# C Programming

## Day1

***Day 1:- C Programming***

***Today's Session  We cover***

--What is Programming Language

--Define Computer Program

--Number System

--Types of Software

--List the component of computer required to execute a program

--Define an algorithms

--Write algorithms for solution of a given problems.

--Draw flowchart for solution of a given problems.

--State different programming languages.

--What is Translators

--Program Development Environment

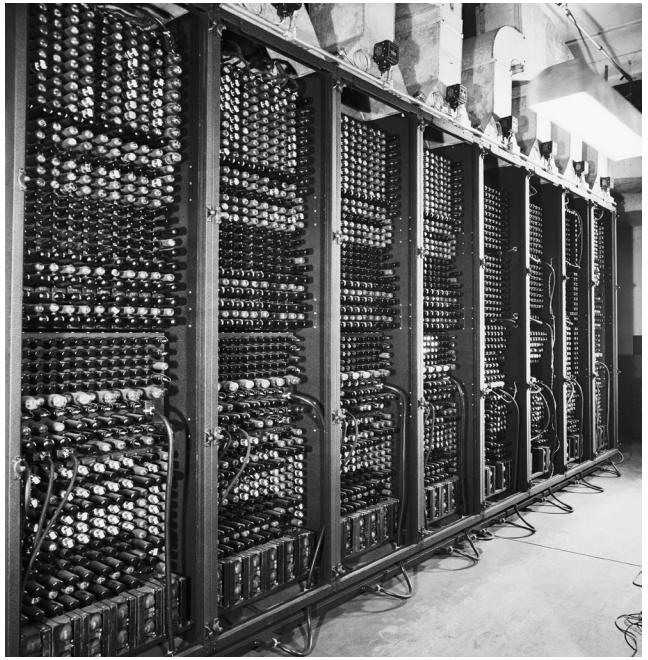
# Classification of generations of computers

- The evolution of computer technology is often divided into five generations.

| Generations of computers | Generations timeline | Evolving hardware |
|---|---|---|
| First generation | 1940s-1950s | Vacuum tube based |
| Second generation | 1950s-1960s | Transistor based |
| Third generation | 1960s-1970s | Integrated circuit based |
| Fourth generation | 1970s-present | Microprocessor based |
| Fifth generation | The present and the future | Artificial intelligence based |

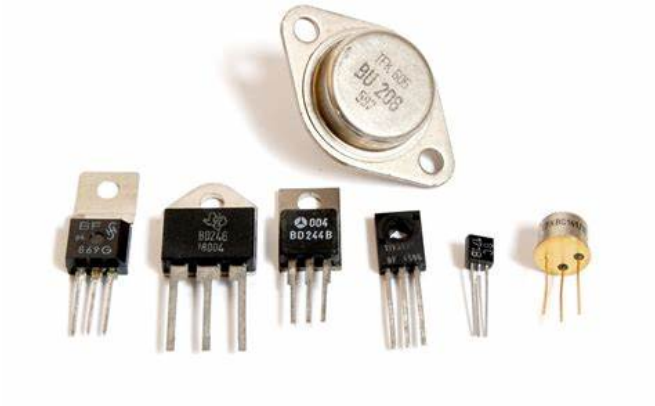
# Main characteristics of first generation computers are

| | |
|---|---|
| Main electronic component | Vacuum tube. |
| Programming language | Machine language. |
| Main memory | Magnetic tapes and magnetic drums. |
| Input/output devices | Paper tape and punched cards. |
| Speed and size | Very slow and very large in size (often taking up entire room). |
| Examples of the first generation | IBM 650, IBM 701, ENIAC, UNIVAC1, etc. |



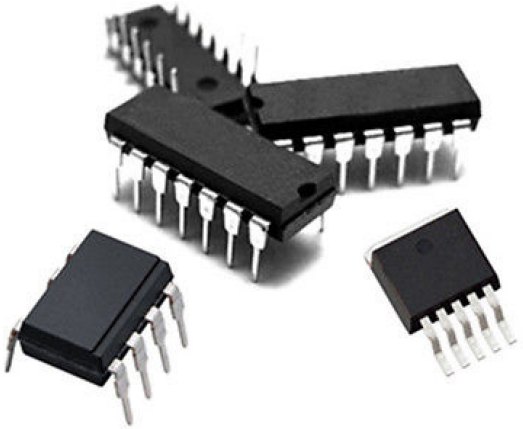
# Main characteristics of second generation computers are:-

| | |
|---|---|
| Main electronic component | Transistor. |
| Programming language | Machine language and assembly language. |
| Memory | Magnetic core and magnetic tape/disk. |
| Input/output devices | Magnetic tape and punched cards. |
| Power and size | Smaller in size, low power consumption, and generated less heat (in comparison with the first generation computers). |
| Examples of second generation | PDP-8, IBM1400 series, IBM 7090 and 7094, UNIVAC 1107, CDC 3600 etc. |



# Main characteristics of third generation computers are:

| | |
|---|---|
| Main electronic component | Integrated circuits (ICs) |
| Programming language | High-level language |
| Memory | Large magnetic core, magnetic tape/disk |
| Input / output devices | Magnetic tape, monitor, keyboard, printer, etc. |
| Examples of third generation | IBM 360, IBM 370, PDP-11, NCR 395, B6500, UNIVAC 1108, etc. |



# Main characteristics of fourth generation computers are:

| | |
|---|---|
| Main electronic component | Very large-scale integration (VLSI) and the microprocessor (VLSI has thousands of transistors on a single microchip). |
| Memory | semiconductor memory (such as RAM, ROM, etc.) |
| Input/output devices | pointing devices, optical scanning, keyboard, monitor, printer, etc. |
| Examples of fourth generation | IBM PC, STAR 1000, APPLE II, Apple Macintosh, Alter 8800, etc. |

# Main characteristics of fifth generation computers are:

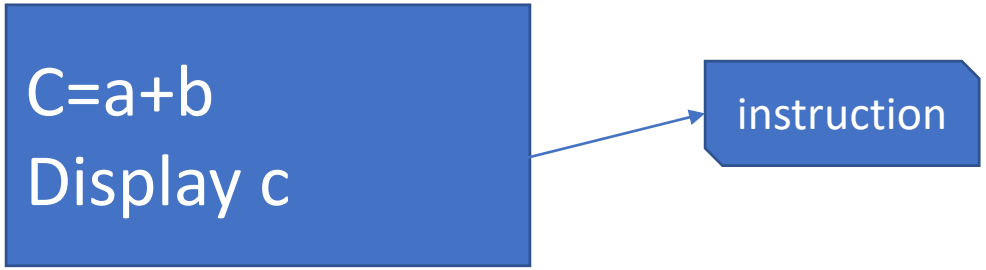
| | |
|---|---|
| Main electronic component | Based on artificial intelligence, uses the Ultra Large-Scale Integration (ULSI) technology and parallel processing method (ULSI has millions of transistors on a single microchip and Parallel processing method use two or more microprocessors to run tasks simultaneously). |
| Language | Understand natural language (human language). |
| Size | Portable and small in size. |
| Input / output device | Trackpad (or touchpad), touchscreen, pen, speech input (recognize voice/speech), light scanner, printer, keyboard, monitor, mouse, etc. |
| Example of fifth generation | Desktops, laptops, tablets, smartphones, etc. |

# Classification of Computers by Size

- **Supercomputers**
- **Mainframe computers**
- **Minicomputers**
- **Personal computers (PCs) or microcomputers**

# *Define Computer Program*

- Program is a set of instructions
- Given in a particular sequence
- Having a predefined meaning
- E.g:-




- An instruction is a combination of some words. Which have a predefined meaning.
- Instructions are given in a particular sequence.

# *What is Programming Language*

**Are Classified into two type:**

**1)Low Level Language:-**

**a)Machine Level Language:-combination of 0 zeros and 1**

**b)Assembly Language:-some symbols called *mnemonic***

**----------**

**2)High Level Language:-human readable language i.e.English**

**3)Middle Level Language:-C Language**

# Translators

- Any program written in a **high level language** is known as **source code**. However, computers cannot understand source code. Before it can be run, source code must first be translated into a form which a computer understands - this form is called **object code**.
- A translator is a **program** that converts source code into object code. Generally, there are three types of translator:
- **compilers**
- **interpreters**
- **assemblers**

# Compilers

- A compiler takes the source code as a whole and translates it into object code all in one go. Once converted, the object code can be run unassisted at any time. This process is called **compilation**.
- Compilers have several advantages:
- Compiled programs run quickly, since they have already been translated.
- A compiled program can be supplied as an **executable** file. An executable file is a file that is ready to run. Since an executable file cannot be easily modified, programmers prefer to supply executables rather than source code.
- Compilers **optimise** code. Optimised code can run quicker and take up less **memory** space

# Compilers

- Compilers have several disadvantages:
- Because the source code is translated as a whole, there must be enough memory space to hold the source code, the compiler and the generated object code. There also needs to be temporary working space for the compiler to perform the translation. Modern systems either have enough memory or use **virtual memory** to hold all the data.
- Compilers do not usually spot errors - the program has to be compiled and run before errors are encountered. This makes it harder to see where the errors lie.
- The source code must be re-compiled every time the programmer changes the program.
- Source code compiled on one platform will not run on another - the object code is specific to the processor's architecture.

# Interpreters

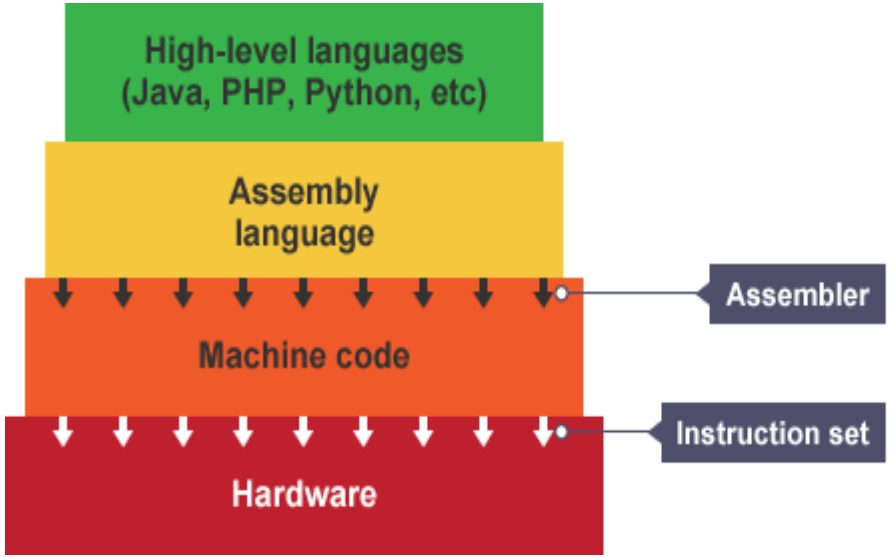
- An interpreter translates source code into object code one **instruction** at a time. It is similar to a human translator translating what a person says into another language, sentence by sentence, as they speak. The resulting object code is then executed immediately. The process is called **interpretation**.
- Interpreters have several advantages:
- Instructions are executed as soon as they are translated.
- Since instructions are executed once translated, they are not stored for later use. As a result, interpreters require less available memory.
- Errors can be quickly spotted - once an error is found, the program stops running and the user is notified at which part of the program the interpretation has failed. This makes interpreters extremely useful when developing programs.

# Interpreters

- Interpreters also have several disadvantages:
- Interpreted programs run more slowly as the processor has to wait for each instruction to be translated before it can be executed.
- Additionally, the program has to be translated every time it is run.
- Interpreters do not produce an executable file that can be distributed. As a result, the source code program has to be supplied, and this could be modified without permission.
- Interpreters do not optimise code - the translated code is executed as it is.

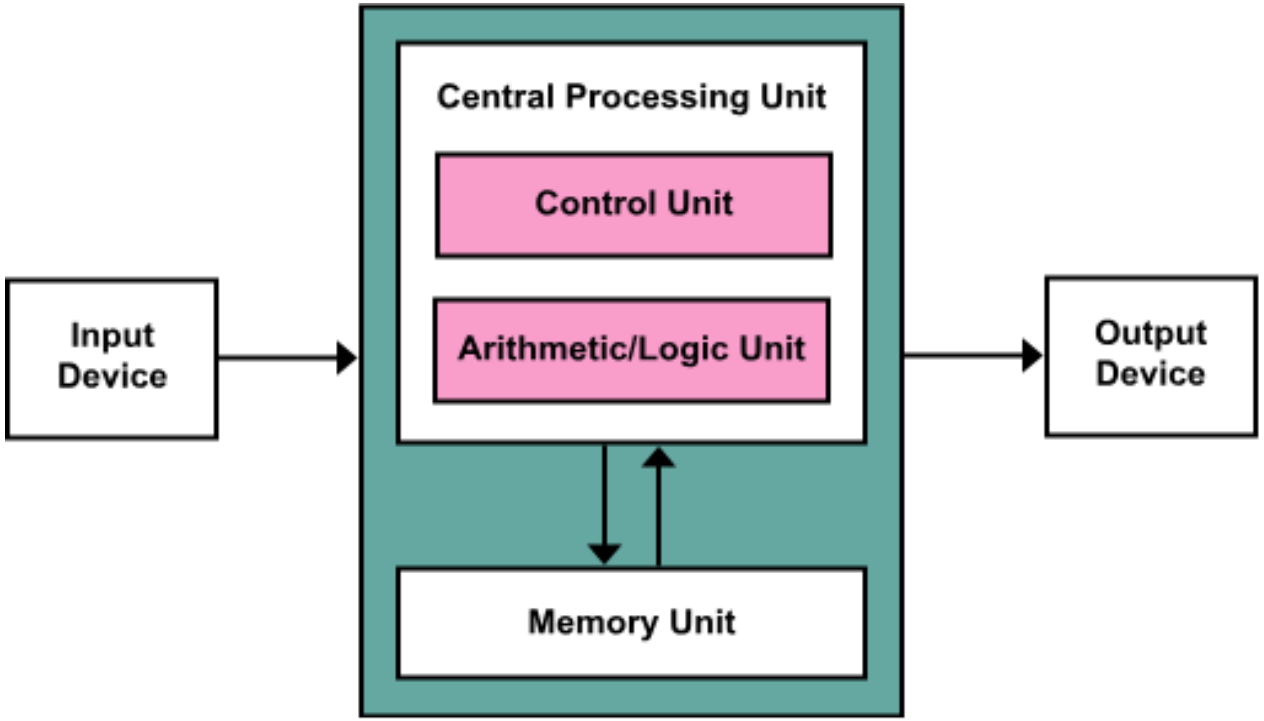
# Assemblers

- Assemblers are a third type of translator. The purpose of an assembler is to translate **assembly language** into object code. Whereas compilers and interpreters generate many machine code instructions for each high level instruction, assemblers create one machine code instruction for each assembly instruction.



# *Programming Instructions*

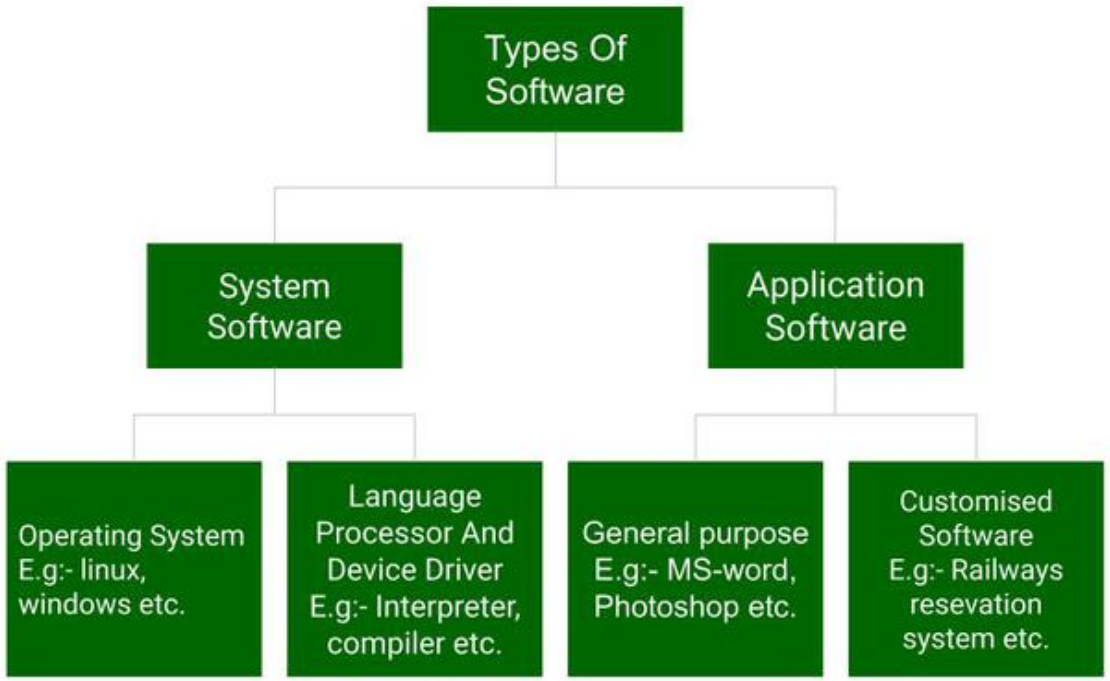
A Computer has two inseparable parts:-
Hardware and Software



# *What is Software & Hardware?*

- **The instructions are the software**
- **The physical component of a computer that are used in the process are hardware**.
- Application S/W:- software that fulfills a specific need or performs tasks
- System S/W:- is designed to run a computer's hardware and provides a platform for applications to run on top of.

# Types of Software



# *Why Number System:-*

- C=a+b
- Display c

a,b and c are numbers

Input as well as output have to be understood by the computer and the user respectively

A number system plays an important role so that the same instructions can be manipulated by giving different numbers as input.

# *Number System:-*

- Way of counting things
- Computer use different types of number systems.
- 1.Decimal(base 10)
- 2.Binary(base 2) 0 1
- 3.Octal(base 8)
- 4.Hexadecimal(base 16)

# *Binary Number System:-*

- Most modern computer systems operate using binary number system.
- The binary number system works like the decimal number system except that it uses 2 as its base and has only two digits 0 and 1
- The weighted values for each position is determined as follows:

| 2^7 | 2^6 | 2^5 | 2^4 | 2^3 | 2^2 | 2^1 | 2^0 | 2^-1 | 2^-2 |
|---|---|---|---|---|---|---|---|---|---|
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | .5 | .25 |

# *Algorithm:*

- ----The word Algorithm means “a process or set of rules to be
- followed in calculations or other problem-solving operations”.
- -----Therefore Algorithm refers to a set of rules/instructions
- that step-by-step define how a work is to be executed upon
- In order to get the expected results.

# *Flowchart:-*

- A flowchart is the graphical or pictorial representation of an algorithm with the help of different symbols, shapes, and arrows to demonstrate a process or a program. With algorithms, we can easily understand a program. The main purpose of using a flowchart is to analyze different methods. Several standard symbols are applied in a flowchart:

- The symbols above represent different parts of a flowchart. The process in a flowchart can be expressed through boxes and arrows with different sizes and colors. In a flowchart, we can easily highlight certain elements and the relationships between each part.

# *Flowchart & Symbols:-*

| | |
|---|---|
| Terminal Box - Start / End |  |
| Input / Output |  |
| Process / Instruction |  |
| Decision |  |
| Connector / Arrow |  |

# *Difference between Algorithm and Flowchart*

| Algorithm | Flowchart |
| --- | --- |
| Algorithm is the step-by-step instruction to solve a specific problem. | Flowchart is a pictorial representation to show the algorithm using geometrical diagrams and symbols. |
| Difficult to understand compared to flowcharts | Easier to understand |
| Complex representation of branching and looping | Easy representation of branching and looping |
| Easy debugging of errors | Difficult debugging of errors |
| Does not follow any rules to write | Has certain predefined rules of construction |

- 
- ***Thank You***