1. ES – Overview

System

A system is an arrangement in which all its unit assemble work together according to a set of rules. It can also be defined as a way of working, organizing or doing one or many tasks according to a fixed plan. For example, a watch is a time displaying system. Its components follow a set of rules to show time. If one of its parts fails, the watch will stop working. So we can say, in a system, all its subcomponents depend on each other.

Embedded System

As its name suggests, Embedded means something that is attached to another thing. An embedded system can be thought of as a computer hardware system having software embedded in it. An embedded system can be an independent system or it can be a part of a large system. An embedded system is a microcontroller or microprocessor based system which is designed to perform a specific task. For example, a fire alarm is an embedded system; it will sense only smoke.

An embedded system has three components:

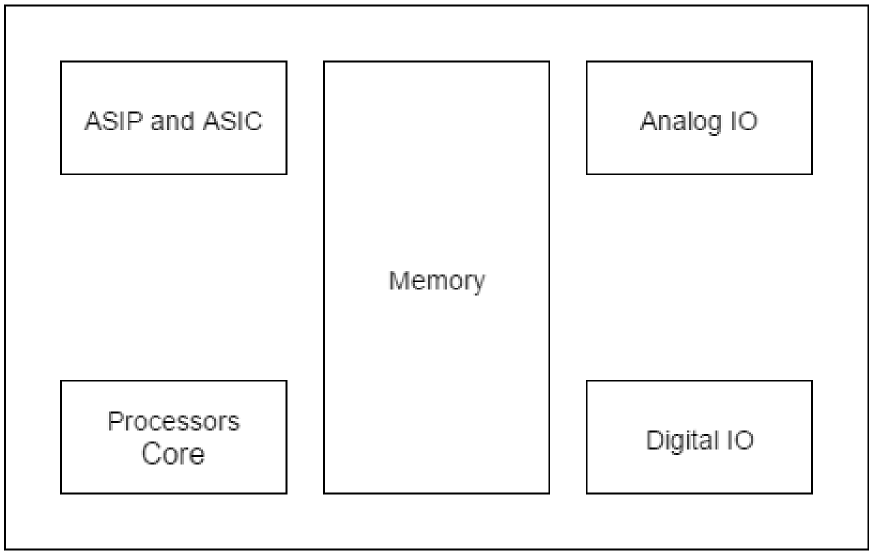
- It has hardware.
- It has application software.
- It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies. RTOS defines the way the system works. It sets the rules during the execution of application program. A small scale embedded system may not have RTOS.

So we can define an embedded system as a Microcontroller based, software driven, reliable, real-time control system.

Characteristics of an Embedded System

- **Single-functioned** – An embedded system usually performs a specialized operation and does the same repeatedly. For example: A pager always functions as a pager.
- **Tightly constrained** – All computing systems have constraints on design metrics, but those on an embedded system can be especially tight. Design metrics is a measure of an implementation's features such as its cost, size, power, and performance. It must be of a size to fit on a single chip, must perform fast enough to process data in real time and consume minimum power to extend battery life.
- **Reactive and Real time** – Many embedded systems must continually react to changes in the system's environment and must compute certain results in real time without any delay. Consider an example of a car cruise controller; it continually monitors and reacts to speed and brake sensors. It must compute acceleration or de-accelerations repeatedly within a limited time; a delayed computation can result in failure to control of the car.

- **Microprocessors based** – It must be microprocessor or microcontroller based.
- **Memory** – It must have a memory, as its software usually embeds in ROM. It does not need any secondary memories in the computer.
- **Connected** – It must have connected peripherals to connect input and output devices.
- **HW-SW systems** – Software is used for more features and flexibility. Hardware is used for performance and security.



Advantages

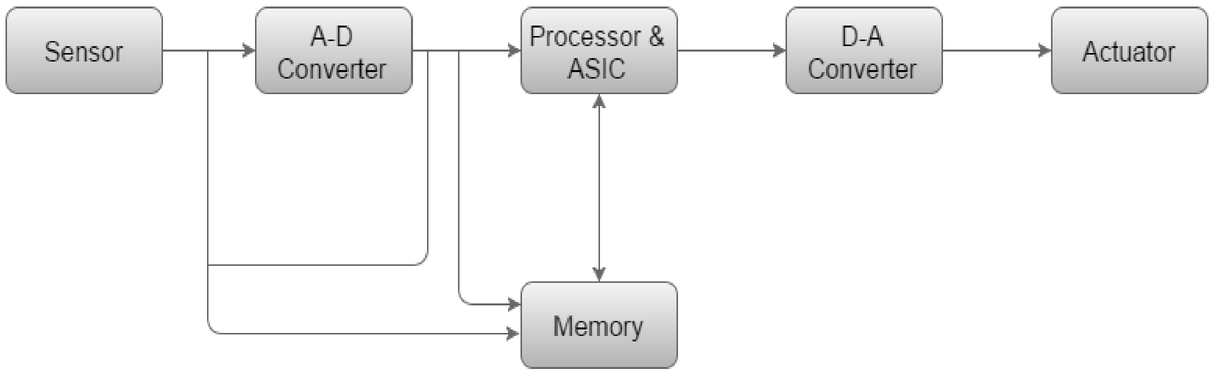
- Easily Customizable
- Low power consumption
- Low cost
- Enhanced performance

Disadvantages

- High development effort
- Larger time to market

Basic Structure of an Embedded System

The following illustration shows the basic structure of an embedded system:



- **Sensor** – It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.

- **A-D Converter** – An analog-to-digital converter converts the analog signal sent by the sensor into a digital signal.

- **Processor & ASICs** – Processors process the data to measure the output and store it to the memory.

- **D-A Converter** – A digital-to-analog converter converts the digital data fed by the processor to analog data.

- **Actuator** – An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

2. ES – Processors

Processor is the heart of an embedded system. It is the basic unit that takes inputs and produces an output after processing the data. For an embedded system designer, it is necessary to have the knowledge of both microprocessors and microcontrollers.

Processors in a System

A processor has two essential units:

- Program Flow Control Unit (CU)
- Execution Unit (EU)

The CU includes a fetch unit for fetching instructions from the memory. The EU has circuits that implement the instructions pertaining to data transfer operation and data conversion from one form to another.

The EU includes the Arithmetic and Logical Unit (ALU) and also the circuits that execute instructions for a program control task such as interrupt, or jump to another set of instructions.

A processor runs the cycles of fetch and executes the instructions in the same sequence as they are fetched from memory.

Types of Processors

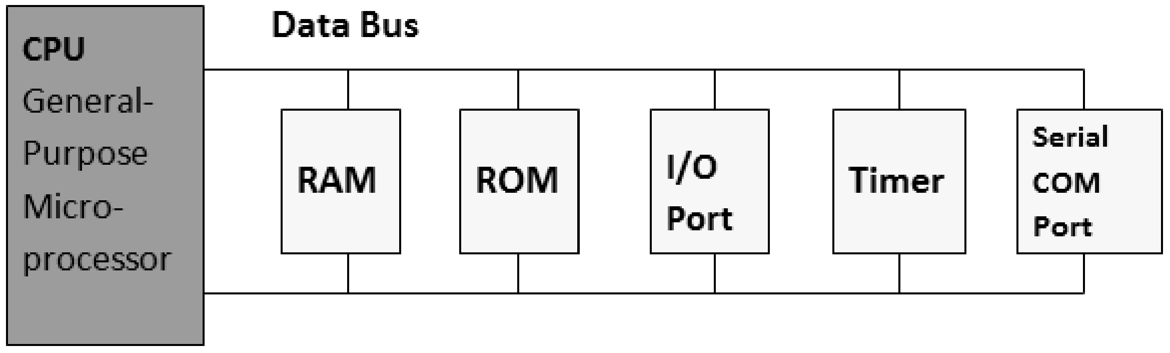
Processors can be of the following categories:

- General Purpose Processor (GPP)
    - Microprocessor
    - Microcontroller
    - Embedded Processor
    - Digital Signal Processor
    - Media Processor
- Application Specific System Processor (ASSP)
- Application Specific Instruction Processors (ASIPs)
- GPP core(s) or ASIP core(s) on either an Application Specific Integrated Circuit (ASIC) or a Very Large Scale Integration (VLSI) circuit

Microprocessor

A microprocessor is a single VLSI chip having a CPU. In addition, it may also have other units such as coaches, floating point processing arithmetic unit, and pipelining units that help in faster processing of instructions.

Earlier generation microprocessors' fetch-and-execute cycle was guided by a clock frequency of order of ~1 MHz. Processors now operate at a clock frequency of 2GHz



A SIMPLE BLOCK DIAGRAM OF A MICROPROCESSOR

Microcontroller

A microcontroller is a single-chip VLSI unit (also called **microcomputer**) which, although having limited computational capabilities, possesses enhanced input/output capability and a number of on-chip functional units.

CPU	RAM	ROM
I/O Port	Timer	Serial COM Port

Microcontroller Chip

Microcontrollers are particularly used in embedded systems for real-time control applications with on-chip program memory and devices.

Microprocessor vs Microcontroller

Let us now take a look at the most notable differences between a microprocessor and a microcontroller.

Microprocessor	Microcontroller
Microprocessors are multitasking in nature. Can perform multiple tasks at a time. For example, on computer we can play music while writing text in text editor.	Single task oriented. For example, a washing machine is designed for washing clothes only.
RAM, ROM, I/O Ports, and Timers can be added externally and can vary in numbers.	RAM, ROM, I/O Ports, and Timers cannot be added externally. These components are to be embedded together on a chip and are fixed in numbers.
Designers can decide the number of memory or I/O ports needed.	Fixed number for memory or I/O makes a microcontroller ideal for a limited but specific task.
External support of external memory and I/O ports makes a microprocessor-based system heavier and costlier.	Microcontrollers are lightweight and cheaper than a microprocessor.
External devices require more space and their power consumption is higher.	A microcontroller-based system consumes less power and takes less space.

3. ES – Architecture

The 8051 microcontrollers work with 8-bit data bus. So they can support external data memory up to 64K and external program memory of 64k at best. Collectively, 8051 microcontrollers can address 128k of external memory.

When data and code lie in different memory blocks, then the architecture is referred as **Harvard architecture**. In case data and code lie in the same memory block, then the architecture is referred as **Von Neumann architecture**.

Von Neumann Architecture

The Von Neumann architecture was first proposed by a computer scientist John von Neumann. In this architecture, one data path or bus exists for both instruction and data. As a result, the CPU does one operation at a time. It either fetches an instruction from memory, or performs read/write operation on data. So an instruction fetch and a data operation cannot occur simultaneously, sharing a common bus.

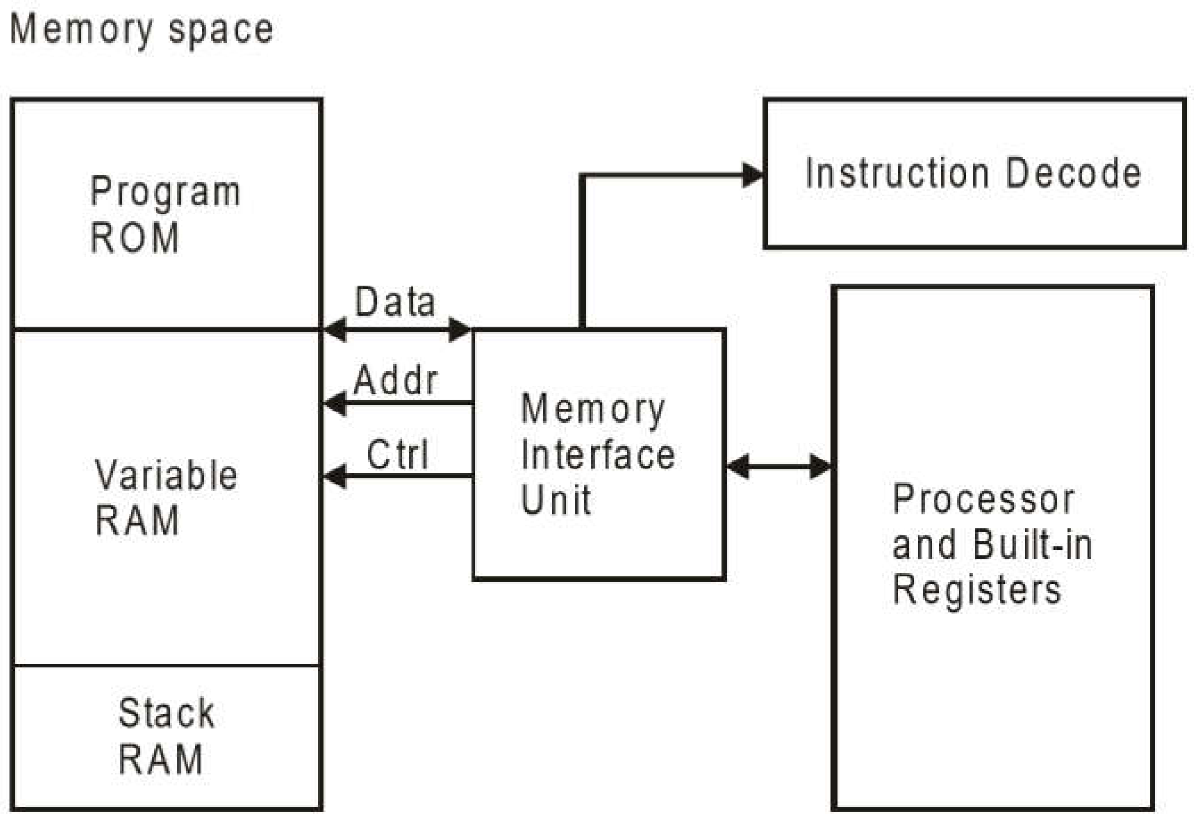


Figure: Von-Neumann Architecture

Von-Neumann architecture supports simple hardware. It allows the use of a single, sequential memory. Today's processing speeds vastly outpace memory access times, and we employ a very fast but small amount of memory (cache) local to the processor.

Harvard Architecture

The Harvard architecture offers separate storage and signal buses for instructions and data. This architecture has data storage entirely contained within the CPU, and there is no access to the instruction storage as data. Computers have separate memory areas for program instructions and data using internal data buses, allowing simultaneous access to both instructions and data.

Programs needed to be loaded by an operator; the processor could not boot itself. In a Harvard architecture, there is no need to make the two memories share properties.

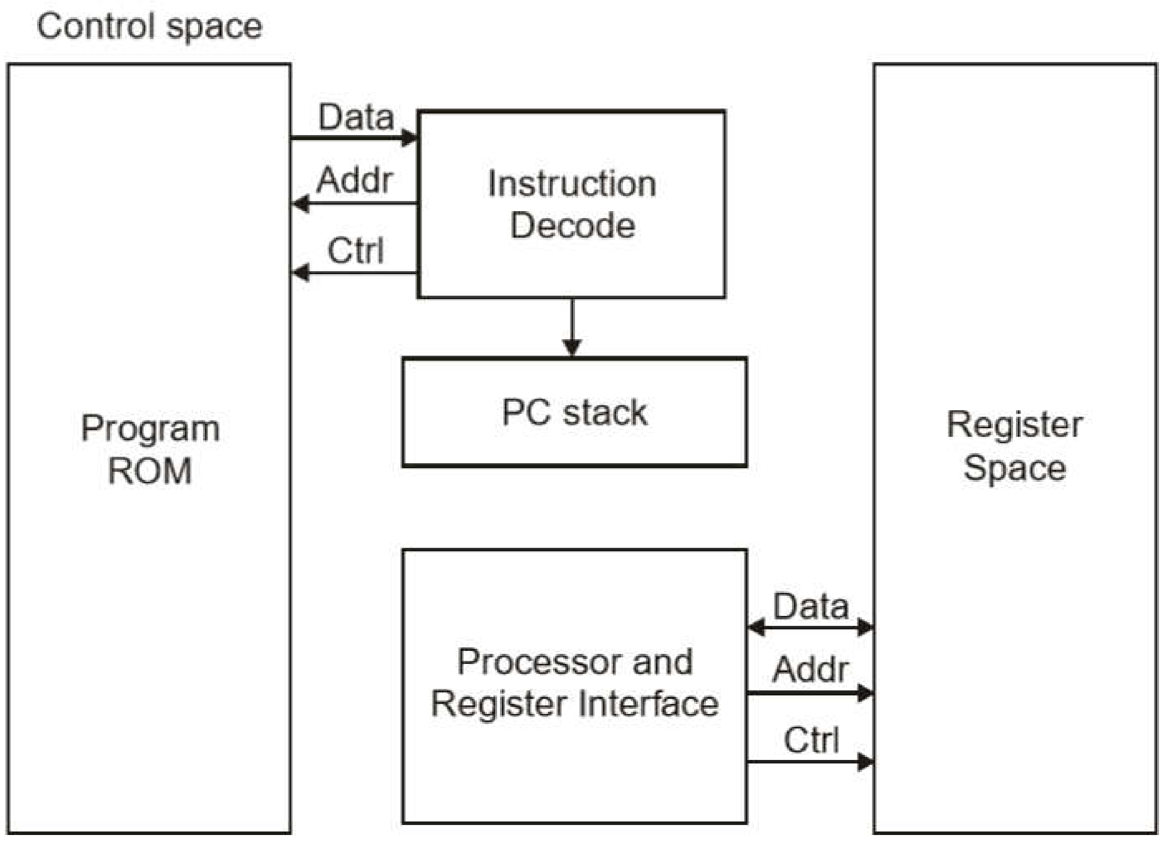


Figure: Harvard Architecture

Von-Neumann Architecture vs Harvard Architecture

The following points distinguish the Von Neumann Architecture from the Harvard Architecture.

Von-Neumann Architecture	Harvard Architecture
Single memory to be shared by both code and data.	Separate memories for code and data.

Processor needs to fetch code in a separate clock cycle and data in another clock cycle. So it requires two clock cycles.	Single clock cycle is sufficient, as separate buses are used to access code and data.
Higher speed, thus less time consuming.	Slower in speed, thus more time-consuming.
Simple in design.	Complex in design.

CISC and RISC

CISC is a Complex Instruction Set Computer. It is a computer that can address a large number of instructions.

In the early 1980s, computer designers recommended that computers should use fewer instructions with simple constructs so that they can be executed much faster within the CPU without having to use memory. Such computers are classified as Reduced Instruction Set Computer or RISC.

CISC vs RISC

The following points differentiate a CISC from a RISC –

CISC	RISC
Larger set of instructions. Easy to program.	Smaller set of Instructions. Difficult to program.
Simpler design of compiler, considering larger set of instructions.	Complex design of compiler.
Many addressing modes causing complex instruction formats.	Few addressing modes, fix instruction format.
Instruction length is variable.	Instruction length varies.
Higher clock cycles per second.	Low clock cycle per second.
Emphasis is on hardware.	Emphasis is on software.
Control unit implements large instruction set using micro-program unit.	Each instruction is to be executed by hardware.
Slower execution, as instructions are to be read from memory and decoded by the decoder unit.	Faster execution, as each instruction is to be executed by hardware.
Pipelining is not possible.	Pipelining of instructions is possible, considering single clock cycle.

4. ES – Tools and Peripherals

Compilers and Assemblers

Compiler

A compiler is a computer program (or a set of programs) that transforms the source code written in a programming language (the source language) into another computer language (normally binary format). The most common reason for conversion is to create an executable program. The name "compiler" is primarily used for programs that translate the source code from a high-level programming language to a low-level language (e.g., assembly language or machine code).

Cross-Compiler

If the compiled program can run on a computer having different CPU or operating system than the computer on which the compiler compiled the program, then that compiler is known as a cross-compiler.

Decompiler

A program that can translate a program from a low-level language to a high-level language is called a decompiler.

Language Converter

A program that translates programs written in different high-level languages is normally called a language translator, source to source translator, or language converter.

A compiler is likely to perform the following operations –

- Preprocessing
- Parsing
- Semantic Analysis (Syntax-directed translation)
- Code generation
- Code optimization

Assemblers

An assembler is a program that takes basic computer instructions (called as assembly language) and converts them into a pattern of bits that the computer's processor can use to perform its basic operations. An assembler creates object code by translating assembly instruction mnemonics into opcodes, resolving symbolic names to memory locations. Assembly language uses a mnemonic to represent each low-level machine operation (opcode).

Debugging Tools in an Embedded System

Debugging is a methodical process to find and reduce the number of bugs in a computer program or a piece of electronic hardware, so that it works as expected. Debugging is difficult when subsystems are tightly coupled, because a small change in one subsystem can create bugs in another. The debugging tools used in embedded systems differ greatly in terms of their development time and debugging features. We will discuss here the following debugging tools:

- Simulators
- Microcontroller starter kits
- Emulator

Simulators

Code is tested for the MCU / system by simulating it on the host computer used for code development. Simulators try to model the behavior of the complete microcontroller in software.

Functions of Simulators

A simulator performs the following functions –

- Defines the processor or processing device family as well as its various versions for the target system.
- Monitors the detailed information of a source code part with labels and symbolic arguments as the execution goes on for each single step.
- Provides the status of RAM and simulated ports of the target system for each single step execution.
- Monitors system response and determines throughput.
- Provides trace of the output of contents of program counter versus the processor registers.
- Provides the detailed meaning of the present command.
- Monitors the detailed information of the simulator commands as these are entered from the keyboard or selected from the menu.
- Supports the conditions (up to 8 or 16 or 32 conditions) and unconditional breakpoints.
- Provides breakpoints and the trace which are together the important testing and debugging tool.
- Facilitates synchronizing the internal peripherals and delays.

Microcontroller Starter Kit

A microcontroller starter kit consists of:

- Hardware board (Evaluation board)
- In-system programmer

- Some software tools like compiler, assembler, linker, etc.
- Sometimes, an IDE and code size limited evaluation version of a compiler.

A big advantage of these kits over simulators is that they work in real-time and thus allow for easy input/output functionality verification. Starter kits, however, are completely sufficient and the cheapest option to develop simple microcontroller projects.

Emulators

An emulator is a hardware kit or a software program or can be both which emulates the functions of one computer system (the guest) in another computer system (the host), different from the first one, so that the emulated behavior closely resembles the behavior of the real system (the guest).

Emulation refers to the ability of a computer program in an electronic device to emulate (imitate) another program or device. Emulation focuses on recreating an original computer environment. Emulators have the ability to maintain a closer connection to the authenticity of the digital object. An emulator helps the user to work on any kind of application or operating system on a platform in a similar way as the software runs as in its original environment.

Peripheral Devices in Embedded Systems

Embedded systems communicate with the outside world via their peripherals, such as following:

- Serial Communication Interfaces (SCI) like RS-232, RS-422, RS-485, etc.
- Synchronous Serial Communication Interface like I2C, SPI, SSC, and ESSI
- Universal Serial Bus (USB)
- Multi Media Cards (SD Cards, Compact Flash, etc.)
- Networks like Ethernet, LonWorks, etc.
- Fieldbuses like CAN-Bus, LIN-Bus, PROFIBUS, etc.
- Timers like PLL(s), Capture/Compare and Time Processing Units.
- Discrete IO aka General Purpose Input/Output (GPIO)
- Analog to Digital/Digital to Analog (ADC/DAC)
- Debugging like JTAG, ISP, ICSP, BDM Port, BITP, and DP9 ports

Criteria for Choosing Microcontroller

While choosing a microcontroller, make sure it meets the task at hand and that it is cost effective. We must see whether an 8-bit, 16-bit or 32-bit microcontroller can best handle the computing needs of a task. In addition, the following points should be kept in mind while choosing a microcontroller –

- **Speed** – What is the highest speed the microcontroller can support?
- **Packaging** – Is it 40-pin DIP (Dual-inline-package) or QFP (Quad flat package)? This is important in terms of space, assembling, and prototyping the end-product.

- **Power Consumption** – This is an important criteria for battery-powered products.
- **Amount of RAM and ROM** on the chip.
- **Count of I/O pins and Timers** on the chip.
- **Cost per Unit** – This is important in terms of final cost of the product in which the microcontroller is to be used.

Further, make sure you have tools such as compilers, debuggers, and assemblers, available with the microcontroller. The most important of all, you should purchase a microcontroller from a reliable source.