SOFTWARE CONCEPTS

The system software is a type of computer program that is designed to run a computer's hardware and application programs. If we think of the computer system as a layered model, the system software is the interface between the hardware and user applications. The operating system (OS) is the best-known example of system software. The OS manages all the other programs on a computer.

Besides the application software, there is another software called system software. The system software is the operating system. This is very important for the working of the PC.

Example – Windows 98, Windows 95, Windows XP, Solaris, Linux, Unix, Vista, etc.

When a user wants to store any data or program, the data or the program is stored at a location that is known only to the operating system. Therefore, the operating system performs the task of storage management.

Other examples of system software include:

- The BIOS (basic input/output system) gets the computer system started after you turn it on and manages the data flow between the operating system and attached devices such as the hard disk, video adapter, keyboard, mouse, and printer.
- The boot program loads the operating system into the computer's main memory or random-access memory (RAM).
- An assembler takes basic computer instructions and converts them into a pattern of bits that the computer's processor can use to perform its basic operations.
- A device driver controls a particular type of device that is attached to your computers, such as a keyboard or a mouse. The driver program converts the more general input/output

instructions of the operating system to messages that the device type can understand.

Additionally, system software can also include system utilities, such as the disk defragmenter and System Restore, and development tools, such as compilers and debuggers.

System software and application programs are the two main types of computer software. Unlike system software, an application program (often just called an application or app) performs a particular function for the user. Examples include browsers, email clients, word processors, and spreadsheets.

Application Software

Application software is a program or group of programs designed for end-users. These programs are divided into two classes: system software and application software. While system software consists of low-level programs that interact with computers at a basic level, application software resides above system software and includes applications such as database programs, word processors, and spreadsheets. Application software may be bundled with system software or published alone.

Application software may simply be referred to as an application.

Different types of application software include:

- **Application Suite:** Have multiple applications bundled together. Related functions, features, and user interfaces interact with each other.
- **Enterprise Software:** Addresses an organization's needs and data flow in a huge distributed environment
- **Enterprise Infrastructure Software:** Provides capabilities required to support enterprise software systems

- **Information Worker Software: Addresses** individual needs required to manage and create information for individual projects within departments
- **Content Access Software:** Used to access content and addresses a desire for published digital content and entertainment
- **Educational Software:** Provides content intended for use by students
- **Media Development Software:** Addresses individual needs to generate and print electronic media for others to consume

Types of Computer Language

- **Low-Level Languages:** A language that corresponds directly to a specific machine
- **High-Level Languages:** Any language that is independent of the machine

There are also other types of languages, which include;

- **System languages:** These are designed for low-level tasks, like memory and process management
- **Scripting languages:** These tend to be high-level and very powerful
- **Domain-specific languages:** These are only used in very specific contexts
- **Visual languages:** Languages that are not text-based
- **Esoteric languages:** Languages that are jokes or are not intended for serious use

Language is a means of communication. Normally people interact with each other through communication. On the same pattern, communication with computers is carried out through a language. The language is understood both by the user and the machine. Normally every language has its grammatical rules; similarly, every

computer language is bound by rules known as the SYNTAX of the language.

Programming language

A programming language is an artificial language that can be used to write programs that control the behaviour of a machine, particularly a computer.

Programming languages are defined by rules which describe their structure and meaning respectively.

Many programming languages have some form of written specification of their syntax.

There are two levels of language.

1. High-level programming language
2. Low-level programming language

High-level programming language

These languages are normal, English like. Easy to understand statements to pass the instruction to the computer. The languages are problem-oriented. It offers:

- Readability
- Easy Debugging
- Portability
- Easy software Development

Example: – BASIC, COBOL, FORTRAN, PASCAL, and C.

Low-level programming language

Low-level programming languages are sometimes divided into two categories:

I. Machine Language

Machine Language is the only language that is directly understood by the computer. It does not need any translator program. We also call it machine code and it is written as strings of 1's (one) and 0's (zero). When this sequence of codes is fed to the computer, it recognizes the codes and converts it into electrical signals needed to run it. For example, a program instruction may look like this: 1011000111101. It is not an easy language for you to learn because of its difficult to understand. It is efficient for the computer but very inefficient for programmers. It is considered to the first-generation language. It is also difficult to debug the program written in this language.

Advantages and Limitations of Machine Languages

Programs written in machine language can be executed very fast by the computer. This is due to the fact that machine instructions are directly understood by the CPU and no translation of the program is required. But writing a program in machine language has some disadvantages which are given below:

1. Machine dependence: Since the Internal design of a computer varies from machine to machine, the machine language is different from computer to computer. Thus, a program written in machine language in one computer needs modification for its execution on another computer.

2. Difficult to the program: A machine language programmer must have thorough knowledge about the hardware structure of the computer.

3. Error-prone: For writing programs in machine language, a programmer has to remember the OPCODES and has to keep track of the storage locution of data and instructions. In the process, it becomes very difficult for him to concentrate fully on the logic of the problem and as a result, some errors may arise in programming.

4. Difficult to modify: It is very difficult to correct or modify machine language programs.

II. Assembly Language

It uses only letters and symbols. Programming is simpler and less time consuming than machine language programming. It is easy to locate and correct errors in Assembly language. It is also machine-dependent. The programmer must have knowledge of the machine on which the program will run. An assembler is a program that translates an assembly language program into a machine language program.

Assembly languages have the following **advantages** over machine languages;

1. Easier to understand and use: Assembly languages are easier to understand and use because mnemonics are used instead of numeric op-codes and suitable names are used for data.

2. Easy to locate and correct errors: While writing programs in assembly language, fewer errors are made and those that are made arc easier to find and correct because of the use of mnemonics and symbolic names.

3. Easier to modify: Assembly language programs are easier for people to modify than machine language programs. This is mainly because they are easier to understand and hence it is easier to locate, correct, and modifies instructions as and when desired.

4. No worry about addresses: The great advantage of assembly language is that it eliminates worry about address for instructions and data.

Disadvantages

1. Machine Dependence: Programs written in assembly language are designed for the specific make and model of the processor being used and are therefore machine-dependent.

2. Knowledge of hardware is required: Since assembly language is machine-dependent, the programmer must be aware of a particular machine's characteristics and requirements as the program is written. Machine and assembly codes are based on the basic design of computers and are referred to as 'low-level language'.

Compilers

- Besides the application software and the system software, there is a third kind of software called the compiler software.
- A compiler is a system program that translates source code (user-written program) into object code (binary form).
- The whole source code file is compiled in one go and a complete.
- This means that the program can only be executed once the translation is complete.
- It is 5-25 times faster than an interpreter.
- Ex- C & C++ are most popular compiled language.

Interpreter

Translate the high-level language and execute the instruction before passing on to the next instruction.

- An Interpreter is a contrast to a compiler, analyses & executes the source code line- by – line without looking at the entire program.
- First, it translates & executes the first line then it moves to the next line of the source code & repeats the process.
- It is a slow process.
- It is used in the FORTRAN program.
- Ex- JavaScript & VBScript are interpreted language.

INTERPRETER	COMPILER
It translates the program line by line,	It assembles the whole program.
The debugging process is easy.	The debugging process is complex as it generates errors only at the end of the compilation.
The object code of the statement produced by the interpreter is not saved.	The object code produced by the compiler is permanently saved for future reference.
It is a smaller program compared to a compiler. Thus, it occupies lesser memory space and has a lower execution time.	It is a complicated process compared to an interpreter. Thus, it has a higher execution time and occupies larger memory space.
It is a slow process.	It is 5-25 times faster than an interpreter.
It is used in the FORTRAN program.	It is used in C language program.