Professional Elective-II: Embedded Systems (PECCSE601B)

Module - I

Introduction to embedded systems

Prepared by – Dr. Susmita Biswas
Associate Professor
University of Engineering and Management, Kolkata

What do you mean by a system?

System: It is an arrangement, in which all its units assemble and work together according to a fixed plan/ program/ set of instruction.

Do you know basic architecture of a computing system?

Well it is quite similar to the human brain.

First consider the operation of a human brain.

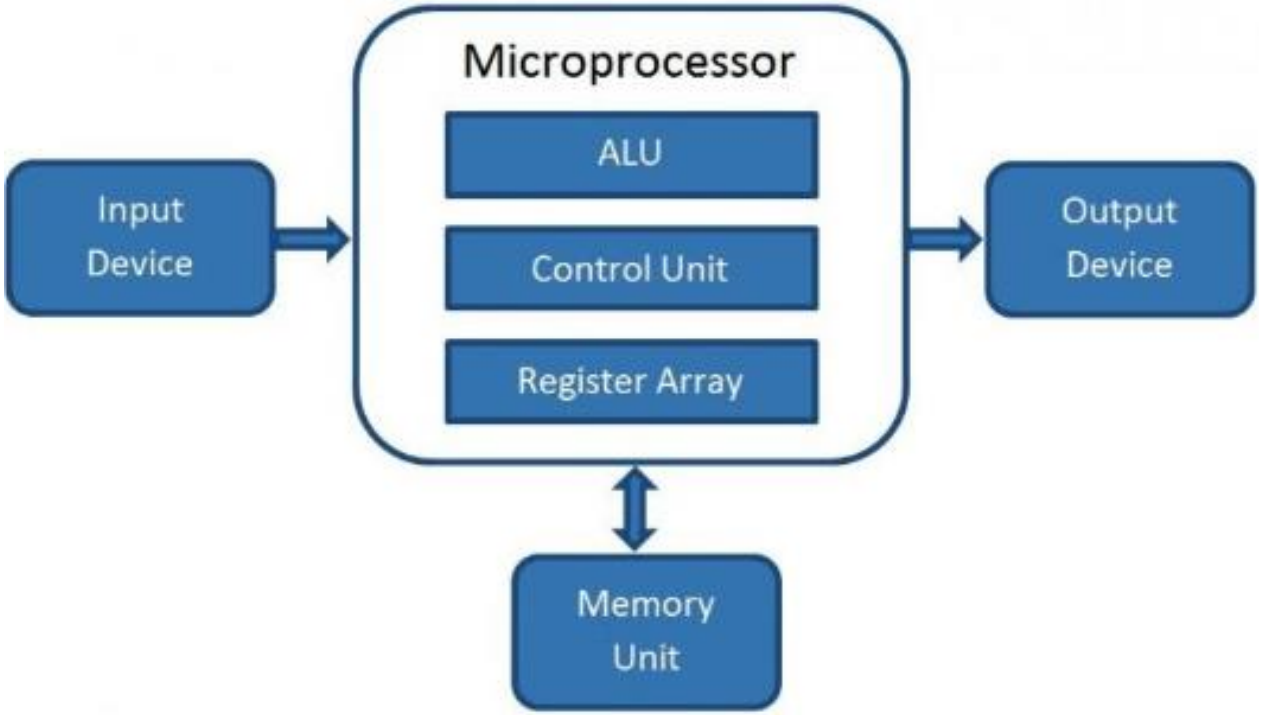
First of all, brain takes some inputs through different sensing organs (input devices) in form of some visible portion of electromagnetic wave, sound wave, smell, taste and touch.

Then those input information are superimposed on some electrical carrier signal, that carry information to a particular portion of our brain through neuron.

Those information then either saved in our memory or processed by certain portion of our brain.

Whatever be the output, it is either saved in our memory for future purpose or transferred to different organs of our body (output devices) to act according to the instruction of the brain.

Architecture of a computing system

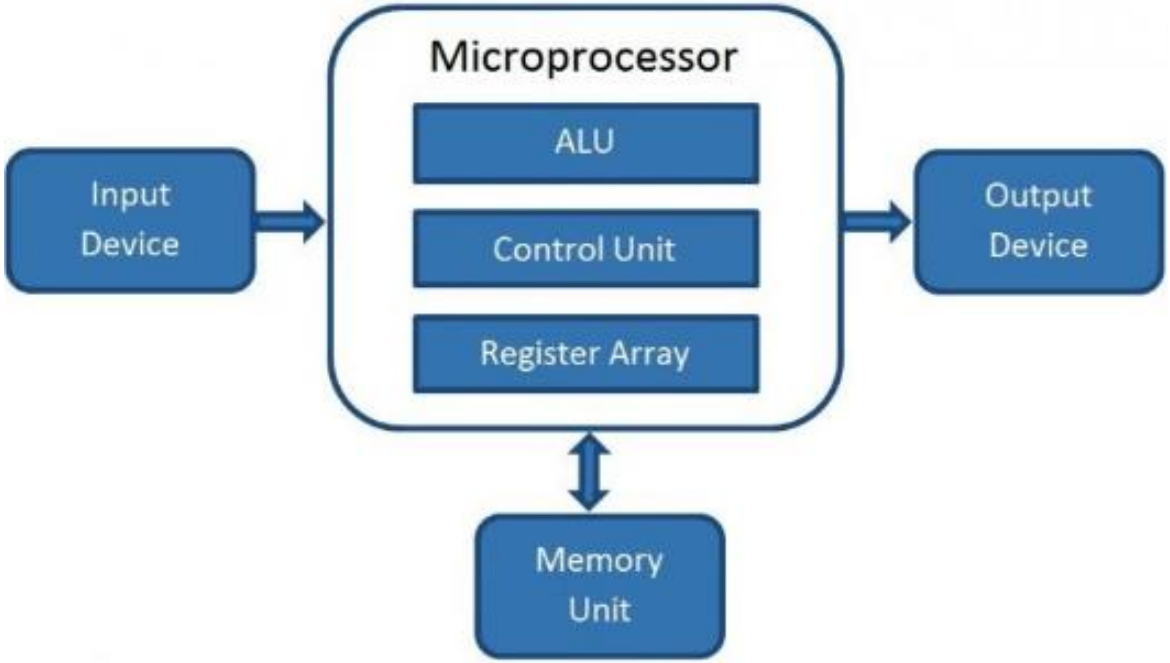


It consists of a Central Processing Unit (CPU), which again consist of the arithmetic logic unit (ALU), the control unit (CU) and the registers.

CPU takes inputs from some input devices/peripherals (such as Keyboard, Mouse, Touchscreen, Touchpad, Scanner, Microphone).

Input information (instruction and data) are either stored in memory for future purpose or executed by ALU.

Architecture of a computing system



Results of execution are either stored in different register array and memory for future purpose or are transferred to output devices/peripherals (such as Monitor, Printing machine, Speakers, Digital projector).

The data transfer process inside a computer system is basically performed through different bus: unidirectional address bus and bidirectional data bus.

The control unit controls all the operations and it use control bus for transferring control signal.

What is an embedded system?

What is the significance of the term ‘Embedded’?

What is the significance of the term ‘Embedded’?

Types of computers we are familiar with:

Desktops: machines in which we learn computing.

Laptops: today laptops have become very much affordable and most of us own our personal laptops.

But there is another type of computing system that is often hidden from us. Hidden means we cannot see those computing system.

We know desktop can compute, laptop can compute.

But if you look at your air conditioning machine, does it look like a computing machine?

The answer is no!

What is the significance of the term ‘Embedded’?

But inside those machines there is some computing brain hidden. They are hidden in their environment for which they are created. Such systems are traditionally referred to as embedded systems.

These are computing systems but they are embedded inside the environment (the surroundings or actually the scenario for which the system was designed).

For example: the computing system, which is embedded in an air conditioning machine, for that computing system – the air conditioning machine is the environment.

It does not interact with anybody outside the air conditioning machine and it is responsible for controlling the air conditioning machine.

What is an embedded system?

Computers, which are embedded within other systems.

Embedded system

It is basically an electronic/ electromechanical system, which consist of specialized hardware and software and performs a specific task.

Example: Mobile handset, Microwave oven, Washing machine, Camera, Air conditioner, Automobiles, Aircraft, Missiles

Common Features of Embedded system

1. They are special-purpose or single-functioned.

- Like an air conditioning machine, there is an processor inside, the sole purpose of the processor is to ensure that the air conditioning machine is working properly nothing else. Hence, it is very special purpose. Typically it executes a single program related to the application for which it was built.

2. Tight constraints on cost, energy, memory and form factor.

- Processors, utilized in normal household devices, should have low cost, low power, limited memory (for accumulating single program for specific application), small size and relatively fast response time.
- However, for bigger applications such as Automobiles, Aircraft, Missiles those constraints may be relaxed. We can use costly processors there.

3. They must react to events in real-time.

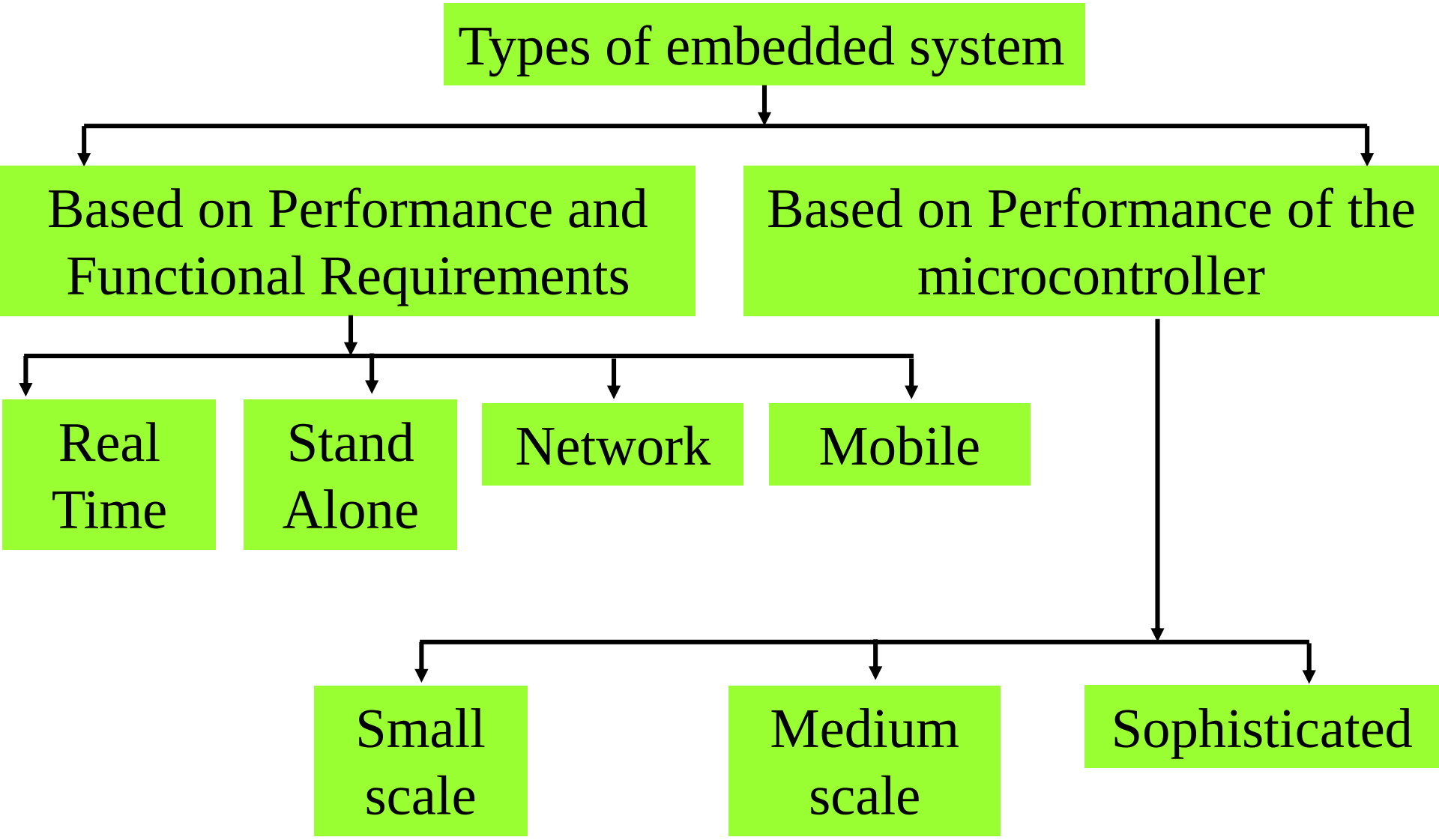
- Respond to inputs from the system's environment without delay.
- "Without delay" is a subjective term. The tolerable delay varies from application to application.

Classification of embedded systems

Embedded systems can be classified on the basis of two constraints:

1. Based on Performance and Functional Requirements
2. Based on Performance of the microcontroller

Classification of embedded systems



Classification based on Performance and Functional Requirements

Real Time embedded systems

- A Real-Time Embedded System is strictly time specific which means these embedded systems **provides output in a particular/defined time interval.** These type of embedded systems provide quick response in critical situations which gives most priority to time based task performance and generation of output.

- Utilized in defense sector, health care sector and industrial applications

- The real time Embedded systems are divided into two parts:

Real Time embedded systems

1. Hard real time embedded system

- In these types of embedded systems time/deadline of task is strictly followed. Task must be completed in between time frame (defined time interval) otherwise result/output may not be accepted.
- Example: Temperature controlling system in a chamber, Traffic control system

Suppose, temperature of a chamber should be maintained at 25℃. If the temperature just crosses the 25℃ limit, the system should open a vault within say 10 millisecond otherwise accident will occur. That will cause property damage and even life loss. There is a hard time constraints. This is the example of a hard real time embedded system.

Real Time embedded systems

2. Soft real time embedded system

- In these types of embedded systems time/deadline is not so strictly followed. If deadline of the task is passed (means the system didn't give result in the defined time) still result or output is accepted.
- Example: Microwave oven

Here, we are giving some input by pressing buttons, Those mechanical input then converted to some electrical signal. The embedded processor should process the input signal and produce output within few second. However, if the system fails to produce output within this time limit, then we can say the system is not that much efficient. But there is no chance of property damage and life loss in this case. Here delayed output is accepted.

Stand Alone embedded systems

- Stand Alone Embedded Systems are independent systems which can work by themselves they don't depend on a host system. It takes input in digital or analog form, process the input signal and provides the output.
- Less complex
- Example:

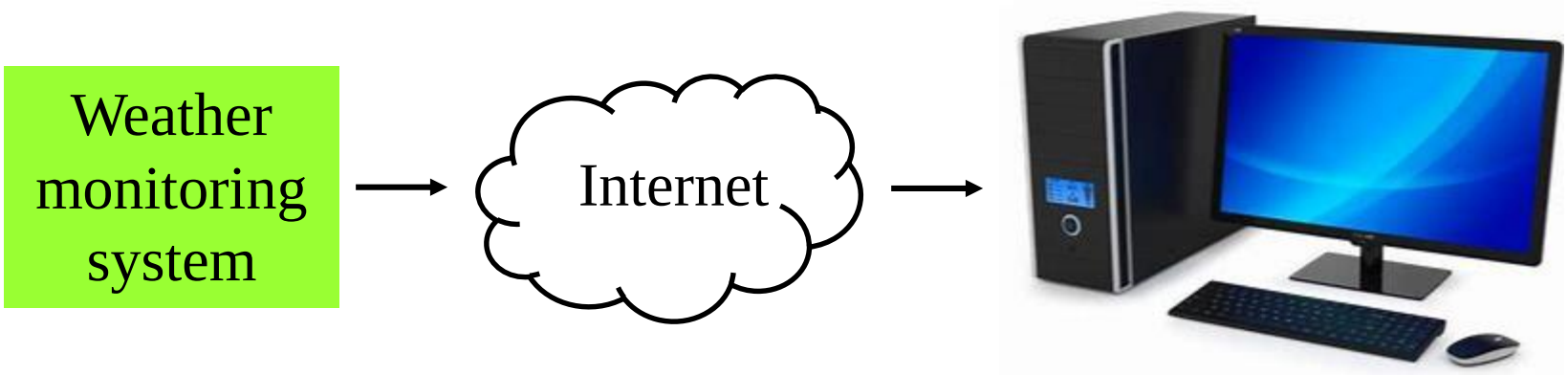
Doorbell, Calculator, Mp3 player

For calculator, we are giving some input by pressing buttons, Those mechanical input then converted to some electrical signal. The embedded processor then process the input signal and produce output which is displayed in the LCD screen.

Network embedded systems

- Networked Embedded Systems are connected to a network which may be wired or wireless to provide output to the attached device. They communicate with the server or with individual node using network.
- Communication may established by using LAN, WAN or other protocols.
- Example:

ATM machine, Card swipe machine, IOT Devices, Weather monitoring system.



Mobile embedded systems

- Mobile embedded systems are small and easy to use and requires less resources. They are the most preferred embedded systems. In portability point of view mobile embedded systems are also best.
- Example:

Mobile phones, Digital camera

Classification based on Performance of the microcontroller

Small scale embedded systems

- 8051 family, Hitachi H8, PIC 16F8X family
- Low cost, low performance
- May/may not contain operating system
- A single 8 or 16 bit microcontroller
- Little hardware and software complexities
- Involve board level design
- Battery operated /direct power supply
- C language is used for writing programs
- Programming tools:

Editor, assembler and cross assembler

- Example:

Digital watches, Automatic door lock

Medium scale embedded systems

- 8051MX, PIC 16F876
- Medium performance
- A single or few 16 or 32 bit microcontroller or DSPs or RISCs
- Hardware and software both are complex
- Faster than the small scale embedded systems
- Programming tools:

C, C++, Java, RTOS, Simulator, Debugger and IDE

- Example:

Routers for networking, cameras, Smartphones

Sophisticated embedded systems

- ARM family, Cortex M15
- High performance
- More than 32 or 64 bit microcontroller
- Hardware and software complexity is very large
- Perform large scale complex function
- Programming tools:

May not readily available at reasonable cost

- Example:

Washing machine, Security products

Components of embedded systems

- Hardware
- Application software
- Real-time operating systems (RTOS)

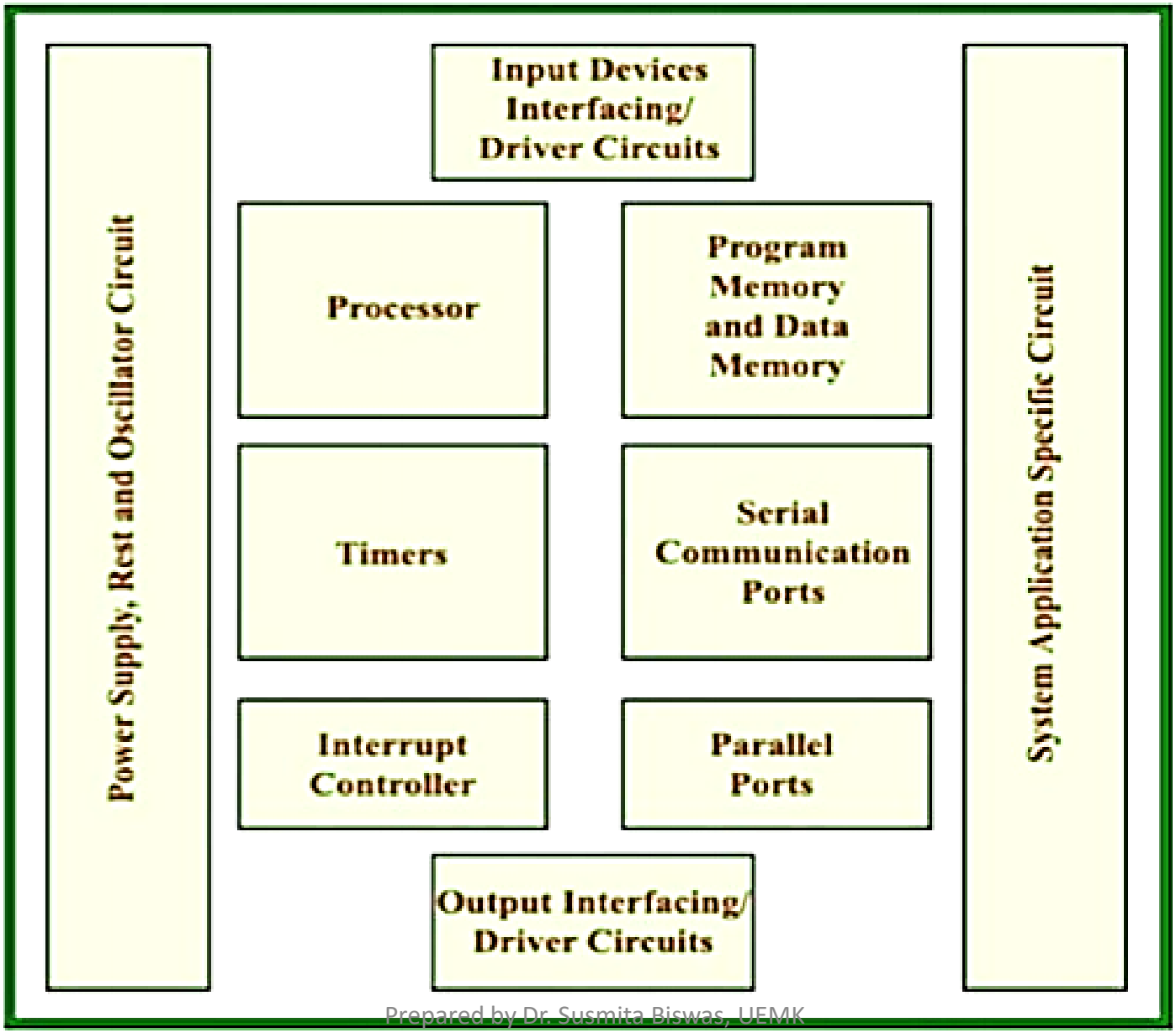
Hardware and software components both make up an embedded system. It also goes by the name “firmware” when it performs certain tasks.

Embedded hardware units

- **Hardware units:**

1. **Embedded processor**
2. **Power supply**
3. **Reset**
4. **Oscillator circuit**
5. **Timers**
6. **Program & data memory**
7. **Interrupt controller**
8. **I/O ports**
9. **Input& output device interfacing/driver circuits**
10. **System Application specific circuits**

Embedded hardware units



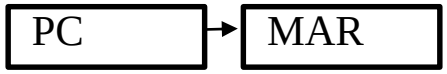
1. Embedded processor:

- It is the heart of the embedded system.
- It takes response from sensors of input devices in digital form and processes these input responses to produce output in real time processing environment.
- For example, microprocessor, microcontroller. When the processor, memory, input/output devices are fabricated in a single semiconductor chip – it is called the microcontroller.
- It has two essential units : control unit and execution unit.
- **Control unit** perform the program flow control operation inside an embedded system. It fetches the set of instructions which are stored inside a memory. That's why this is also called as the fetching unit.

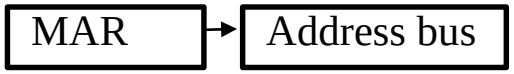
- **execution unit** executes various tasks inside a processor. It includes ALU and circuits to perform execution of the instructions for a program control task.

The execution of a particular instruction can be performed through different micro-operations.

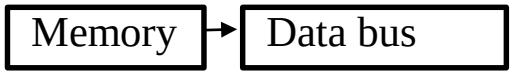
- First we need to fetch the instruction from memory. PC holds the memory address of this particular instruction when previous instruction is being executed. At the end of this execution, the memory address will be transferred to the MAR.



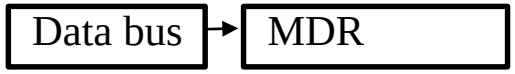
- The address will be then copied to the address bus from MAR and will activate the particular memory location through memory decoder.



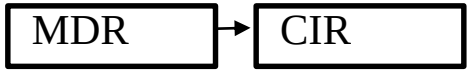
- The control unit (CU) of CPU will send the read signal to memory. Binary instruction in that particular memory location will be copied to the data bus.



- The instruction will be placed in the MDR from data bus.



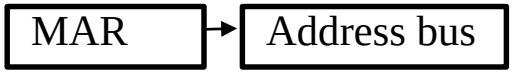
- Instruction will be transferred from MDR to CIR for decoding and execution.



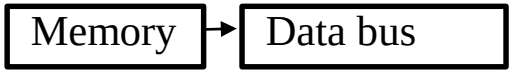
- If the instruction isn't to be immediately executed, then it will be placed in the IBR.

- The CPU decodes the instruction and places the address of required memory location in the MAR for data fetching.

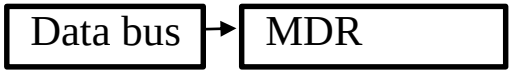
- The address will be then copied to the address bus from MAR and will activate the particular memory location through memory decoder.



- The control unit (CU) of CPU will send the read signal to memory. Binary data in that particular memory location will be copied to the data bus.

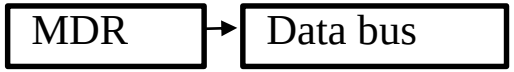


- The data will be placed in the MDR from data bus.

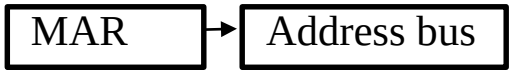


- Data from MDR will be transferred from MDR to ACC or GPR for execution. The result will be placed on ACC. The data will be transferred to MDR.

- The data will be copied from MDR to the data bus.



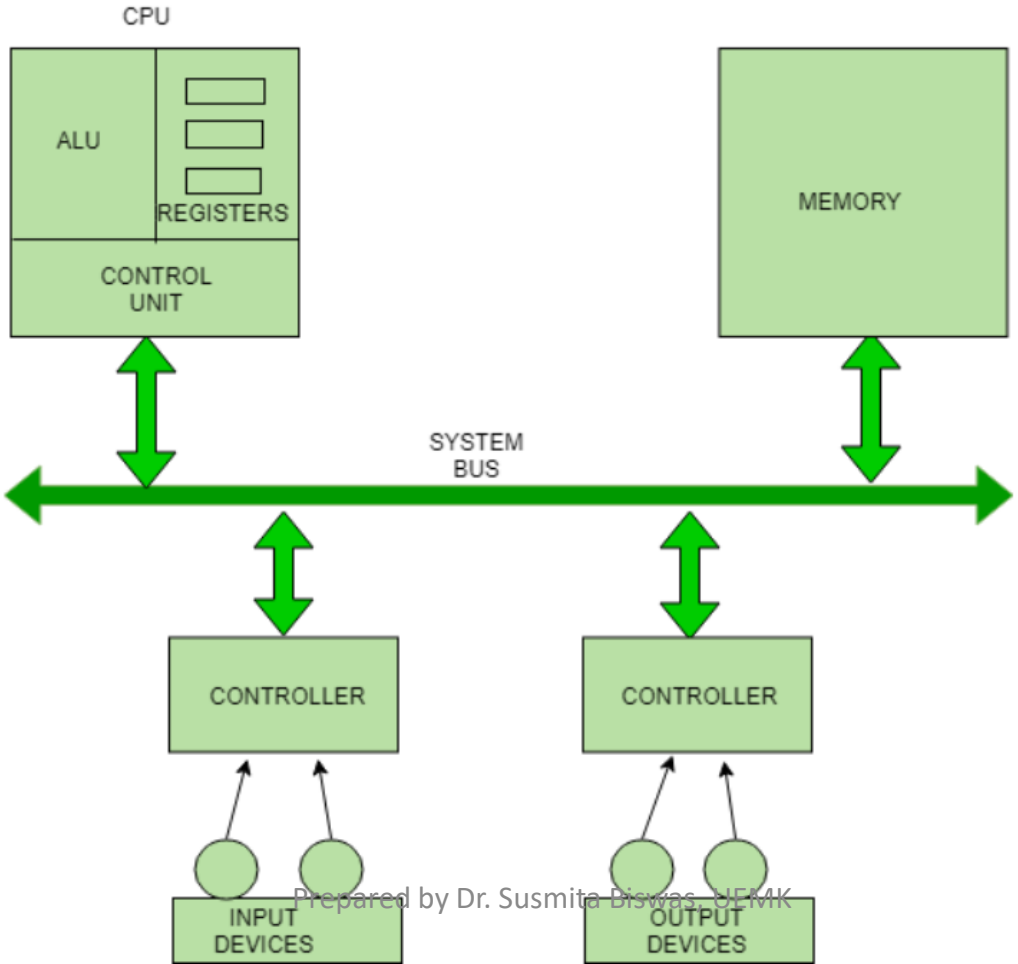
- Similar to a read operation, the address of the needed memory location to write the data will be placed in the MAR by CPU. The address will be then copied to the address bus from MAR and will activate the particular memory location through memory decoder.



- The control unit (CU) of CPU will send the write signal to memory. Binary data from data bus will be copied to that particular memory location.

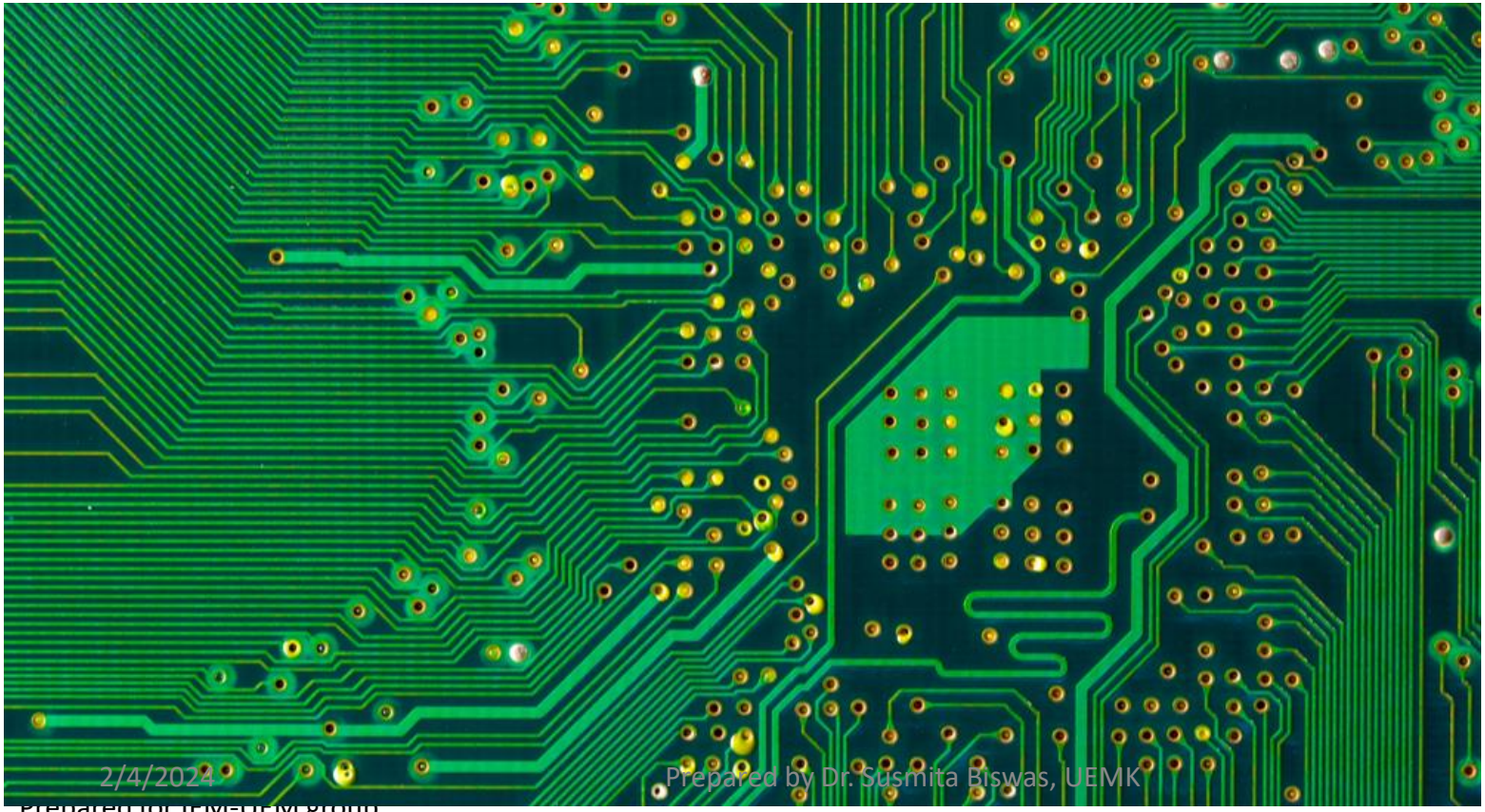
Interconnection between Functional Components

❑ **System Bus:** A bus is a transmission path, made of a set of conducting wires over which data or information in the form of electric signals, is passed from one component to another in a computer. The bus can be of three types – **Address bus, Data bus** and **Control Bus.**

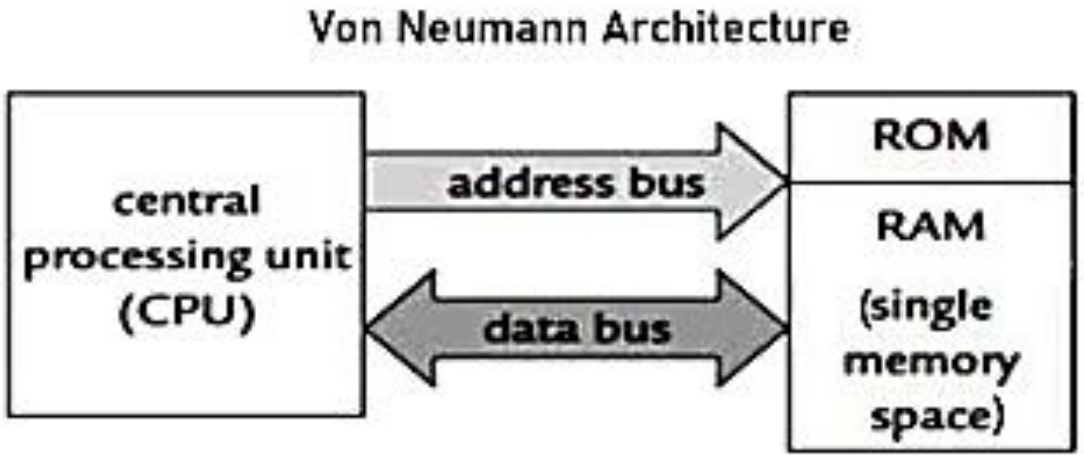


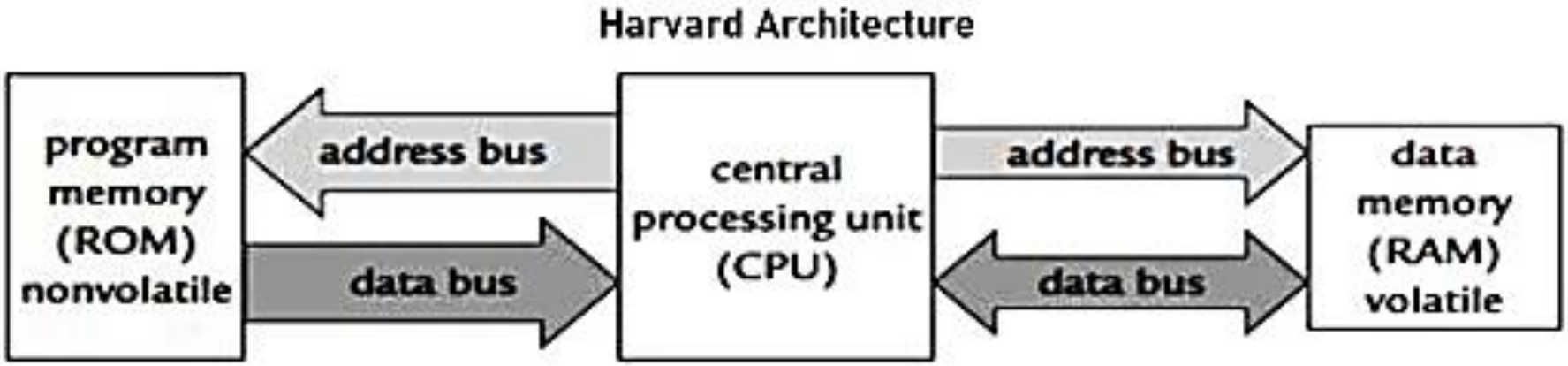
Interconnection between Functional Components

❑ The **address bus carries the address location of the data or instruction**. The **data bus carries data from one component to another** and the **control bus carries the control signals**. The system bus is the common communication path that carries signals to/from CPU, main memory and input/output devices. The input/output devices communicate with the system bus through the controller circuit which helps in managing various input/output devices attached to the computer.



Von Neumann & Harvard Architecture





2. Power supply: Most of the systems have their own power supply. Some embedded systems do not have their own power supply. These embedded systems are powered by external power supply.

3. Reset: Reset means that processor begins processing of instructions from starting address set by default in program counter on power up.

4. Oscillator circuit: It generates clock pulse to synchronize work of all parts of the CPU. The clock circuit controls execution time of instructions, CPU machine cycles.
For example, on chip crystal oscillator which produces clock pulse of frequency 12 MHz.

5. **Timers:** Timer circuit is suitably configured as system clock or RTC (Real time clock). To schedule various tasks and for real time programming an RTC (Real Time Clock), or system clock is needed.

6. **Program & data memory:** In embedded system, secondary memory like disk is avoided. Most of the embedded processors have internal memory such as ROM, RAM, PROM, EPROM, flash/EEPROM for storing program and data.

7. Interrupt controller: It is an interrupt handling mechanism which must exist in embedded system to handle interrupts from various processes and for handling multiple interrupts simultaneously pending for service.

8. **I/O ports:** I/O ports are used to interface external devices like sensors, key buttons, transducers, LEDs, LCD actuators, alarms, motors, values, printer etc. There are two types of ports, parallel and serial port. The parallel ports are used in short distance communication while serial ports are used in long distance communication.

9. Input& output device interfacing/driver circuits: Some I/O devices like motors, actuators, valves, sensors are not compatible with the processor. Hence the I/O interface circuits are designed to drive such input and output devices interfaced to the embedded processor.

10. System Application specific circuits: These are the circuits that can control specific target circuits. They consist of ADC, DAC, relays, sensors etc.

Embedded software

Embedded software

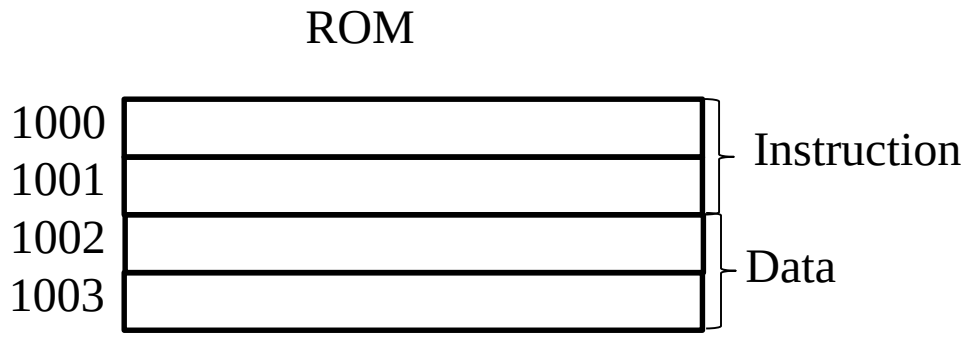
Embedded software is **a piece of software that is embedded in hardware or non-PC devices**. It is written specifically for the particular hardware that it runs on and usually has processing and memory constraints because of the device's limited computing capabilities.

Software components and important terms / tools:

- Hardware is the heart of embedded system and **software is the brain of embedded system**. Software is controlling and coordination every operations in an embedded system.
- Embedded system has a **application specific software**. The system specific instruction code and data at the final phase are written on to ROM (Read-only memory) or flash memory.

Embedded software

- The final phase of data and code based on which the embedded is actually working is burned into ROM and is called **ROM Image.**
- ROM image also consist of a table which contains address of every byte written in the ROM image.



Coding

- Coding or instructions can be written in

1. Machine Language
2. Assembly Language
3. High Level Language

Machine Language

- Machine language is a low-level programming language that is understood by computers. Machine language is also known as machine codes or object code. The CPU processes this machine code as input.
- Machine language is made up of binary bits 0 and 1. Machine language is basically binary language: 01100110 00001010.
- As machine language consists of only 0 and 1, that's why it is difficult to understand in raw form. Machine language cannot understood by humans.
- Machine languages are platform dependent.
- Machine Language Instruction Components: Operand (data that the operation must be performed on) and Opcode (operation that the processor must perform).

Assembly Language

- Assembly language is a low-level language that helps to communicate directly with computer hardware.
- It uses mnemonics to represent the operations that a processor has to do. (A mnemonic is a name that groups different opcode that have the same purpose. Example: ADD B (it adds the content of accumulator to the content of the register B and result will be stored in the accumulator). Here, ADD B is an opcode, B is operand, ADD is the mnemonic)
- It is an intermediate language between high-level languages like C++ and the binary language.
- It uses hexadecimal and binary values, and it is readable by humans.
- This language is hardware specific.
- In order to burn the code/instruction in the hardware (memory) we need to convert it into the machine code or binary code. For that purpose we need an assembler. Some memory addresses are allocated to generated machine codes.

Assembly Language

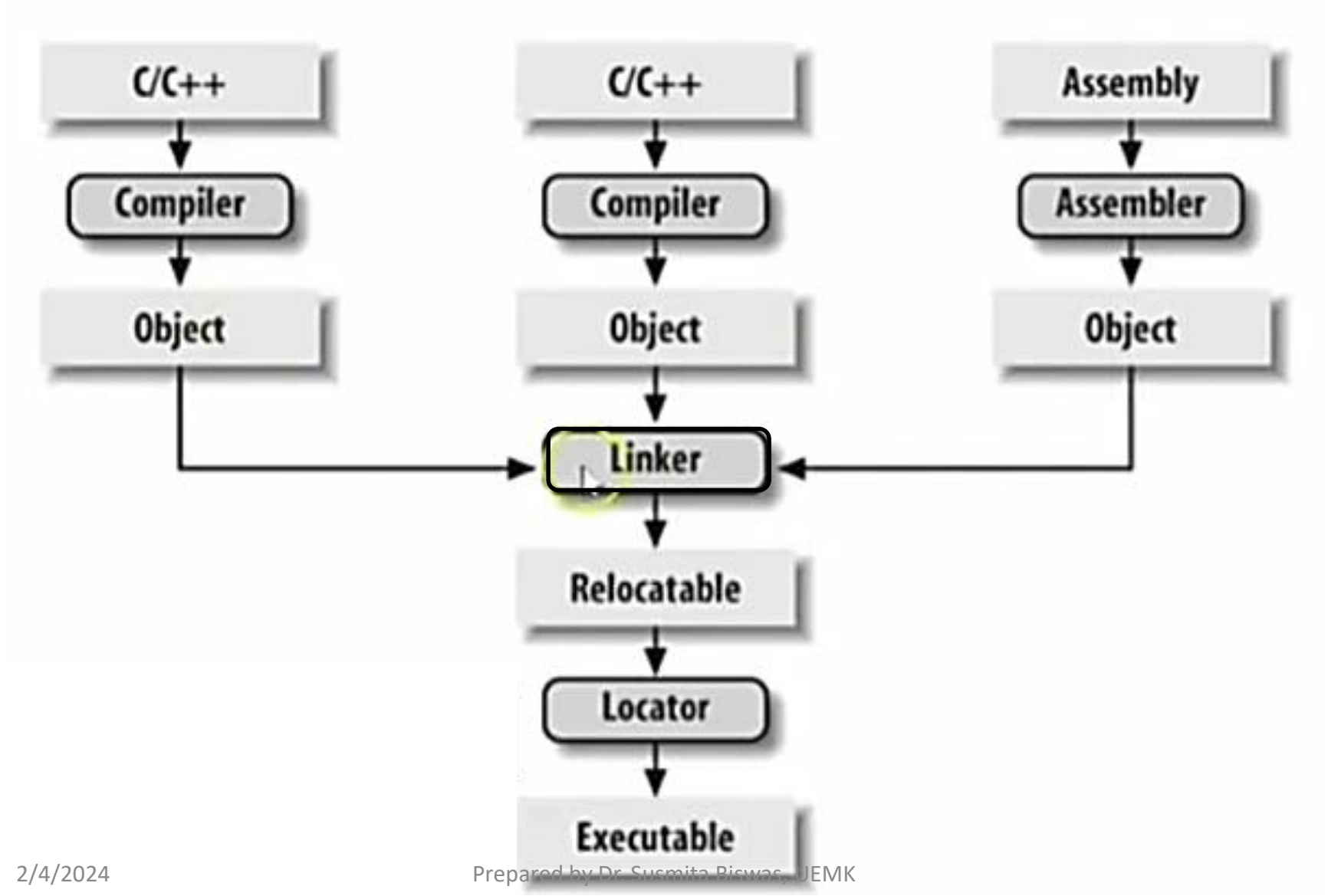
- In assembly codes there are some **library functions** (which is predefined). When processor decode one particular opcode, it will call specific library function and read the definition. Here, we need a linker. Linker will actually fetch the definition of specific library function. The definition is also a binary of machine code. The binary definition will be linked to the machine code of actual program. The machine codes are ready now to be written in the memory.
- Linker also accumulate different object codes and send the combined code to locator.
- Then machine codes are brunt in the ROM. Data are also written in the specific address location of the memory to form ROM image.

High Level Language

- High level languages are programming languages which are used for writing programs or software which could be understood by the humans and computer. High level languages are easier to understand for humans because it uses lot of symbols letters phrases to represent logic and instructions in a program. It contains high level of abstraction compared to low level languages. Example: Python, Java, C++.

- A Compiler is a software that typically takes a high level language (Like C++ and Java) code as input and converts the input to a lower level language at once.

High Level Language

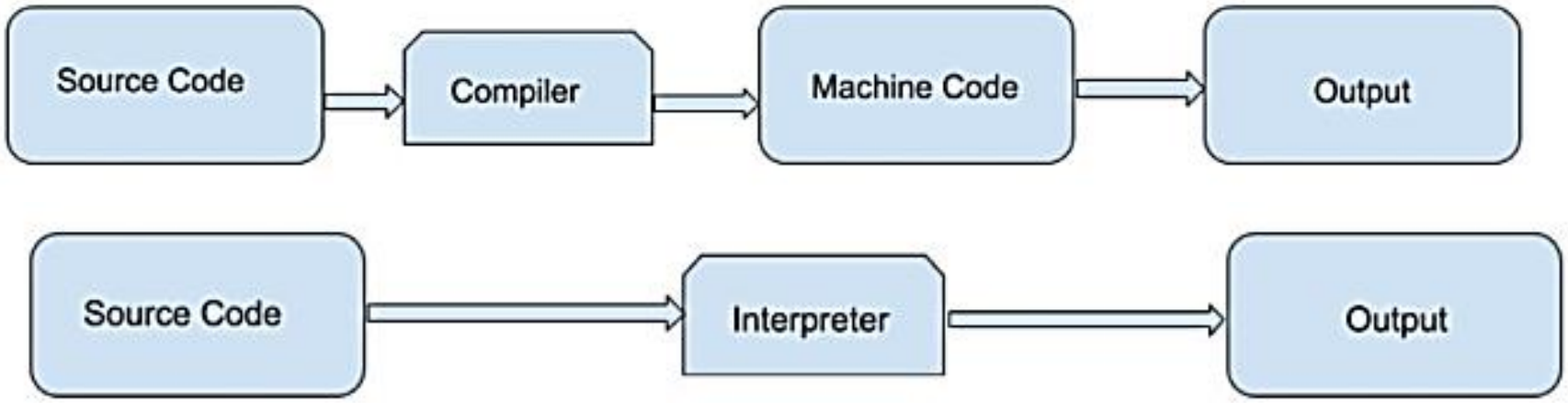


Software development tools

- **Assembler:** Converts assembly language codes to machine language code (assembling).
- **Compiler:** Converts high level codes to machine language.
- **Linker:** Links definition of specific library function to object code and assemble different object codes and also it will convert it to the executable format.
- **Locator:** Allocates the space in the memory where object code will be loaded for execution. It also keeps track of the available address at ROM. It will then produce an output file containing a binary memory image / ROM image that can be loaded into the target ROM.
- **Editor:** It allows programmer to write and edit codes. Suppose when you write a program in MATLAB editor window, you can make some corrections in the written code.

Software development tools

- **Interpreter:** An interpreter is a program that directly executes the instructions in a high-level language, without converting it into machine code.
- In programming, we can execute a program in two ways.
- Firstly, through compilation and secondly, through an interpreter. The common way is to use a compiler.



Characteristics of embedded systems

Characteristics of embedded systems

Some characteristics of an embedded system that make it different from a general purpose computer are:

- **Application and Domain specific:** An embedded system is designed for a specific purpose only. It will not do any other task. Example. A washing machine can only wash, it cannot cook.
- **Reactive and Real time:** Certain Embedded systems are designed to react to the events that occur in the nearby environment. These events also occur real-time. Example. An air conditioner adjusts its mechanical parts as soon as it gets a signal from its sensors to increase or decrease the temperature when the user operates it using a remote control.
- **Operation in harsh environment:** Certain embedded systems are designed to operate in harsh environments like very high temperature of the deserts or very low temperature of the mountains or extreme rains. These embedded systems have to be capable of sustaining the environmental conditions it is designed to operate in.

Characteristics of embedded systems

- **Distributed:** Certain embedded systems are part of a larger system and thus form components of a distributed system. These components are independent of each other but have to work together for the larger system to function properly. Example. A car has many embedded systems controlled to its dash board. Each one is an independent embedded system yet the entire car can be said to function properly only if all the systems work together.
- **Small size and weight:** An embedded system that is compact in size and has light weight will be desirable or more popular than one that is bulky and heavy. Example. Currently available cell phones. The cell phones that have the maximum features are popular but also their size and weight is an important characteristic.
- **Power concerns:** It is desirable that the power utilization and heat dissipation of any embedded system be low. If more power is required then a battery of higher power or more batteries need to be accommodated in the embedded system.

Quality attributes of embedded systems

Quality attributes of embedded systems

There are the some attributes that together form the deciding factor about the quality of an embedded system. There are two types of quality attributes: Operational Quality Attributes and Non-Operational Quality Attributes.

Operational Quality Attributes: These are attributes related to operation or functioning of an embedded system. The way an embedded system operates affects its overall quality.

- **Response:** Response is a measure of quickness of the system. It gives you an idea about how fast your system is tracking the input variables. Most of the embedded system demand fast response which should be real-time.
- **Throughput:** Throughput deals with the efficiency of system. It can be defined as rate of production over a stated period of time.
- **Reliability:** Reliability is a measure of how much percentage you rely upon the proper functioning of the system. Mean time between failures and mean time to repair are terms used in defining system reliability. Mean time between failures can be defined as the average time the system is functioning before a failure occurs. Mean time to repair can be defined as the average time the system has spent in repairs.

Quality attributes of embedded systems

- **Maintainability:** Maintainability deals with support and maintenance to the end user or a client in case of technical issues and product failures or on the basis of a routine system checkup. It can be classified into two types: Scheduled or Periodic Maintenance and Maintenance to unexpected failure.
- **Security:** Confidentiality, Integrity and Availability are three corner stones of information security. Confidentiality deals with protection data from unauthorized disclosure. Integrity gives protection from unauthorized modification. Availability gives protection from unauthorized user. Certain Embedded systems have to make sure they conform to the security measures. Example. An Electronic Safety Deposit Locker can be used only with a pin number like a password.
- **Safety:** Safety deals with the possible damage that can happen to the operating person and environment due to the breakdown of an embedded system or due to the emission of hazardous materials from the embedded products. A safety analysis is a must in product engineering to evaluate the anticipated damage and determine the best course of action to bring down the consequence of damages to an acceptable level.

Quality attributes of embedded systems

Non-Operational Quality Attributes: These are attributes not related to operation or functioning of an embedded system. The way an embedded system operates affects its overall quality. These are the attributes that are associated with the embedded system before it can be put in operation.

- **Testability and Debug-ability:** It deals with how easily one can test his/her design, application and by which mean he/she can test it. In hardware testing the peripherals and total hardware function in designed manner, Firmware testing is functioning in expected way, Debug-ability is means of debugging the product as such for figuring out the probable sources that create unexpected behavior in the total system.
- **Evolvability:** For embedded system, the qualitative attribute "Evolvability" refer to ease with which the embedded product can be modified to take advantage of new firmware or hardware technology.

Quality attributes of embedded systems

- **Portability:** Portability is measured of "system Independence". An embedded product can be called portable if it is capable of performing its operation as it is intended to do in various environments irrespective of different processor and or controller and embedded operating systems.
- **Time to prototype and market:** Time to Market is the time elapsed between the conceptualization of a product and time at which the product is ready for selling or use. Product prototyping help in reducing time to market.
- **Per unit and total cost:** Cost is an important factor which needs to be carefully monitored. Proper market study and cost benefit analysis should be carried out before taking decision on the per unit cost of the embedded product.