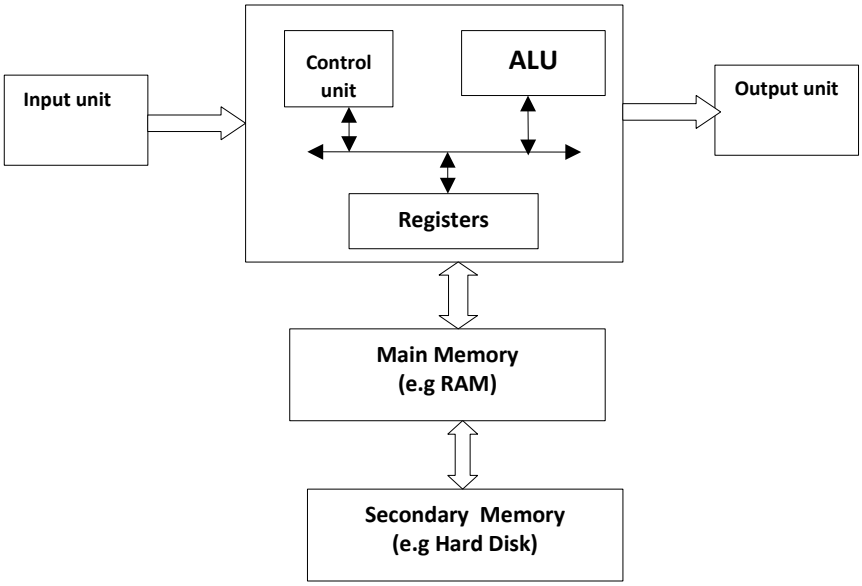
Unit -1 Embedded systems

Block diagram of General purpose computer system



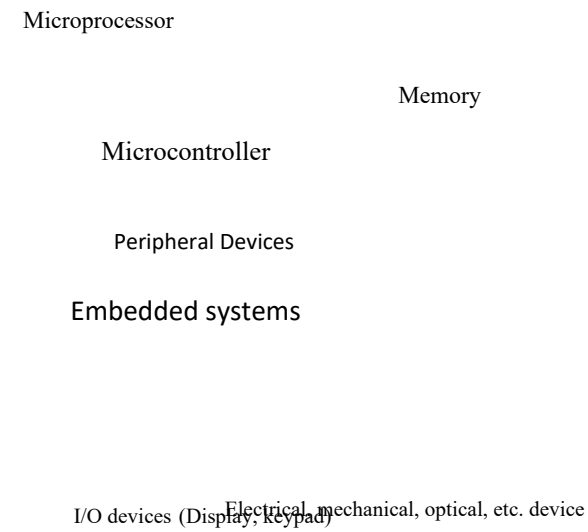
A computer is a combination of two components namely hardware and software. Hard disk, CPU, memory (volatile also called RAM), CD-ROM and printers are a few examples of the hardware which are considered as part of a computer. The software are applications for various purposes and operating systems (Windows and Linux are two highly used operating systems) to manage the applications and computer hardware efficiently. Desktops and laptops are examples of such digital computer systems.

Input devices, such as keyboard and mouse, allow a user to provide input to the computer system. The users can see useful information on output devices such as monitors and printers. These input/output (I/O) devices communicate with the CPU through different types of interfaces. The CPU carries out all kinds of processing and comprises three basic building blocks, namely, arithmetic logic unit (ALU), control unit and set of registers.

The main memories are random access memory (RAM) and read-only memory (ROM). These memories are accessible directly from the processor. Low access-time and random access are their major attributes. Hard disk is an example of the secondary memory and does not interact with the CPU directly. Its information access-time is relatively higher than that of the primary memories

EMBEDDED SYSTEMS

A generic block diagram of an embedded system is shown in Figure



An embedded system is developed for a specific application to perform dedicated task(s). An embedded system consists of a processor, memory and other interfaces and is the building blocks of a microcontroller which is embedded inside a specific embedded system. The software is designed to handle a range of different tasks and is normally programmed to ROM memory. The ROM in embedded systems is employed to store programs permanently even when the power is turned off.

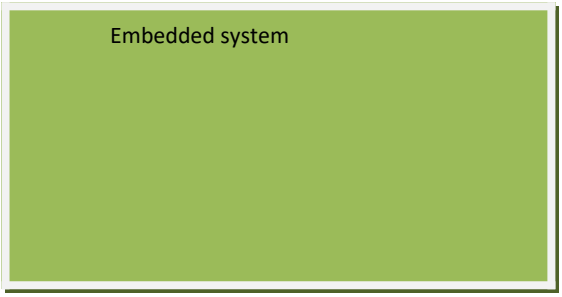
Advantages:

- small size,
- low cost, and
- low power requirements

Examples of Embedded system:

- cell-phones and other handheld devices,
- blood pressure monitoring equipment
- digital multi-meters
- temperature and humidity measuring devices

From the system hardware perspective, an embedded system embeds a microcontroller inside it, which in turn contains a microprocessor core. This is a generic hierarchical view



MICROCONTROLLERS

1. **Applications in communication:** Radio, telephone, cellular phones, answering machines,fax Machines, wireless routers.
2. **Consumer electronics**: Washing machine, clocks and watches, games and toys, remote controls, Audio/video electronics.
3. **Automotive systems:** Braking system, electronic ignition, locks, power windows and seats, collision avoidance
4. **Commercial usage:** ATM machines, bar code readers, elevator controllers.
5. **Medical treatments:** Cancer treatments, dialysis machines, blood pressure measuring equipment, (ECG), etc.
6. **Industrial:** Process automation, oil refineries, food processing plants, paper and board mills, etc.
7. **Military use:** Missile guidance systems, global positioning systems, surveillance systems.

Design Parameters of Embedded Systems

While designing an embedded system, the important factors to b considered are -

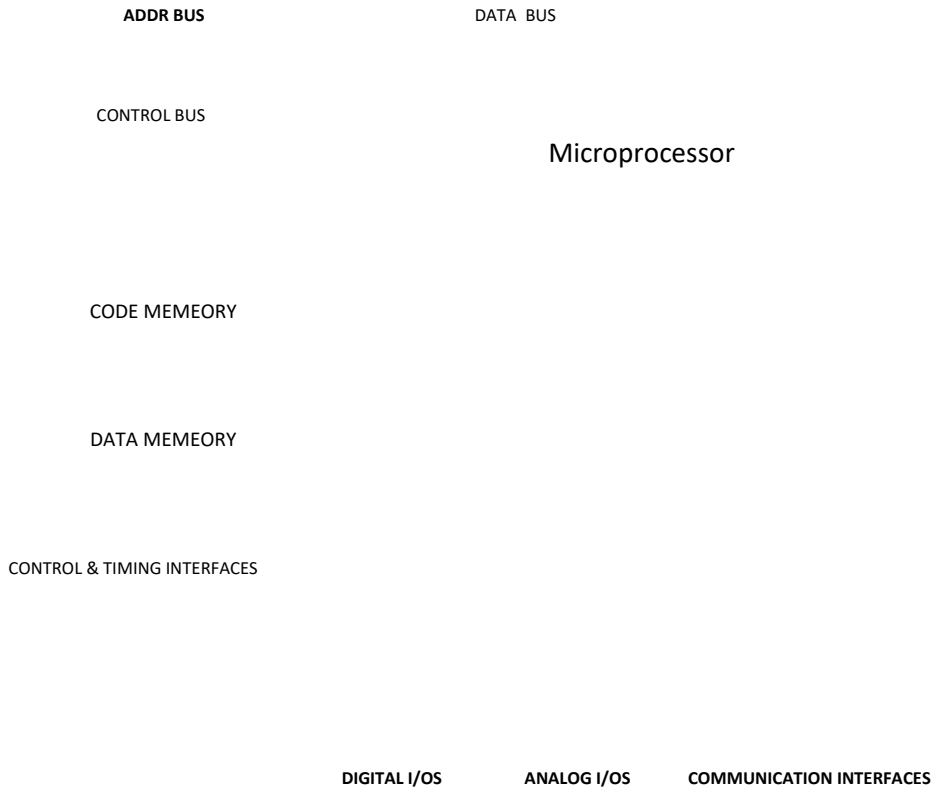
1. Power consumption
2. Speed of execution
3. System size and weight
4. Performance

Microcontrollers

A microcontroller combines a microprocessor, read only memory (ROM), random access memory (RAM), and input/output (I/O) peripheral devices on a single chip. For this communication among subsystems, different buses are used. A bus is a collection of wires over which digital signals propagate in the form of 0s

and 1s. Buses internal to the chip can be broadly categorized in three groups namely data, address, and control.

The basic block diagram of a microcontroller is as shown below.



Any memory location inside RAM, ROM and in registers is identified by unique numbers called addresses.

Address signals flow over the address bus. The size of the address bus decides the unique addresses generated by a microprocessor.

The data bus carries the data. Increasing the size of the data bus allows more number of bits to be communicated between two subsystems.

Control signals such as read/write inform whether the CPU is interested in reading some information or wants to write some information to the location whose address was generated over the address bus.

Major component of a microcontroller is the general purpose peripheral device called port, which provides a physical connection between the microcontroller and the outside world devices. Based on the type of the device being connected with the microcontroller, ports can be configured as input or output. In

addition, some of the port registers are used to monitor the device status, while other registers are used to configure the direction of the port.

Information exchange between a microcontroller and a device can take place in parallel or serially. Broadly, I/O ports can be classified in two categories

• **Parallel Port:** When two communicating entities are connected with a group of lines, a number of data bits can be exchanged simultaneously.

• **Serial Port:** In this case, there is only one line for the data to travel between two communicating parties.

Difference between the Microcontrollers and Microcomputers

Microcomputers	Microcontrollers
RAM, ROM, CPU and I/O interfaces are integrated in different chips on a motherboard	RAM, ROM, CPU and I/O interfaces are embedded inside a single chip
More speed of memory and other peripherals	The speed of memory and other peripherals is less
Size of the memory is more	Size of the memory is less

Memory: Information Storage Device

All Digital systems store information in binary digits called bits. These bits are used for representing Operators, operands, and addresses. There are two states for a binary bit. A `1' represents the presence of a voltage and a `0' represents the absence of voltage. This representation is called positive logic.

To store binary information, CMOS circuits can be built and a collection of a large number of such circuits is called memory. Memory stores data sequentially which is as shown in Figure.

Memory Address	Data
0x201A 3241	10110111
0x201A 3242	01010100
. . . . .	. . . . .
0x201A 3244	1011 1101
0x201A 3243	1010 1101
0x201A 3242	1011 1100

Memories are made up of data storage locations, which are uniquely addressable and can be accessed by the processor. Each storage location can hold 8 bits of information. Since each byte is uniquely addressable, such memories are called byte addressable memories.

During a write operation, data is transferred from processor to memory. Whereas in Read operation, data is transferred from memory to the processor

Based on read and write operations, memories are grouped in two categories namely

- Read-only memory (ROM) and
- Random access memory (RAM).

The ROM allows read operations only

In case of RAM both read as well as write operations can be performed by the microprocessor.

Read Only Memory

The memory that allows the processor to only read its contents is Read Only Memory (ROM). Another characteristic of ROM is that they can store information permanently even when no power is applied. The information in the ROM is therefore non volatile.

Instructions or a program code are stored in a ROM.

Example: The boot sequence with which an operating system is loaded.

Based on different methods of programming the ROM, ROM is classified into

- PROM
- EPROM
- EEPROM
- Flash memory

Programmable ROMs (PROMs):

These can be programmed only once. Any change to the contents will require the replacement of the chip.

Erasable Programmable ROMs (EPROMs)

These can be reused by erasing and re-programming. Conventionally, information was erased and written to with the help of ultraviolet (UV) light. The changes cannot, however, be made while the chip is installed in the system. Another drawback of EPROM is that the whole chip needs to be completely erased.

Electrically Erasable PROMs (EEPROMs)

These are EPROMs but their contents can be erased and written to by applying electric signals to the storage cells. Furthermore, erasure and writing can take place while the chip is installed in the circuit.

Flash memory

It is a type of EEPROM that allows read and write operations to be carried out in large multi-byte blocks. In general, the erase cycles for non-volatile memories are slow. Flash memory allows erasing large block sizes, which provides these memories a significant speed. One of the limitations of ash memories as well as EEPROMs is their limited number of read and write cycles.

Random Access Memory

The memory which allows the processor to read from and write to its locations is known as Random Access Memory (RAM).

One limitation of RAM is that information stored in it is lost as soon as the power applied to it is removed. On the other hand, RAM is not limited by the number of read and write cycles and is more suitable for storing data that is updated frequently. The read and write operations of RAMs are faster than those of ROMs.

There are different types of RAMs namely-

- Dynamic RAM (DRAM)
- Static RAM (SRAM)

Dynamic RAM (DRAM) is the most commonly used type of RAM. Each memory cell of a DRAM, can store one bit of information. It is made up of two transistors and a capacitor. The transistor acts as a switch while the

Capacitor holds the charge. As the capacitor discharges, the voltage representing `1' needs to be refreshed. This refresh operation is performed a number of times in one second and results in reducing the memory operating speed.

Static RAM (SRAM) employs a flip-flop for storing a bit in a memory cell. A Flip-flop requires 4 to 6 transistors and does not require refreshing circuitry. SRAM is faster than DRAM because of the absence of refresh cycles.

On a given chip SRAM yields less memory space as compared to DRAM. This is because more numbers of transistors required by an SRAM cell. This also makes them more expensive. Because of their features, SRAM is normally used for building cache memories while DRAM is employed for RAM chips. In comparison to ROM, the access time of RAMs is less.

Differences between DRAM & SRAM

Static RAM	Dynamic RAM
➢ SRAM uses transistor to store a single bit of data	➢ DRAM uses a separate capacitor to store each bit of data
➢ SRAM does not need periodic refreshment to maintain data	➢ DRAM needs periodic refreshment to maintain the charge in the capacitors for data
➢ SRAM's structure is complex than DRAM	➢ DRAM's structure is simplex than SRAM
➢ SRAM are expensive as compared to DRAM	➢ DRAM's are less expensive as compared to SRAM
➢ SRAM are faster than DRAM	➢ DRAM's are slower than SRAM
➢ SRAM are used in Cache memory	➢ DRAM are used in Main memory

Aligned and Unaligned Memory Accesses

For a byte (8bit) addressable memory, each memory access of size byte is inherently aligned. A 32-bit processor will have a 32-bit data bus to access the memory. It is possible to perform memory read/write operations of half word (16-bit) and word (32-bit) size. If a memory access, of size halfword, is performed from an even address, then this memory access is an aligned access. On the other hand, if an odd address is used, then the resulting memory access is an unaligned access.

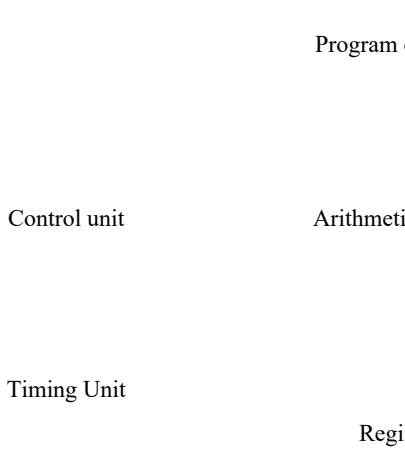
Similarly, for a 32-bit word memory access, word aligned means the data is stored on a word boundary, i.e., the memory address accessed is divisible by 4. A word memory access from an address not divisible by 4 is termed unaligned word access.

THE MICROPROCESSOR

A Microprocessor, also known as a central processing unit, has a capability of executing instructions at an extremely high speed. Broadly, a microprocessor has five basic components namely –

- Arithmetic logic unit (ALU),
- Control unit,
- Bank of registers,
- Interconnection buses, and
- Timing unit.

The block diagram of a microprocessor is shown in Figure.



The functions of various blocks of a microprocessor is described below

Arithmetic Logic Unit (ALU)

Arithmetic logic unit (ALU) is to perform arithmetic operations such as addition, subtraction and logical operations such as, such as AND, OR operations.

Control unit:

As the Microprocessor works with only binary digits, each operation should be encoded by a unique binary combination, which is termed the operational code or the op-code. Various steps required for the execution of an operation are performed using binary signals, which are known as control signals. These control signals are generated by the control unit.

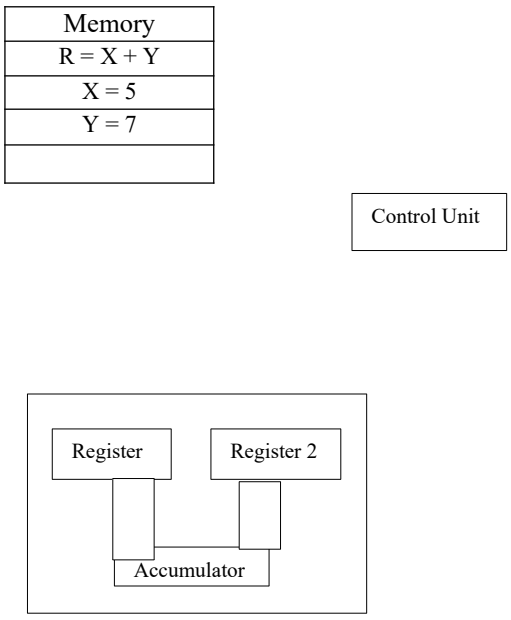
The operations and operands are initially stored in the memory. They are brought inside the microprocessor using buses and are stored temporarily in Microprocessor registers. Data transfers among different processor registers as well as ALU are performed using the buses inside the microprocessor.

The function of a microprocessor is to execute a user program. The executable program is stored in memory initially and is executed instruction by instruction. During program execution, the microprocessor performs different operations.

The instructions from memory and data from memory or I/O ports is stored temporarily inside the microprocessor using registers.

Working of a microprocessor

The working of the microprocessor is as shown in the following figure.



Let us assume that the microprocessor executes R = X + Y addition instruction. Let's assume that this instruction is compiled and is present in the memory. This instruction will consist of three fields. One of them will contain the operational code representing addition, while the other two fields will contain the memory addresses of the locations where variables X and Y are stored. The PC register contains its memory address.

The steps that are carried out by the microprocessor are listed below.

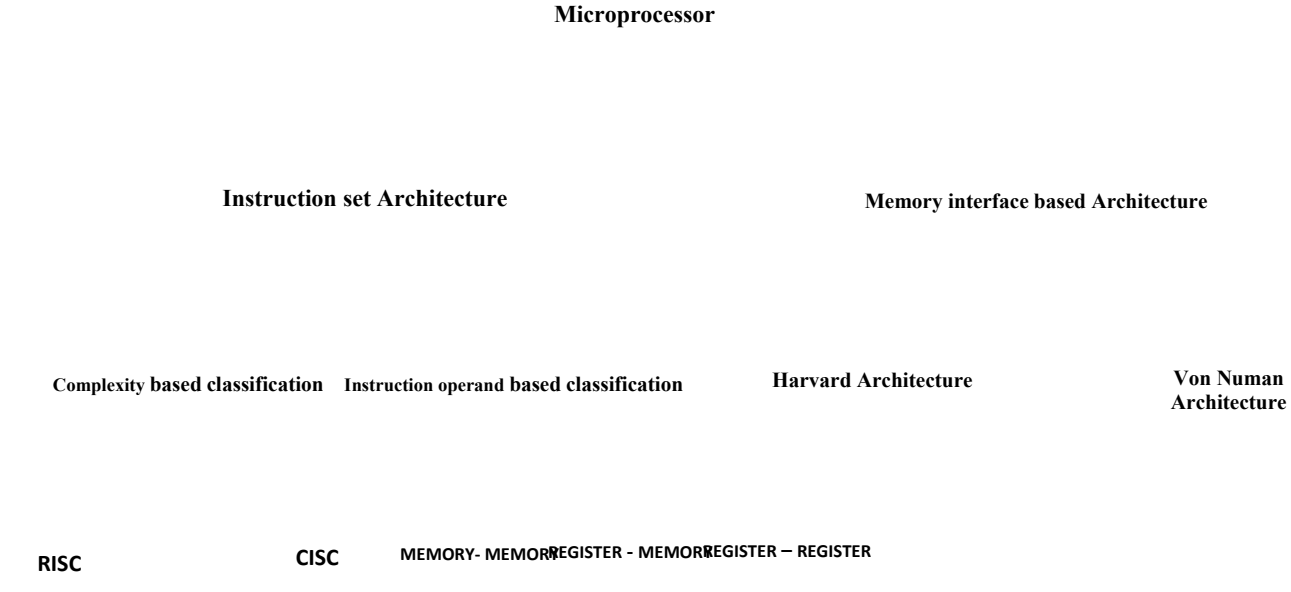
- The microprocessor fetches the instruction from the address that is present in PC on the address bus
- The control unit decodes its opcode
- Processor executes R = X + Y by
    - fetching the current value of X from the memory
    - fetching the current value of Y from the memory
    - instructing the ALU to add these two numbers
    - writing the sum back to the memory address of R

These are the steps carried out by the processor to execute one instruction.

Difference between Microprocessor and Microcontroller:

Sl. No	Microprocessor	Microcontroller
1	CPU is stand alone, RAM,ROM ,I/O devices and timers are separate and interfaced with CPU	CPU, RAM, ROM,I/O devices and TIMERS are all located on a single chip.
2	Designer can decide on the amount of RAM, ROM and I/O ports.	Fixed amount of RAM,ROM and I/O ports
3	Expensive applications	Applications in which cost, space and power are critical
4	Versatile and general purpose	Not very versatile
5	Program memory and data memory are same	Uses different program memory and data memory.
6	Large number of instructions with flexible addressing modes	Limited number of instructions with few addressing modes
7	Very few instructions which have bit handling capability	Many instructions with bit handling capability
8	System cost is more	System cost is less
9	Clock frequency > 1 G Hz	Clock frequency 10-20 M Hz

Microprocessor Architecture Classification



The architecture of the microprocessor can be classified as-

1. Instruction Set Architecture (ISA)
2. Memory interface based Architecture

Instruction Set Architecture (ISA)

The Instruction set Architecture is further subdivided based on

- Complexity of instructions and
- Instruction operands.

Complexity-Based ISA classification

Based on the ISA complexity the microprocessors are categorized into two groups:

1. Complex instruction set computer (CISC) and
2. Reduced instruction set computer (RISC).

The differences between RISC & CISC architecture is as shown below

RISC Architecture		CISC Architecture
Complex Tool Development	Software Tools	Simplified Tool Development
Simplified Hardware	Processor	Complex Hardware

Complex instruction set computers (CISC):

A complex instruction set computer (CISC) is a computer architecture in which single instructions can execute several low-level operations (such as a load from memory, an arithmetic operation, and a memory store) or are capable of multi-step operations or addressing modes within single instructions.

The key features of the CISC architecture are discussed below

- This architecture is suited where the processor speed faster than available memories
- A single complex instruction can perform many operations.
- Complex instructions require more clock cycles to complete
- Most of the instructions can access memory.
- A program running on a CISC architecture involves a small number of complex instruction.
- In CISC the complexity is embedded in the processor hardware, making the compilation tools design simpler.

  Example: Intel (x86) and Freescale 9S12.

Reduced Instruction Set Computers (RISC):

A Reduced Instruction Set Computer is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions rather than the highly-specialized set of instructions.

The key features of the RISC architecture are discussed below

- This architecture is suited where the processor speed matches that of memories.
- Simplified instructions set.
- Each complex operation is broken into multiple simplified operations and dedicated instructions are provided for these operations.

  Example: load and store instructions
- As the instructions set are simple they are executed in a single processor clock cycle.
- Requires fewer memory addressing modes.
- Simple hardware architecture.

  Example: MIPS, ARM, SPARC and PowerPC.

Difference between the RISC and CISC Processors

RISC	CISC
It is a Reduced Instruction Set Computer.	It is a Complex Instruction Set Computer.
It requires multiple register sets to store the instruction.	It requires a single register set to store the instruction.
RISC has simple decoding of instruction.	CISC has complex decoding of instruction.
It uses a limited number of instructions	It uses a large number of instructions
It uses LOAD and STORE in the register-to-register a interaction of program.	It uses LOAD and STORE instruction in the memory-to-memory interaction of a program.
RISC has more transistors on memory registers.	CISC has transistors to store complex instructions.
The execution time of RISC is very short.	The execution time of CISC is longer.
It has fixed format instruction.	It has variable format instruction.
The program written for RISC architecture needs to take more space in memory.	Program written for CISC architecture tends to take less space in memory.
Example: ARM, Power Architecture, Alpha, AVR, ARC and the SPARC.	**Example:** VAX, Motorola 68000 family and the Intel x86 CPUs.

Instruction Operand-Based ISA Classification

The operands for an instruction can be specified either using memory or registers or combination of both. The ISA classification based on how the operands are specified can be categorized in the following groups.

1. **Memory-memory:**

   This type of ISA allows more than one operand to be specified in memory.

   Example: VAX and PDP series.

2. Register-memory:

These architectures allow one operand of an instruction to be specified in memory, while the other operand is in CPU register. In this ISA the individual instructions execute faster. However, this may require more number of instructions to complete the same task.

Example: x86 and Motorola 68k.

3. Register-register:

This ISA classification is also called **load-store architecture**. Memory is accessed using **load and store** instructions. All instructions other than load and store instructions get their operands and store their results to registers. The execution of these instructions is very fast. But this ISA requires the most number of instructions to complete a given task. Since in many cases the immediate results are not stored to the memory, and are temporarily placed in the registers, it might be used by the subsequent instructions.

Example: ARM and MIPS.

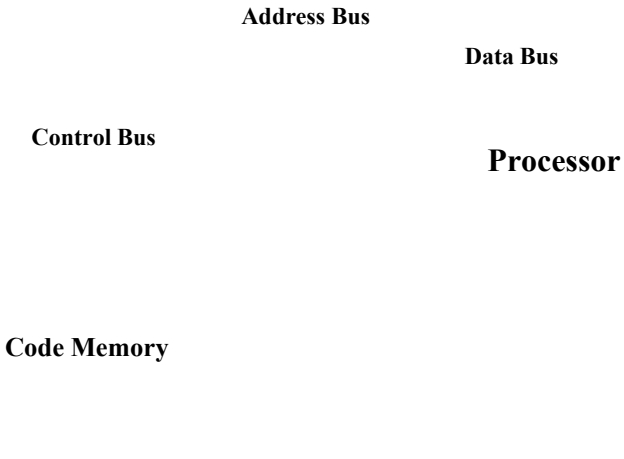
Memory interface based Architecture classification

There are two types of memory interface architectures, namely,

- **Von Neumann Architecture and**
- **Harvard Architecture**

Von Neumann Architecture

The following figure shows the architecture of Von Neumann Architecture



The von Neumann architecture uses a common bus for both data and code memory. As a result either an instruction can be fetched from memory or data can be read/written to/from memory during each memory access cycle. Instructions and data are stored in the same memory subsystem and share a common bus to the processor.

Example: Micrprocessors

Harvard Architecture

The following figure shows the architecture of Harvard Architecture

CODE MEMORY DATA MEMORY

The Harvard architecture, uses separate buses for accessing code and data memories. Hence data and instructions can be accessed simultaneously. In addition, the next instruction may be fetched from memory at the time when the previous instruction is about to finish its execution. However, main memory access time is a major bottleneck in the overall performance of the system.

Example:. ARM-based microcontrollers.

Difference between Von Neumann Architecture and Harvard Architecture

Von Neumann Architecture	Harvard Architecture
Based on the stored program computer concept	Harvard Mark I relay based computer system
Use common bus to transfer data and the instructions	Use separate buses to transfer data and the instruction
Use same physical memory address for instructions and the data	Use separate physical memory address for instructions and the data
Need two clock cycles to execute single instruction	Instructions are execute in single cycle no need two cycles
Cheaper to use	Costly than Von Neumann
Simple to use	More complex than Von Neumann
CPU have not access to read or write data and the instructions in a same time	CPU have access to read or write data and the instructions in a same time
Mainly use in personal computers	Mainly use in the micro controllers
Speed is limited than Harvard	Speed is high rather than von Neumann

Performance Comparison of Different Architectures

A microprocessor's architecture can be compared against other architectures by running benchmark programs. The following performance evaluation equation is used to quantify the execution speed of a microprocessor.

$$\textbf{Execution Time} = T_c \sum_{i=0}^{N} Ci \quad \text{- - - - - - - -- - - -(1.1)}$$

In (1.1), Tc is the cycle time, Ci represents the number of cycles required for i th instruction execution, and N is the number of instructions in the test program. If each instruction requires the same number of cycles for execution, then the expression in (1.1) is simplified to

$$\textbf{Execution Time} = T_c N C. \text{- - - - - - - - - - - - - - - -- - - - - - -- (1.2)}$$

From the expression in (1.2) we observe that there are three different possible ways to speed up the execution of a microprocessor and are listed below:

1. Use fewer instructions for a given program. In other words, it is possible to improve the execution speed by efficient programming.
2. Reduce the number of cycles for the instructions. This is mainly dependent on the instruction set architecture of the microprocessor.
3. Speed up the clock frequency of the microprocessor or equivalently reduce the cycle time. This refers to the maximum clock frequency of the processor. The maximum clock frequency depends on –
    a) The physical limitation of processor clock frequency
    b) The technical limitation associated with increasing the clock frequency

a) The physical limitation of processor clock frequency:

Let us assume a microprocessor is running at 3 GHz clock frequency. i.e., one clock cycle takes $\frac{1}{3}$ ns. Now the electrical signals travel a physical distance of approximately 10 cm during $\frac{1}{3}$ ns. Assuming that a circuit is working synchronously, it is necessary that a clock signal is available at different parts of the circuit simultaneously. This can be ensured by requiring the propagation delay much less than clock cycle time. Let us use a factor of 10, which will require the size of the circuit to be about 1 cm. The CPU core die size of Intel Core2 Duo Processor is approximately 1 cm. Now if we want to double the speed of this processor, the size of the processor core should become one-half. If this size limitation is violated, then some parts of the processor circuit are operating in the current clock cycle, while some other parts of the circuit are still not done with the previous clock cycle. This type of distributed system is very hard to deal with in practical systems.

b) The technical limitation associated with increasing the clock frequency.

As the CPU cores become smaller and smaller, the problem of heat dissipation starts arising. As processors are made up of transistors and these transistors can be switched ON and OFF at the clock frequency, each transistor dissipates power when switched from one state to another. Switching at fast speed leads to more power dissipation. When the processor core size is reduced to increase the clock speed, the corresponding amount of heat generated by these switching transistors couldn't be dissipated easily due to the reduced chip size. The amount of speed gain and the corresponding increase in the cost of associated cooling mechanism are some of the practical barriers in increasing the processor speeds further. One natural solution to this limitation is to use multiple processors of moderate speed rather than to have one processor of too high speed, leading to multi-core processor architecture.

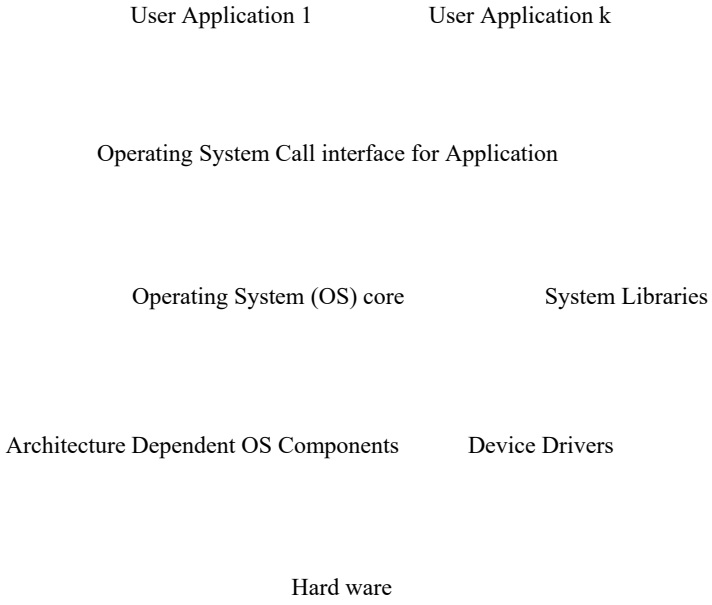
Software System and Development Tools

The intelligence of a microcontroller-based embedded system lies in its software which is a collection of instruction sequences, also called functions, that are stored in memory (either ROM or RAM). These functions are implemented to perform certain tasks and are fundamental building blocks of a software system. Based on the collective functionality of a set of functions, the software subsystem includes -

- Operating system,
- Device drivers, libraries and
- User applications.

Operating system:

Operating system includes the microprocessor, memory, Input/Output devices. The operating system manages these devices using the hardware architecture dependent OS components. Also it allows user applications to use these devices in a systematic way through OS. These interfaces provided by the OS is as shown in Figure



Device drives or drivers:

A device driver is a software sub-system, which allows the operating system to communicate with the hardware devices. Drivers for some of the standard devices are sometimes integrated as an essential component of the operating system. But for some hardware devices, the drivers are provided by the vendor.

Libraries:

The software library is a collection of function calls developed for specific job and is made available to the user. By using a library, the software application developer reuses the specific functionality already implemented by the library. Libraries allow sharing the code in a modular fashion, as well as easing the distribution of the code.

Applications:

Application is a collection of one or more programs, which are designed to perform operations for a specific application requirement. Application software cannot run (or execute) on its own. It is dependent on the OS and libraries.

Software Development Tools

The process of converting user program to executable outputs involves two main steps:

(1) Compilation and

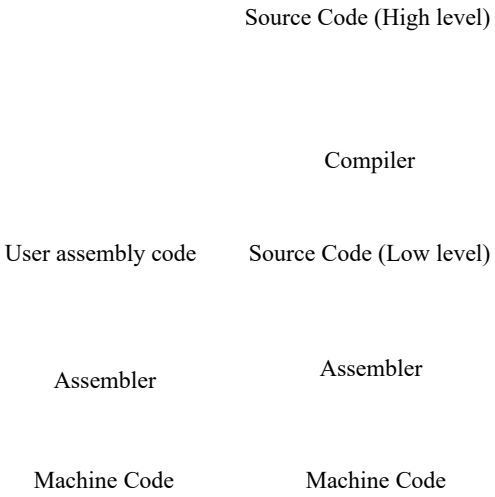
(2) Linking.

Compilation Process

The tool used in the compilation process is called either a compiler or an assembler depending on the type of the user program source file.

If the user program is written in assembly language, then an assembler is used to convert source code to an object code, which is also called machine code. In this process, an assembly instruction is converted to its equivalent opcode or machine code. Assembly language programs are also called low level programs.

If the user program is written in a high level language, then a compiler is used to convert it to the machine code. Most of the compilers also provide the assembly equivalent code, in addition to machine code. The process of converting a user program to machine code using either a compiler or an assembler is as shown in Figure



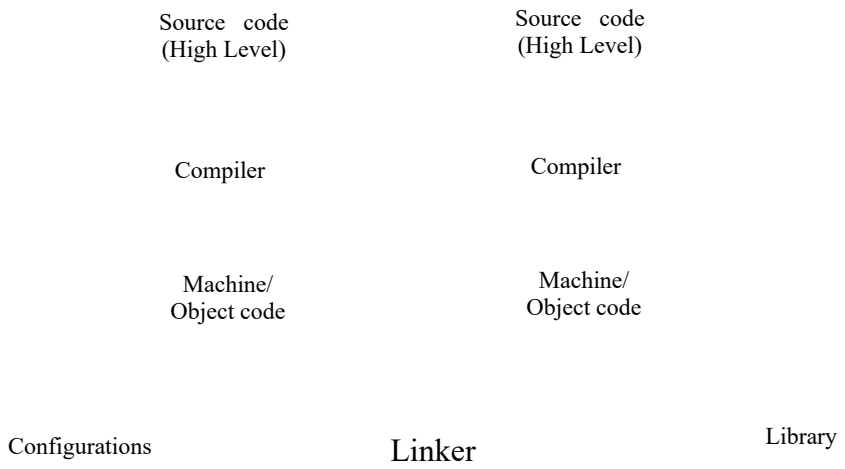
The selection of a compiler is not only dependent on the choice of the high level language, but also on the hardware platform to be used for running the program.

For example, a program is written in C language and the user wants to run its corresponding executable on an Intel microprocessor then we need to compile the C language program using Intel's C compiler.

Building an Executable Using a Linker

The job of a linker is to construct an executable by combining different object codes (or object files) obtained after compiling the user program codes. In the process of linking, the linker has to decide the locations (addresses) of different object codes and data segments.

In addition, some of the functions used in the program are not implemented by the programmer, rather they are provided by a library. It is the job of a linker to get the desired modules from different libraries and integrate them in the process of constructing the executable. The functioning of the linker is shown in Figure



The content of a library is nothing but a collection of different object codes and the library itself is constructed using a linker. The linker, in addition to building the executable, can generate, many other files containing useful information for the programmer. Similarly, the list file can also be generated by the tools, which provides the disassembly of the code which is helpful to the programmer in debugging.

The MAP and list files optional outputs from the linker. Another important related concept is the dynamic linking, where the libraries are not used at the time of building an executable, rather they are used during the execution at runtime and are called dynamic link libraries or in short DLLs.