E Content

Of

Computer Fundamentals and IT Tools

For

B.A/B.Sc Computer Application –Semester 1st (CBCS)- Unit - II

University of Jammu

Subject Code: UCATC-101

Created by

Dr. Parveen Singh
Associate Professor

Table of Contents

COMPUTER SOFTWARE

In today's world, the humans have turned his life into a digital life. Computer has become a necessary part of life in this fast world of technology. With the help of Computers, almost every work is being executed through computer softwares. Right from Microsoft Windows as we switch on the computer to the various web browsers, games on the smartphones are the few examples of computer software.

By definition, Software is a collection of data, programs, procedures, instructions, and documentation that perform various predefined tasks on a computer system. They enable users to interact with the computer.

Essentially, computer **software** controls computer **hardware**. These two components are complementary and cannot act independently of one another. It means without **software**, computer **hardware** is useless. For instance, without the help of your web browser software, you will not be able to surf the Internet. Similarly, without an operating system, no application can run on your computer.

Now a days, there are endless number of software types that can be overwhelming for anybody, especially when one does not understand the various types of software and their uses thoroughly.

Types of Software

Broadly, the computer softwares are classified into three types:

1. System Software

System software assists the user and the hardware to function and interact with each other. Basically, it is a programme for controlling the actions of computer hardware to provide basic functionalities that are requested by the user. In simple words, we can say that system software is an intermediator or a middle layer between the user and the hardware. These computer software set a platform for the other softwares to work in. This is the reason why system software is very important in managing the entire computer system. When you first turn on the computer, it is the system software that gets initialized and gets loaded in the memory of the system. The system software runs in the background and is not used by the end-users. This is the reason why system software is also known as 'low-level software'.

Some common system software examples are:

- **Operating System:** An operating system (OS) is the program that, after being initially loaded into the computer by a boot program, manages all of the other application programs in a computer. The application programs make use of the operating system by making requests for services through a defined application program interface (API). In addition, users can interact directly with the operating system through a user interface, such as a command-line interface (CLI) or a graphical UI (GUI). Although each

Operating System is different, most of them provide a Graphical User Interface through which a user can manage the files and folders and perform other tasks. Every device, whether a desktop, laptop or mobile phone requires an operating system to provide the basic functionality to it. As an OS essentially determines how a user interacts with the system, therefore many users prefer to use one specific OS for their device. There are various types of operating system such as real-time, embedded, distributed, multiuser, single-user, internet, mobile, and many more. It is important to consider the hardware specifications before choosing an operating system. Some examples of Operating systems given below:

- Android
- CentOS
- iOS
- Linux
- Mac OS
- MS Windows
- Ubuntu
- Unix

- **Device Drivers:**

It is a type of software that controls particular hardware which is attached to the system. Without the required device driver, the corresponding hardware device fails to work. Device drivers are operating system-specific and hardware-dependent. A device driver acts as a translator between the hardware device and the programs or operating systems that use it. Hardware devices that need a driver to connect to a system include displays, sound cards, printers, mice and hard disks. Further, there are two types of device drivers: Kernel Device Drivers and User Device Driver. Some examples of device drivers are:

- BIOS Driver
- Display Drivers
- Motherboard Drivers
- Printer Drivers
- ROM Drivers
- Sound card Driver
- USB Drivers
- USB Drivers
- VGA Drivers
- VGA Drivers
- Virtual Device Drivers

It is essential that a computer have the correct device drivers for all its parts to keep the system running efficiently.

- **Programming Language Translators:** These are mediator programs on which software programs rely to translate high-level language code to simpler machine-level code. Besides simplifying the code, the translators also do the following :

    - Assign data storage
    - Enlist source code as well as program details
    - Offer diagnostic reports
    - Rectify system errors during the runtime
    - Examples of Programming Language Translators are Interpreter, Compiler and Assemblers.

- **Utility:** Utility software is designed to assist in analyzing, configuring, optimizing and maintaining a computer system. It supports the computer infrastructure. This software works on how an Operating System functions and then accordingly it decides its direction to smoothen the functioning of the system. Softwares like antiviruses, disk cleanup & management tools, compression tools, defragmenters, etc are all utility tools. Some examples of utility tools are:
    - Avast Antivirus
    - Directory Opus
    - McAfee Antivirus
    - Piriform CCleaner
    - Razer Cortex
    - Windows File Explorer
    - WinRAR
    - WinZip

2. Application Software

Application Software, also known as end-user programs or productivity programs are software that helps the user in completing tasks such as doing online research, jotting down notes, setting an alarm, designing graphics, keeping an account log, doing calculations or even playing games. They lie above the system software. Unlike system software, they are used by the end-user and are specific in their functionality or tasks and do the job that they are designed to do. For example, a browser is an application designed specifically for browsing the internet or MS Powerpoint is an application used specifically for making presentations. Application Software or simply apps can also be referred to as non-essential software as their requirement is highly subjective and their absence does not affect the functioning of the system. All the apps that we see on our mobile phones are also examples of Application Software. There is certain software that is exclusively made for app development like Meteor and Flutter. These are examples of Application software too.

There are various types of application software:

- **Word Processors:** These applications for documentation. Along with that it also helps I storing, formatting and printing of these documents. Some examples of word processors are:

    - Abiword
    - Apple iWork- Pages
    - Corel WordPerfect
    - Google Docs
    - MS Word

- **Database Software:** This software is used to create and manage a database. It is also known as the Database Management System or DBMS. They help with the organization of data. Some examples of DBMS are:

    - Clipper
    - dBase
    - FileMaker
    - FoxPro
    - MS Access
    - MySQL

- **Multimedia Software:** It is the software that is able to play, create or record images, audio or video files. They are used for video editing, animation, graphics, and image editing, Some examples of Multimedia Software are:

    - Adobe Photoshop
    - Inkscape
    - Media Monkey
    - Picasa
    - VLC Media Player
    - Windows Media Player
    - Windows Movie Maker

- **Education and Reference Software:** These types of software are specifically designed to facilitate learning on a particular subject. There are various kinds of tutorial software that fall under this category. They are also termed as academic software. Some examples are:

    - Delta Drawing
    - GCompris
    - Jumpstart titles
    - KidPix
    - MindPlay
    - Tux Paint

- **Graphics Software:** As the name suggests, Graphics Software has been devised to work with graphics as it helps the user to edit or make changes in visual data or images. It comprises of picture editors and illustration software. Some examples are:

    - Adobe Photoshop
    - Autodesk Maya
    - Blender
    - Carrara
    - CorelDRAW
    - GIMP
    - Modo
    - PaintShop Pro

- **Web Browsers:** These applications are used to browse the internet. They help the user in locating and retrieving data across the web. Some examples of web browsers are:

    - Google Chrome
    - Internet Explorer
    - Microsoft Edge
    - Mozilla Firefox
    - Opera
    - Safari
    - UC Browser

Other than these, all the software that serves a specific purpose fall under the category of Application Software.

3. Firmware:

Firmware is the permanent software that is embedded into a read-only memory. It is a set of instructions permanently stored on a hardware device. It provides essential information regarding how the device interacts with other hardware. Firmware can be considered as 'semi-permanent' as it remains permanent unless it is updated using a firmware updater. Some examples of firmware are:

- BIOS
- Computer Peripherals
- Consumer Applications
- Embedded Systems
- UEFI

Introduction to Types of Computer Language

The Computer only understands binary numbers that is 0 and 1 to perform various operations but the languages are developed for different types of work on a Computer. The computer language is the main medium to communicate with computers.A Computer Language is a set of instructions that instructs the computer hardware to perform specific tasks. With the development of Technology, several languages are now used to communicate with the computers.

Broadly the computer language can be classified into three categories machine language, assembly language, and high-level language. The machine language is considered as oldest computer language among all three. In machine language, the input is directly given as binary input which is processed by the machine.

Different Types of Computer Language

Below are the top 3 types of computer language:

1. Machine Language

This is one of the most basic low level languages. The language was first developed to interact with the first generation computers. The machine language is sometimes referred to as machine code or object code which is set of binary digits 0 and 1. These binary digits are understood and read by a computer system and interpret it easily. It is considered a native language as it can be directly understood by a central processing unit (CPU). The machine language is not so easy to understand, as the language uses the binary system in which the commands are written in 1 and 0 form which is not easy to interpret. The operating system of the computer system is used to identify the exact machine language used for that particular system.

The operating system defines how the program should write so that it can be converted to machine language and the system takes appropriate action. The computer programs and scripts can also be written in other programming languages like C, C++, and JAVA. However, these languages cannot be directly understood by a computer system so there is a need for a program that can convert these computer programs to machine language. The compiler is used to convert the programs to machine language which can be easily understood by computer systems. The compiler generates the binary file and executable file.

Example of machine language for the text “Hello World”.

01001000 0110101 01101100 01101100 01101111 00100000 01010111 01101111 01110010 01101100 01100100.

2. Assembly Language

The assembly language is considered a low-level language for microprocessors and many other programmable devices. The assembly language is also considered as second-generation language. The first generation language is machine language. The assembly language is mostly famous for writing an operating system and also in writing different desktop applications. The operations carried out by programmers using assembly language are memory management, registry access, and clock cycle operations. The drawback of assembly language is the code cannot be reused and the language is not so easy to understand. The assembly language is considered a group of other languages. It is used to implements the symbolic representation of machine code which is used to program CPU architecture. The other name of assembly language is assembly code. For any processor, the most used programming language is assembly language.

In assembly language, the programmer does the operation which can be directly executed on a central processing unit (CPU). The language has certain drawbacks as it does not contain any variables or functions in programs and also the program is not portable on different processors. The assembly language uses the same structure and commands which machine language does use but it uses names in place of numbers. The operations performed using the assembly language is very fast. The operations are much faster when it is compared to high-level language.

3. High-Level Language

The development of high-level language was done when the programmers face the issue in writing programs as the older language has portability issues which mean the code written in one machine cannot be transferred to other machines. Thus lead to the development of high-level language. The high-level language is easy to understand and the code can be written easily as the programs written are user-friendly in a high-level language. The other advantage of code written in a high-level language is the code is independent of a computer system which means the code can be transferred to other machines. The high-level of language uses the concept of abstraction and also focus on programming language rather than focusing on computer hardware components like register utilization or memory utilization.

The development of higher-level language is done for a programmer to write a human-readable program that can be easily understood by any user. The syntax used and the programming style can be easily understood by humans if it is compared to low-level language. The only requirement in a high-level language is the need of compiler. As the program written in a high-level language is not directly understood by the computer system. Before the execution of high-level programs, it needs to be converted to machine level language. The examples of high-level language are C++, C, JAVA, FORTRAN, Pascal, Perl, Ruby, and Visual Basic.

- **JAVA:** The JAVA programming language is an object- oriented language that is based on objects and classes. The main motto of the development of this language is to make a

computer program run on any system. The JAVA code is machine-independent code means the code needs to be written once and can be executed on any machine. The memory management is done automatically in the java programming language.

- **C:** The C is a procedural and general-purpose programming language used for writing programs. This language is mostly used for writing operating system applications and desktop applications.
- **PASCAL:** The Pascal is a procedural programming language which is based on data structures. It uses the concept of recursive data structures such as graphs, lists, and graphs.

Conclusion

As there is continuous development in the field of the programming language from machine language to low-level language to high-level language the programmers get the maximum benefit as they don't have to write a complex program. The programs can be written easily which can be easily understood by a human. The only need is to convert it into machine language.

TRANSLATORS:

Communication between computer and human being is done by computer language and there are two types of languages i.e. low-level language and High-level language. But the computer only understands the machine language (i.e. binary language in the form of 1's& 0's). Therefore, to execute any program of assembly language and high level language, a translator is required that identify and convert it into machine language.

Thus, *the translator is defined as a software programming tool that transforms the program from one computer language to another one .i.e. from source code into machine code* as shown in figure 1.

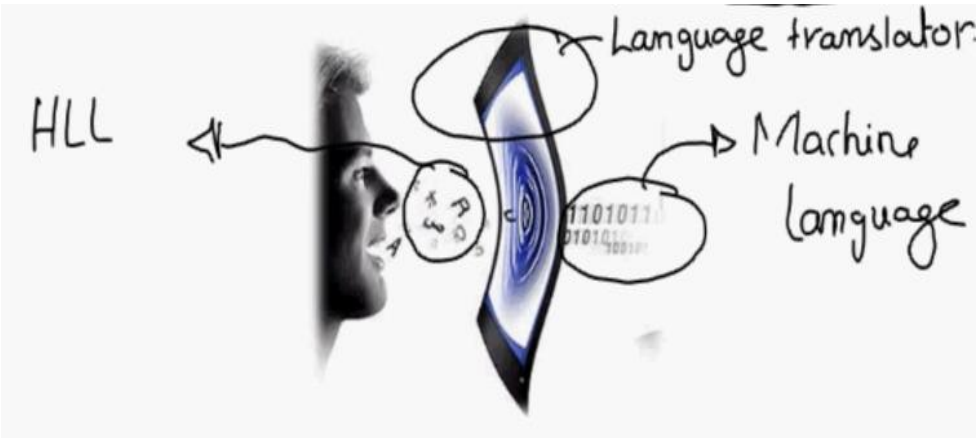

Fig. 1

Each High-level language such as Basic, C, C++, Java, pascal has its own translator and that translator converts instructions of that language into the machine language. Only one type of translator can be used in any high-level language and no language can have more than one translator.

Types of Translators:

1. Compiler.
2. Interpreter.
3. Assembler.

1. **Compiler**:

It is a software that helps to translate high level language programs into machine language as a whole one at a time. It takes source code as input and produces machine language code for the machine on which it is to be executed as output. During the process of translation, the compiler reads the source program as a whole in one attempt and checks for syntax errors and thus its translation is faster than interpreter. It also helps the user to locate the line containing the syntax error & the computer generates a print out of the same. This action is known as diagnostics. The compiler belongs to the high-level language only. During the compilation if there occurs any syntax error, the process of translation is stopped and after removal of that error, the compiler translates the whole program again. It is difficult for the compiler to detect the syntax error as compared to the Interpreter because it compiles the program as a whole.

2. **Interpreter**:

Another way to get machine code to run on your processor is to use an interpreter, which is not the same as a compiler. An interpreter is also a software that converts the source code into the object code line by line and therefore it is slower than the Compiler. The interpreter also belongs to the High-Level language. It also detects the syntax errors at the time of translation but it is easy for the interpreter to detect the error as compared to the compiler. When any syntax error occurs , interpreter stops the translation and after removal of the error, the translation resumes and this is the major difference in between the Interpreter and Compiler.

3. **Assembler**:

An assembler is a type of computer program that interprets software programs written in assembly language into machine language, code and instructions that can be executed by a computer.

An assembler enables software and application developers to access, operate and manage a computer's hardware architecture and components.

An assembler is sometimes referred to as the compiler of assembly language. It also provides the services of an interpreter.

An assembler primarily serves as the bridge between symbolically coded instructions written in assembly language and the computer processor, memory and other computational components. An assembler works by assembling and converting the source code of assembly language into object code or an object file that constitutes a stream of zeros and ones of machine code, which are directly executable by the processor.

Assemblers are classified based on the number of times it takes them to read the source code before translating it; there are both single-pass and multi-pass assemblers. Moreover, some high-end assemblers provide enhanced functionality by enabling the use of control statements, data abstraction services and providing support for object-oriented programming structures.

LINKER:

It is the program that links the object file with some additional library files (also known as Header Files) and creates the executable file having ".exe" extension. This executable file directly runs on the computer. This can be elaborated in the figure 2 below:

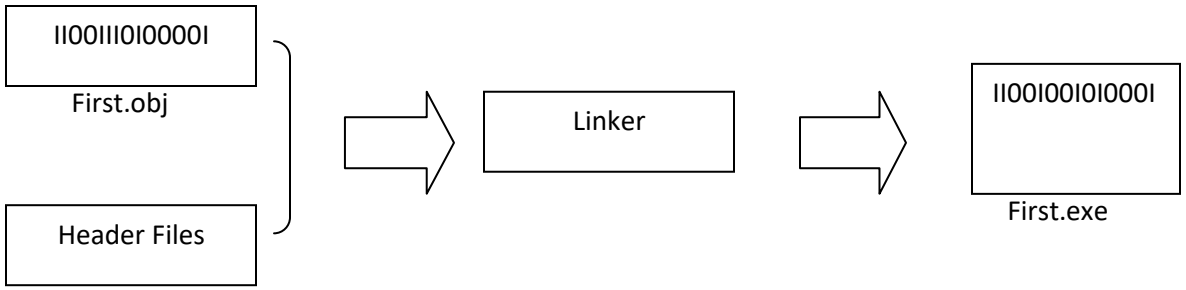

Fig. 2

All header files contain the functionality of the built-in functions that are used in the source program. When the user runs a program, the object file automatically linked with the header files and creates a new executable file. If linker does not find a library of a function then it informs to compiler and then compiler generates an error. The compiler automatically invokes the linker as the last step in compiling a program

LOADER:

Loader is a program that loads machine codes of a program into the system memory. In Computing, a loader is the part of an Operating System that is responsible for loading programs. It is one of the essential stages in the process of starting a program. Because it places programs into memory and prepares them for execution. Loading a program involves reading the contents of executable file into memory. Once loading is complete, the operating system starts the program by passing control to the loaded program code. All operating systems that support program loading have loaders. In many operating systems the loader is permanently resident in memory.

References:

1. https://www.techopedia.com
2. https://www.educba.com.
3. https://www.squareboat.com
4. https://www.tutorialspoint.com
5. https://teachcomputerscience.com/translators/
6. https://www.microcontrollertips.com/compilers-translators-interpreters-assemblers-faq/
7. https://sites.google.com/site/simplestjava/evolution-of-oop.
8. https://www.geeksforgeeks.org/differences-between-procedural-and-object-oriented-programming/