISC means 'Reduced Instruction Set Computer' which uses fewer, simpler instructions. It is more efficient and has faster execution.

(d) Based on Application there are two types:

1. **General Purpose Microcontrollers:** These are used in a wide range of applications like home automation, consumer electronics, etc.
2. **Application-Specific Microcontrollers:** These are designed for specific applications like Automotive MCUs for engine control, Industrial MCUs for PLCs, sensors, mMedical MCUs for patient monitoring devices.

(e) Based on Memory Type:

1. **Embedded Memory Microcontrollers:** In these, Internal ROM and RAM are built into the chip. These are compact and used in mass production.
2. **External Memory Microcontrollers:** In these external ROM/RAM is used. These are more flexible, suitable for complex systems with large memory needs.

Following table gives list of all types of Microcontrollers.

Table 1.3

Criterion	Examples
Bit Width	8-bit (8051), 16-bit (MSP430), 32-bit (ARM Cortex)
Memory Architecture	Harvard (AVR), Von Neumann (ARM)
Instruction Set	RISC (ARM), CISC (8051)
Memory Type	Embedded (AVR), External (legacy)
Application Type	General purpose, automotive, medical, industrial

1.2.4 Applications

Microcontrollers are found in nearly every modern electronic device, from household appliances to industrial machinery. Few applications are listed below:

- Televisions, microwaves, washing machines, air conditioners
- Remote controls, digital clocks, toys and gaming consoles
- Parking sensors and infotainment systems
- Programmable Logic Controllers (PLCs)
- Robotic arms
- Factory automation and monitoring systems
- Heart rate monitors
- Blood pressure machines
- Modems
- Mobile phones
- Bluetooth and Wi-Fi modules
- Smart lighting

- Security systems (CCTV, smart locks)
- Automatic curtains and irrigation systems
- Home automation systems
- Navigation systems
- Flight control.

1.2.5 Differences between Microcontroller and Microprocessor

The microprocessor or CPU (Central Processing Unit) is a heart of a computer. Microprocessor mainly consists of ALU (Arithmetic Logic Unit) which performs all arithmetic and logical operations. However, microprocessor does not have memory and input/output ports inside the chip. Both have to be connected to microprocessor externally through interface unit. Microcontroller have inbuilt memory and I/O ports and other strong features. A stand alone embedded system can be build using microcontroller.

Table 1.4: The main difference between microprocessor and microcontroller

Microprocessor	Microcontroller
1. Microprocessor is used in computer system. It is the heart of computer.	1. Microcontroller is used in embedded systems.
2. It consists of Memory and I/O components which are externally connected to it.	2. Microcontroller consists of inbuilt internal memory and I/O ports.
3. Since memory and I/O are externally connected, the circuit becomes large.	3. Since memory and I/O are inbuilt, the circuit is small.
4. Cost of the entire system is more.	4. Cost of the entire system is less.
5. Total power consumption is high as compared with microcontrollers, so cannot be used in battery operated devices.	5. Total power consumption is less and can be used in battery operated devices.
6. Most of the microprocessors do not have power saving features.	6. Most of the microcontrollers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
7. Since memory and I/O components are externally connected, each instruction will need external operation, hence it is relatively slower.	7. Since components are in-built, most of the operations are done by internal instructions, hence speed is fast.
8. Microprocessors are based on Von-Neumann architecture where program and data are stored in same memory module.	8. Microcontrollers are based on Harvard architecture where program memory and Data memory are separate.
9. Mainly used in personal computers.	9. Mainly used in washing machine, MP3 players, stand alone systems.

1.2.6 Criteria for Choosing a Microcontroller

Selecting the right microcontroller for any embedded system is critical. The choice depends on several technical and practical factors based on the application's needs.

Following is the criteria to select microcontroller:

1. **Application requirements:** Check the application and its requirements, define input/output requirements, processing speed, memory size and peripherals.
2. **CPU performance:** Depending on the complexity of the application, decide CPU. For basic tasks, an 8-bit MCU may suffice; for intensive applications, 32-bit ARM Cortex is preferred.
3. **Memory:** Decide RAM and Flash memory size depending upon the application. Flash memory is used for program storage and RAM is used for data processing. Select sufficient memory to avoid overflow.
4. **Power consumption:** It is important for battery operated or portable devices. Select microcontroller with low power consumption, check sleep functions, idle power ratings of microcontroller.
5. **Input/Output ports:** Check input/output ports of microcontroller depending upon the application, ensure enough ports are available to interface peripherals.
6. **Peripheral support:** Check built-in modules such as ADC/DAC, Timers/Counters, UART, SPI, I2C, CAN, ethernet of a microcontroller. It will reduce need for external components and save board space.
7. **Cost:** The cost is a crucial factor of any application. Cost must fit within the project's budget, there should be balance between features and the cost.
8. **Availability and support:** Select microcontroller which are readily available and not obsolete. Prefer microcontroller which is widely used with strong support, documentation and technical assistance.
9. **Development Tools:** Check availability of IDE, complier, debugger, simulators, programmers. Select programmer which user friendly.
10. **Packaging and size:** Select packaging type depending on the application. Surface mounted devices (SMD) components are preferred due its small size.

While selecting a microcontroller, always start by clearly defining the requirements of the application. The best microcontroller is the one that meets technical needs of user's applications, fits the budget, and has good support and availability.

1.3 ARCHITECTURES

Harvard and Von-Neumann Architecture:

There are two main types of architectures Harvard and Von-Neumann architecture which are discussed below.

1.3.1 Von-Neumann Architecture

- In past 25 years, computer architecture has undergone tremendous changes from the number of circuits that can be integrated onto silicon wafers to the degree of sophistication with different algorithms that can be mapped directly to a computer's hardware. However, the concept of Von-Neumann's computer design remains constant.

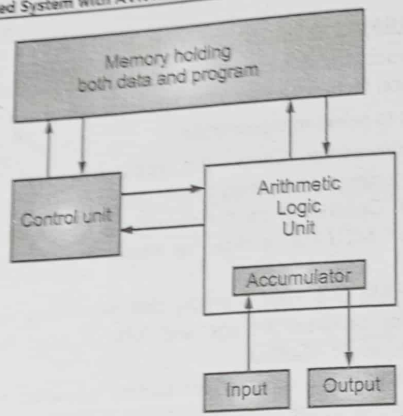

Fig. 1.3: The Von-Neumann or Stored program architecture

- The basic concept of Von-Neumann architecture is its ability to store program instructions in memory along with the data on which those instructions operate.
- Before that, each computing machine was designed and build for a single predetermined purpose in which all programming of the machine requires manual wiring of circuits. It is very difficult to detect the mistake in manual wiring.
- Von-Neumann architecture is composed of three distinct components: a central processing unit (CPU), memory and input/output (I/O) interfaces. Fig. 1.3 represents one of several possible ways of interconnecting these components.
- Fig. 1.3 shows the essential features of the Von-Neumann or stored-program architecture. It basically consists of memory, control unit, input/output devices and buses to carry the data and program.
- Memory stores both, data and program processing that data. In modern computers, this memory is nothing but the RAM.
- Control unit will control the process of transferring the data and program into and out of memory and also it manages the execution of program instructions 'one at a time'.
- In this, registers will hold the intermediate values (for e.g. Accumulator is one of such register).
- The 'one-at-a-time' phrase means that, the Von-Neumann architecture is a sequential processing machine.
- In this architecture, the user can interact through Input-Output devices. The values that are passed to and forth are stored once again in some internal registers.
- Arithmetic Logic Unit of this architecture carry out all the arithmetic and logical operations such as addition, multiplication, subtraction, division and also comparisons (such as greater than, less than or equal to) are carried out by this unit.
- The arrows between the blocks are the buses (a bunch of wires) which carry the data and programs from one place to another place. There are three main buses; address bus which carries address, data bus which carries data and control bus which carries control signals.

1.3.2 Harvard Architecture

- The Harvard architecture has separate storage and signal pathways for instruction and data. The term is originated from the Harvard Mark relay-based computer, which stores instructions on punched tape and data in electro-mechanical counters.
- These machines have limited data storage and it is entirely contained within the central processing unit and provided no access to the instruction storage as data. The programs are loaded by an operator as processor could not boot itself.
- The block diagram of Harvard architecture is shown in Fig. 1.4.

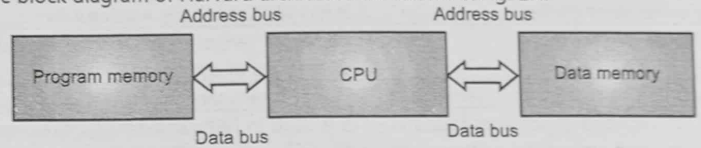

Fig. 1.4: Block diagram of Harvard architecture

- The Harvard architecture uses physically separate memories for their instructions and data, requiring dedicated buses for each of them. Instructions and operands can therefore be fetched simultaneously.
- In the Harvard architecture, the CPU can both read an instruction and perform a data memory access at the same time, even without a cache. A Harvard architecture computer can thus be faster for a given circuit complexity because instruction fetches and data access do not contend for a single memory pathway.
- At higher clock speeds, caches are useful as the memory speed is proportionally slower. Harvard architectures tend to be targeted at higher performance systems, and so caches are nearly always used in such systems.
- In Harvard architecture, as the program and data memory are separate, it does not matter if both are fed at the same time and also it may come from a cache memory. However, there can be a problem with compilation. Compilers generally embed data within the code and it is often also necessary to be able to write to the instruction memory space.
- Harvard architecture is highly inefficient if it is used with a simple unified memory. It is always better to use Von-Neumann architecture which uses separate buses for data and program.

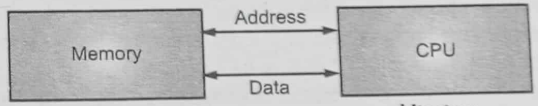

Fig. 1.5: Von-Neumann processor architecture

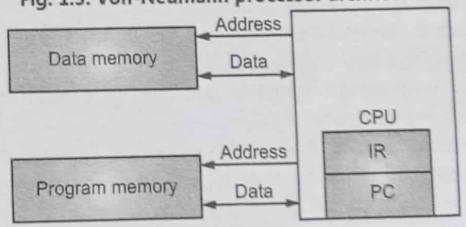

Fig. 1.6: Harvard processor architecture

1.3.3 Difference between Von-Neumann and Harvard Processor Architecture

Table 1.5

Von-Neumann Architecture	Harvard Architecture
The data and program are stored in the same memory.	The data and program memories are separate.
The code is executed serially and takes more clock cycles.	The code is executed in parallel.
There is no exclusive multiplier.	It has MAC (Multiply Accumulate).
Absence of Barrel Shifter.	Barrel Shifter help in shifting and rotating operations of the data.
The programs can be optimized in lesser size.	The program tend to grow big in size.
It is used in conventional processors found in PCs and Servers, and embedded systems with only control functions.	It is used in DSPs and other processors and found in latest embedded systems such as mobile communication systems, audio, speech, image processing systems.
Von-Neumann architectures usually have a single unified cache, which stores both instructions and data.	Harvard architectures have separate caches for each bus.

1.3.4 RISC vs CISC

- In general, computer architectures evolved greater complexity, larger instruction set, more addressing modes, more computa-tional power of the individual instructions, more specialized registers and so on.
- Recent machines falling within such trends are termed as Complex Instruction Set Computer (CISC). Most of the PC's use CPU which are based on this architecture. For example, Intel and AMD CPU's are based on CISC architectures. Typically, CISC chips have a large amount of different and complex instructions.
- The reason behind it is that, hardware is always faster than software. Therefore, one should make a powerful instruction set which provides users with assembly instruction to do a lot with short programs.
- However, a point is reached where the addition of a complex instruction to an instruction set affects the efficiency and the cost of the processor. So, before adding the instructions to the instruction set, the effect of it should be evaluated.
- It was found that some of the instructions provided by CISC processors are so esoteric that many compliers do not attempt to use them.
- Further, having such a powerful instruction set, it is difficult to imagine that they would be used very frequently. This basic concept of not adding such instructions to the instruction set has invoked an increasing interest in invention of computer architecture known as Reduced Instruction Set Computer (RISC).
- RISC processors such as IBM Power PC processor have a simplified and reduced instruction set (about 100 instructions or less). Addressing modes are simplified back to four or less and the length of the codes is fixed in order to allow standardization across the instruction set.

- In common, CISC chips are relatively slow (compared to RISC chips) per instruction, but use little (less than RISC) instructions.
- RISC has fewer and faster instructions than the large, complex and slower CISC instructions. However, more instructions are needed to accomplish a task.
- An advantage of RISC is that it has more simple instructions, it requires fewer transistors which makes them easier to design and cheaper to produce. Also it is easier to write powerful optimised compliers with the fewer instructions.
- But RISC puts a greater burden on the software since its hardware is simpler. Software needs to become more complex. Software developers need to write more lines for the same.

The Features of RISC Processors:

1. Reduced number of instructions
2. Limited addressing modes
3. Large register set
4. Limited number of instruction size
5. Limited number of instruction format

The Features of CISC Processor:

1. Complex instruction set
2. High level instruction set
3. Chips are relatively slow compared with RISC
4. Executes several low level operations (for e.g. Load or arithmetic operation)

Table 1.6: Comparison between RISC and CISC

CISC	RISC
Complex instructions requires multiple cycles.	Reduced instructions take 1 cycle.
Many instructions can reference memory.	Only Load and Store instructions can reference memory.
Instructions are executed one at a time.	Uses pipelining to execute instructions.
Fewer general instructions.	Many general instructions.

- Basically, RISC is designed to maximize speed of operation or minimize execution time, to minimize development cost and sale price.

Table 1.7: Comparison of RISC and CISC based on different parameters

Parameter	CISC	RISC
Instruction Set	Large (100 to 300)	Small (100 or less)
Addressing Modes	Complex (8 to 20)	Simple (4 or less)
Instruction Format	Specialised	Simple
Code Lengths	Variable	Fixed
Execution Cycles	Variable	Standard for most
Cost / CPU Complexity	Higher	Lower
Simplifies	Compilation	Processor design
Complicates	Processor design	Software

- Now-a-days, these two architectures are converging closer to each other, with CPUs from each side incorporating ideas from the other.
- CISC processor uses many of the same techniques as RISC and the instruction set of RISC contain similar number of instructions which are found in CISC. However, it is important to understand the architectures of these two designs.

(d) Concept of Pipelining:

- Pipelining is a technique used in computer systems, especially in microprocessors, to make instruction execution faster and more efficient by doing multiple tasks at the same time, similar to how a factory assembly line works.
- In a non-pipelined system, each instruction is processed one after the other. Each instruction must finish all its steps before the next one starts. This leads to waste of time because while one part of the processor is working, the others stay idle.
- With pipelining, the instruction is broken down into smaller steps or stages (like Fetch, Decode, Execute etc.) and these stages are handled in parallel for different instructions. So, while one instruction is being executed, the next one can be decoded, and a third one can be fetched. All is in the same clock cycle.

  A typical instruction cycle has following stages:

  1. **Fetch:** Get the instruction from memory.
  2. **Decode:** Interpret the instruction and prepare necessary signals.
  3. **Execute:** Perform the operation (e.g., arithmetic, memory access).
  4. **Memory Access:** Read or write data to/from memory (if required).
  5. **Write Back:** Write the result to a register.
- In pipelining, while one instruction is being decoded, another can be fetched and a third can be executed. This overlapping allows the processor to work on several instructions at a time, increasing efficiency.
- Assume a 5-stage pipeline. Each stage takes one clock cycle. Without pipelining, 5 instructions would take 25 cycles (5 per instruction). With pipelining, it takes only 9 cycles: the first instruction takes 5 cycles and each subsequent instruction completes every cycle after that.

Advantages of Pipelining:

- Increased throughput i.e. more instructions are completed in a shorter time.
- Efficient CPU utilization which keeps all parts of the processor busy.
- Faster execution for sequential instructions.

Exercise

[A] Multiple Choice Questions:

1. Which of the following is a characteristic of an embedded system?
   (a) General-purpose use (b) Large memory and storage
   (c) Real-time operation (d) High cost

2. Which component acts as the brain of an embedded system?
   (a) Sensor (b) Memory
   (c) Microcontroller (d) Display
3. An example of an embedded system is:
   (a) Laptop (b) Smartwatch
   (c) Supercomputer (d) Server
4. The cost of designing and developing a product before manufacturing is called:
   (a) Unit cost (b) NRE cost
   (c) Production cost (d) Maintenance cost
5. Which design metric refers to the physical dimensions of an embedded system?
   (a) Unit cost (b) Time to market
   (c) Size (d) Power dissipation
6. Which of the following is essential for battery-operated embedded systems?
   (a) High power dissipation (b) Large size
   (c) High unit cost (d) Low power consumption
7. Which tool converts high-level code into assembly language?
   (a) Assembler (b) Compiler
   (c) Linker (d) Editor
8. What does an IDE provide?
   (a) Only an editor
   (b) Only a compiler
   (c) Integrated environment for development
   (d) Just debugging tools
9. Which tool is used to simulate the behavior of an embedded system?
   (a) Programmer (b) ICE
   (c) Simulator (d) Assembler
10. Microcontrollers are mainly used in:
    (a) Desktop computers (b) Servers
    (c) Embedded applications (d) Database systems
11. Which of the following best describes a microcontroller?
    (a) CPU only (b) CPU with memory and I/O ports
    (c) Memory only (d) Storage unit
12. Which architecture has separate memory for data and instructions?
    (a) Von-Neumann (b) Harvard
    (c) CISC (d) RISC
13. Which architecture uses a single memory for both data and instructions?
    (a) Harvard (b) RISC
    (c) Von-Neumann (d) Stack

14. RISC architecture typically uses _____.
    (a) Many complex instructions
    (b) Fewer simple instructions
    (c) Microcode execution
    (d) None of the above
15. Pipelining in processors is used to:
    (a) Slow down execution
    (b) Increase instruction latency
    (c) Execute multiple instructions simultaneously
    (d) Reduce power consumption
16. Which of the following is a criterion for choosing a microcontroller?
    (a) Clock speed
    (b) Memory size
    (c) Power consumption
    (d) All of the above

Answers

1. (c)	2. (c)	3. (b)	4. (b)	5. (c)	6. (d)	7. (b)	8. (c)	9. (c)	10. (c)
11. (b)	12. (b)	13. (c)	14. (b)	15. (c)	16. (d)				

[II] State True or False:

1. Embedded systems are designed for general-purpose computing.
2. Real-time performance is a key characteristic of many embedded systems.
3. Microcontrollers are commonly used as the main processor in embedded systems.
4. Input/output devices are not considered elements of an embedded system.
5. A microwave oven is an example of an embedded system.
6. NRE (Non-Recurring Engineering) cost includes per-unit manufacturing cost.
7. Unit cost affects the feasibility of large-scale production.
8. Power dissipation is not a concern in embedded systems.
9. Time to market refers to how quickly a product can be developed and launched.
10. Safety and maintenance are ignored in embedded system design.
11. A compiler translates assembly language into machine code.
12. An IDE typically includes editor, compiler, debugger and other tools.
13. A linker is used to combine object files into a single executable file.
14. ICE (In-Circuit Emulator) is used for programming micro-controllers.
15. A simulator mimics hardware behavior to test embedded software.
16. A microcontroller typically integrates CPU, memory, and peripherals on a single chip.
17. Microcontrollers are primarily used in general-purpose desktop computers.
18. Harvard architecture uses the same memory space for data and instructions.
19. RISC architecture emphasizes a large number of simple instructions.
20. Pipelining allows multiple instructions to be processed at the same time.

Answers

1. False	2. True	3. True	4. False	5. True
6. False	7. True	8. False	9. True	10. False
11. False	12. True	13. True	14. False	15. True
16. True	17. False	18. False	19. True	20. True

[III] Fill in the Blanks:

1. An embedded system is a combination of _____ and ____ designed to perform a specific function.
2. Embedded systems are generally dedicated, real-time, and have limited ____and memory.
3. The main design metrics for embedded systems include ____ cost, unit cost, time to market, and power dissipation.
4. ____ cost refers to the one-time development cost of the system.
5. ____ cost is the cost of manufacturing a single unit of the system.
6. Software development tools include editor, ____, linker, compiler, and simulator.
7. An _____ is a software that integrates multiple development tools under one interface.
8. ICE stands for ___.
9. A ___is used to transfer the program into the microcontroller's memory.
10. ____ architecture uses separate memory and buses for program and data.
11. ____ architecture uses a single memory and bus for both program and data.
12. ___architecture uses simple instructions that execute in one clock cycle.
13. CISC architecture uses ___ instructions which may take multiple clock cycles.
14. ____ improves the performance of a processor by executing multiple instructions in an overlapping manner.
15. In pipelining, each instruction is divided into stages such as fetch, ____, and execute.

Answers

1. Hardware, software
2. processing power
3. NRE
4. NRE (Non-Recurring Engineering)
5. Unit
6. Assembler
7. IDE (Integrated Development Environment)
8. In-Circuit Emulator
9. Programmer
10. Harvard
11. Von-Neumann
12. RISC
13. Complex
14. Pipelining
15. decode

[IV] Short Answer Questions:

1. What is an embedded system?
2. List any four characteristics of embedded systems.
3. Name any three elements of an embedded system.
4. Give two examples of real-life applications of embedded systems.
5. What is NRE cost in embedded system design?

6. Define unit cost.
7. Why is time-to-market important in embedded system design?
8. Mention two factors that affect the power dissipation of an embedded system.
9. What is the role of an editor in embedded software development?
10. What is the function of a compiler?
11. What does an assembler do?
12. What is the purpose of a simulator?
13. What is an IDE? Give one example.
14. Define a microcontroller.
15. List two differences between a microcontroller and a microprocessor.
16. Name any two applications of microcontrollers.
17. On what criteria should a microcontroller be selected? (Any two)
18. What is the Harvard architecture?
19. What is the Von-Neumann architecture?
20. Write one difference between RISC and CISC.
21. What is pipelining in processor architecture.

[V] Long Answer Questions:

1. Explain the characteristics of embedded systems with suitable examples. 2
2. Describe the elements of an embedded system. Explain the role of each component. 2
3. Discuss at least five applications of embedded systems in various domains. 3
4. What are design metrics in embedded system development? Explain the significance of NRE cost, unit cost, time to market, size, power dissipation, safety and maintenance. 3
5. Explain how power dissipation and size influence the design of an embedded system. Provide real-world examples. 3
6. Draw the block diagram of embedded system. Explain any three blocks. 2
7. Describe the software development tools used in embedded systems. Explain the role of each: editor, assembler, linker, compiler, IDE, ICE, programmer and simulator. 3
8. What is an IDE? Explain its features and advantages in embedded software development. 4
9. Write a detailed note on the history and classification of microcontrollers. 5
10. Compare microcontrollers and microprocessors. Mention at least five differences with examples. 6
11. Discuss the criteria for selecting a microcontroller for an embedded application. 6
12. Explain Harvard and Von-Neumann architectures with suitable diagrams. State their advantages and disadvantages. 7
13. What is the difference between RISC and CISC architectures? Discuss with advantages, disadvantages and examples. 8
14. Explain the concept of pipelining in microprocessor architecture. What are its advantages and challenges? 9
15. Differentiate between Harvard and Von-Neumann architectures 8

Syllabus ...

1. **Introduction to Embedded System and Microcontroller [06 Lectures]**
   - **Introduction to Embedded Systems:** Embedded Systems: Introduction, Characteristics, Elements and Applications. Design Metrics: NRE Cost, Unit Cost, Time to Market, Safety, Maintenance, Size, Cost and Power Dissipation. Software Development Tools: Editor, Assembler, Linker, Compiler, IDE, ICE, Programmer and Simulator.
   - **Microcontroller and Architectures:** History, Introduction, Classification, Applications. Differences between Microcontroller and Microprocessor, Criteria for Choosing a Microcontroller. Architectures - Harvard and Von-Neumann Architecture, RISC vs CISC. Concept of Pipelining.
2. **Fundamentals of AVR and Its Programming in C [06 Lectures]**
   - **AVR Architecture:** Overview of AVR, Classification of AVR Family, AVR (ATmega16/32) Architecture, AVR Processor Memory Map, CPU Registers, ALU, I/O Ports, Peripherals in AVR.
   - **Programming of AVR in C:** Basic Structure, Data Types, Operators, Library Files, Delay Functions and Bitwise Operation Syntax. Simple C Programs: Data Transfer Operation, Arithmetic Operation, Decision-making and Code Conversion.
3. **AVR Peripherals Programming in C [10 Lectures]**
   - **AVR Timer Programming:** Introduction, Difference between Timer and Counter operation, Basic SFR Registers used – Timer 0, 1 & 2, C Programs for Delay Generation, Counter Programming.
   - **AVR Serial Port Programming:** Basics of Serial Communication (Serial Vs Parallel, Simplex Vs Duplex), Difference between Asynchronous and Synchronous Communication, USART Operation, SFR used, C Programs for Data Transmission and Reception.
   - **I2C and SPI:** Introduction, Specifications, Bus Signals, Master-slave Configuration, Error Handling and Addressing. SFR used in AVR, C Programs to Transfer and Receive Information.
   - **On-chip ADC:** Features, Block Diagram, Operation, SFR used, C Programs to Convert the Analog Signal to Digital.
4. **Real World Interfacing with AVR and Case Studies [08 Lectures]**
   - **I/O Device Interfacing:** LED, Push Button, Buzzer, Seven Segment Display, Thumbwheel Switch, DC and Stepper Motor, Relay Interfacing, 16*2 LCD Interfacing, DAC Interfacing (Waveform Generation using DAC).
   - **Case Studies:** Traffic Light Controller using AVR, Single Digit Event Counter using Opto-interrupter and SSD, Real Time Clock using IC DS1307 Chip, Temperature Monitoring System using LM35 Sensor, Smart Phone Controlled Devices using Bluetooth Module HC05.

■■■