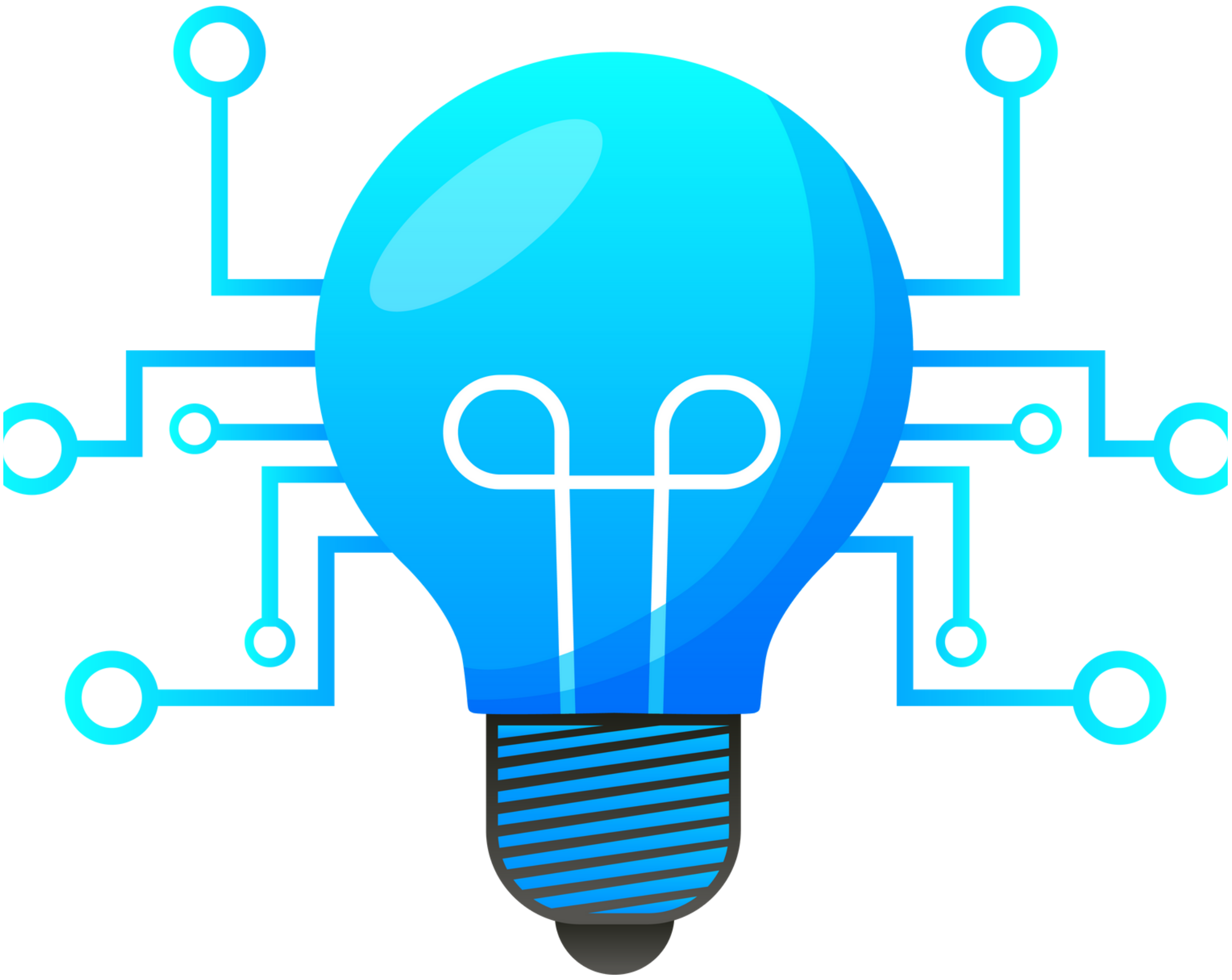

MRT

Embedded systems - GPIO

karim mohamed mohamed elwaraky

Characteristics of an Embedded System

Unlike general computer systems, embedded systems work only for a particular function in a time-bound manner. For instance, a washing machine can not multitask like a laptop. In this regard, here are some unique characteristics of an embedded system.

Sophisticated Functionality

The functionality of no two embedded system applications is bound to be the same. The functionality of a washing machine is different from that of a microwave. However, the functionality of a laptop and a desktop are almost the same.

Real-Time Operation

It doesn't mean live operation. It means the software programs hardware to operate in a time-bound fashion. It could also have two modes: Hard and Soft. The former mode indicates the task has to be completed within the allotted time (ex: clock), but in the soft mode, the system could use additional time over the allotted time (ex: microwave).

Low Manufacturing Cost

As an embedded system design aims for any particular application, it involves less manufacturing cost as compared to a versatile general computing system. As a result, embedded systems also require less power to perform operations.

Processor and Memory

Depending on the type, processor and memory requirements may vary. For instance, small embedded systems would require less memory, but sophisticated systems demand more memory and run on multi-core processors.

Tight Design Constraint

There are many design constraints to consider around the cost, performance, size, and power of an embedded system to realize its absolute performance. These design factors are kept to a minimum to justify their simple function.

What is a Microprocessor?

As its name implies, it is a processing device that converts data into information based on some sets of instructions. It is a very compact electronic chip due to which it is referred to as the **microprocessor**.

In other words, a processing device implemented on a single chip is called a microprocessor. A microprocessor is the most crucial component of a computer or any other computing device. Because, it is entirely responsible for processing data based on instructions to produce information.

In microcomputers, the microprocessor is used as the CPU (**Central Processing Unit**). A typical microprocessor consists of two major parts namely ALU (**Arithmetic Logic Unit**) and CU (**Control Unit**). Intel 8085 or 8086 processing chips are the examples of microprocessors.

Modern microprocessors consist of a small memory unit (cache memory) in addition to the ALU and CU. Now-a-days, microprocessors are being widely used in several applications such as desktop publishing, power plant control, multimeters, medical instruments, etc.

What is a Microcontroller?

A **microcontroller** is an electronic system which consists of a processing element, a small memory (**RAM**, **ROM**, **EPROM**), I/O ports, etc. on a single chip. Therefore, a microcontroller is a tiny resemblance of a microcomputer. It is a quite small and low-cost electronic device which is used in several electronic appliances as the main functioning device.

In electronic systems such washing machines, air conditioners, refrigerators, etc., microcontrollers are used to automate the operation of the device based on user's instructions. Hence, a microcontroller is the backbone of all embedded systems like microwave oven, washing machine, smart refrigerators, etc.

Microprocessors find their application in light sensing and controlling devices, temperature sensing and controlling devices, fire detection and other safety devices, smart measuring instruments, etc.

Types of Memory

computer contains two types of memory: **primary (volatile) and secondary (non-volatile)**. Primary memory (RAM and ROM) allows quick data access and temporary storage for running programs, while secondary memory (HDDs, SSDs, etc.) provides long-term data storage

What Is an Instruction Set Architecture?

An Instruction Set Architecture (ISA) is part of the abstract model of a computer that defines how the CPU is controlled by the software. The ISA acts as an interface between the hardware and the software, specifying both what the processor is capable of doing as well as how it gets done.

The ISA provides the only way through which a user is able to interact with the hardware. It can be viewed as a programmer's manual because it's the portion of the machine that's visible to the assembly language programmer, the compiler writer, and the application programmer.

The ISA defines the supported data types, the registers, how the hardware manages main memory, key features (such as virtual memory), which instructions a microprocessor can execute, and the input/output model of multiple ISA implementations. The ISA can be extended by adding instructions or other capabilities, or by adding support for larger addresses and data values.

What is Computer Architecture?

Computer architecture is the organisation of the components which make up a computer system and the meaning of the operations which guide its function. It defines what is seen on the machine interface, which is targeted by programming languages and their compilers.

Examples:

THE VON NEUMANN ARCHITECTURE

Mathematician John von Neumann and his colleagues proposed the von Neumann architecture in 1945, which stated that a computer consists of: a processor with an arithmetic and logic unit (ALU) and a control unit; a memory unit that can communicate directly with the processor using connections called buses; connections for input/output devices; and a secondary storage for saving and backing up data.

The central computation concept of this architecture is that instructions and data are both loaded into the same memory unit, which is the main memory of the computer and consists of a set of addressable locations. The processor can then access the instructions and data required for the execution of a computer program using dedicated connections called buses – an address bus which is used to identify the addressed location and a data bus which is used to transfer the contents to and from a location.

THE PROS AND CONS OF THE VON NEUMANN ARCHITECTURE

Computers as physical objects have changed dramatically in the 76 years since the von Neumann architecture was proposed. Supercomputers in the 1940s took up a whole room but had very basic functionality, compared to a modern smartwatch which is small in size but has dramatically higher performance. However, at their core, computers have changed very little and almost all of those created between then and now have been run on virtually the same von Neumann architecture.

There are a number of reasons why the von Neumann architecture has proven to be so successful. It is relatively easy to implement in hardware, and von Neumann machines are deterministic and introspectable. They can be described mathematically and every step of their computing process is understood. You can also rely on them to always generate the same output on one set of inputs.

The biggest challenge with von Neumann machines is that they can be difficult to code. This has led to the growth of computer programming, which takes real-world problems and explains them to von Neumann machines.

When a software program is being written, an algorithm is reduced to the formal instructions that a von Neumann machine can follow. However, the challenge is that not all algorithms and problems are easy to reduce, leaving unsolved problems.

HARVARD ARCHITECTURE

Another popular computer architecture, though less so than the von Neumann architecture, is Harvard architecture.

The Harvard architecture keeps instructions and data in separate memories, and the processor accesses these memories using separate buses. The processor is connected to the 'instructions memory' using a dedicated set of address and data buses, and is connected to the ‘data memory’ using a different set of address and data buses.

This architecture is used extensively in embedded computing systems such as digital signal processing (DSP) systems, and many microcontroller devices use a Harvard-like architecture.

Introduction to microcontrollers

A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

Sometimes referred to as an embedded controller or microcontroller unit (MCU), microcontrollers are found in vehicles, robots, office machines, medical devices, mobile radio transceivers, vending machines and home appliances, among other devices. They are essentially simple miniature personal computers (PCs) designed to control small features of a larger component, without a complex front-end operating system (OS).

How do microcontrollers work?

A microcontroller is embedded inside of a system to control a singular function in a device. It does this by interpreting data it receives from its I/O peripherals using its central processor. The temporary information that the microcontroller receives is stored in its data memory, where the processor accesses it and uses instructions stored in its program memory to decipher and apply the incoming data. It then uses its I/O peripherals to communicate and enact the appropriate action.

Microcontrollers are used in a wide array of systems and devices. Devices often utilize multiple microcontrollers that work together within the device to handle their respective tasks.

For example, a car might have many microcontrollers that control various individual systems within, such as the anti-lock braking system, traction control, fuel injection or suspension control. All the microcontrollers communicate with each other to inform the correct actions. Some might communicate with a more complex central computer within the car, and others might only communicate with other microcontrollers. They send and receive data using their I/O peripherals and process that data to perform their designated tasks.

What are the elements of a microcontroller?

The core elements of a microcontroller are:

- The processor (CPU) -- A processor can be thought of as the brain of the device. It processes and responds to various instructions that direct the microcontroller's function. This involves performing basic arithmetic, logic and I/O operations. It also performs data transfer operations, which communicate commands to other components in the larger embedded system.
- Memory -- A microcontroller's memory is used to store the data that the processor receives and uses to respond to instructions that it's been programmed to carry out. A microcontroller has two main memory types:
    a. Program memory, which stores long-term information about the instructions that the CPU carries out. Program memory is non-volatile memory, meaning it holds information over time without needing a power source.
    b. Data memory, which is required for temporary data storage while the instructions are being executed. Data memory is volatile, meaning the data it holds is temporary and is only maintained if the device is connected to a power source.
- I/O peripherals -- The input and output devices are the interface for the processor to the outside world. The input ports receive information and send it to the processor in the form of binary data. The processor receives that data and sends the necessary instructions to output devices that execute tasks external to the microcontroller.

While the processor, memory and I/O peripherals are the defining elements of the microprocessor, there are other elements that are frequently included. The term *I/O peripherals* itself simply refers to supporting components that interface with the memory and processor. There are many supporting components that can be classified as peripherals. Having some manifestation of an I/O peripheral is elemental to a microprocessor, because they are the mechanism through which the processor is applied.

Other supporting elements of a microcontroller include:

- Analog to Digital Converter (ADC) -- An ADC is a circuit that converts analog signals to digital signals. It allows the processor at the center of the microcontroller to interface with external analog devices, such as sensors.
- Digital to Analog Converter (DAC) -- A DAC performs the inverse function of an ADC and allows the processor at the center of the microcontroller to communicate its outgoing signals to external analog components.
- System bus -- The system bus is the connective wire that links all components of the microcontroller together.

- Serial port -- The serial port is one example of an I/O port that allows the microcontroller to connect to external components. It has a similar function to a USB or a parallel port but differs in the way it exchanges bits.

Microcontroller features

A microcontroller's processor will vary by application. Options range from the simple 4-bit, 8-bit or 16-bit processors to more complex 32-bit or 64-bit processors. Microcontrollers can use volatile memory types such as random access memory (RAM) and non-volatile memory types -- this includes flash memory, erasable programmable read-only memory (EPROM) and electrically erasable programmable read-only memory (EEPROM).

Generally, microcontrollers are designed to be readily usable without additional computing components because they are designed with sufficient onboard memory as well as offering pins for general I/O operations, so they can directly interface with sensors and other components.

Microcontroller architecture can be based on the Harvard architecture or von Neumann architecture, both offering different methods of exchanging data between the processor and memory. With a Harvard architecture, the data bus and instruction are separate, allowing for simultaneous transfers. With a Von Neumann architecture, one bus is used for both data and instructions.

Microcontroller processors can be based on complex instruction set computing (CISC) or reduced instruction set computing (RISC). CISC generally has around 80 instructions while RISC has about 30, as well as more addressing modes, 12-24 compared to RISC's 3-5. While CISC can be easier to implement and has more efficient memory use, it can have performance degradation due to the higher number of clock cycles needed to execute instructions. RISC, which places more emphasis on software, often provides better performance than CISC processors, which put more emphasis on hardware, due to its simplified instruction set and, therefore, increased design simplicity, but because of the emphasis it places on software, the software can be more complex. Which ISC is used varies depending on application.

When they first became available, microcontrollers solely used assembly language. Today, the C programming language is a popular option. Other common microprocessor languages include Python and JavaScript.

MCUs feature input and output pins to implement peripheral functions. Such functions include analog-to-digital converters, liquid crystal display (LCD) controllers, real-time clock (RTC), universal synchronous/asynchronous receiver transmitter (USART), timers, universal asynchronous receiver transmitter (UART) and universal serial bus (USB) connectivity. Sensors gathering data related to humidity and temperature, among others, are also often attached to microcontrollers.

Software Design Process

The design phase of software development deals with transforming the customer requirements as described in the SRS documents into a form implementable using a programming language. The software design process can be divided into the following three levels of phases of design:

1. Interface Design
2. Architectural Design
3. Detailed Design

1. **Architecture –** This is the conceptual model that defines the structure, behavior, and views of a system. We can use flowcharts to represent and illustrate the architecture.
2. **Modules –** These are components that handle one specific task in a system. A combination of the modules makes up the system.
3. **Components –** This provides a particular function or group of related functions. They are made up of modules.
4. **Interfaces –** This is the shared boundary across which the components of a system exchange information and relate.
5. **Data –** This is the management of the information and data flow.

Interface Design: *Interface design* is the specification of the interaction between a system and its environment. this phase proceeds at a high level of abstraction with respect to the inner workings of the system i.e, during interface design, the internal of the systems are completely ignored and the system is treated as a black box. Attention is focused on the dialogue between the target system and the users, devices, and other systems with which it interacts. The design problem statement produced during the problem analysis step should identify the people, other systems, and devices which are collectively called *agents*. Interface design should include the following details:

- Precise description of events in the environment, or messages from agents to which the system must respond.
- Precise description of the events or messages that the system must produce.
- Specification of the data, and the formats of the data coming into and going out of the system.

- Specification of the ordering and timing relationships between incoming events or messages, and outgoing events or outputs.