3
Computer Software

3.1 Introduction

If you want to get something done by a persons, you will tell him what to do in a language that he understands. Similarly, if you want to get some work done by the computer, you have to tell the computer in a language that the computer understands, i.e., machine language. The machine language consists of only binary digits, i.e. 0 and 1. It was felt quite difficult and tedious for human beings to thinks in binary numbers. For communicating with the computer, it was thought that it is advisable to develop a third language, a programming language, that can be understood by both human beings and the computer. Thus a programming language is a set of rules that provides a way of instructing the computer to perform certain operations.

Programming languages are said to be lower or higher, depending on whether they are closer to the language the computer itself uses (lower, which means 0s and 1s) or to the language that people use (higher, which means more English like).

3.2 Objectives

After going through this lesson you will be in a position to:

- identify various levels (or generations) of language such as machine language, assembly language, high-level language

- explain the concept of software
- distinguish between system software and application software
- distinguish between compiler and interpreter
- define operating system and its various functions

3.3 Computer Language

The languages in which programs are written are called programming languages. These languages can be classified into following categories.

- Machine language
- Assembly language
- High level language

Machine Languages

We think of computers as being quite complicated, but actually their basis is very simple. It is based on the concept of electricity being turned "on" and "off". From this on/off, yes/no, two-stage system, sophisticated ways of representing data have been constructed using the binary system of numbers. The binary system is based on two digits 0 and 1.

By contrast, the decimal system that we all use is based on ten digits 0 through 9. The numbers 1, 2 and 3 in the decimal system are represented in the binary system as 1, 10 and 11 respectively. Letters of the alphabet are also represented in binary numbers. In one system, the letter A is represented as 1000001. Commas, semicolons and other special characters are also represented as bunches of 0s and 1s.

In the early days of computers, with machines as the ENIAC, which uses vacuum tubes, one could actually see the tubes lit up or unlit, corresponding to the 1/0 binary state— the switch was either on or off.

In addition, in those early days there was no such things as software. There was only hardware with electrical on/off switches. Whenever a program was to be run, all the switches had to be set - sometimes as many as 6,000 switches for a single program. Then for the next program, the switches had to be reset, a process that might take weeks.

Since those days, machine switches have been replaced by machine programming, programs with statements consisting of 0s and 1s that electrically set the switches, with 0 representing off and 1 responding on. This has made changing from one program to another considerably easier.

Still, programming directly in machine language - the lowest level of programming language, in which instruction is represented as 0s and 1s - is very tedious and time consuming. A programmer must keep track of a tremendous amount of detail. Moreover, he/she must understand the technical operations of the computer. For example, consider a hypothetical line from a program segment, which multiplies two numbers.

```
11110010   01110011   11010010    00010000   01110000  00101011
```

Clearly, working with this kind of code is not for everybody.

Programming in machine code has one advantage over programming at other language levels - its execution is very fast and efficient because the computer can accept the machine code as it is. However, in addition to the complexity involved in working at this level, there is a severe disadvantage to machine language - there is no one standard machine language. Machine language differs from machine to machine.

The languages are machine-dependent, and the programs written in machine language for one computer model will not, in all likelihood, run on a computer of different model. Although many programs or subroutines in machine language for a particular computer is supplied by the manufacturer, still few applications programs are written by users in machine language.

Assembly Languages

From the above discussion it was felt that working with 0s and 1s could turn people themselves into ciphers. In the 1950s, to reduce programming complexity and provide some standardization, assembly languages were developed. Assembly languages, also known as symbolic languages use abbreviations or memonic code - codes more easily memorized to replace the 0s and 1s of machine languages.

The hypothetical machine language segment we saw above is as follows:

```
 11110010   01110011   11010010   00010000   01110000  00101011
```

This could be expressed in assembly language statement as :

PACK 210 (8, 13), 02B (4,7)

Actually, assembly languages do not replace machine languages. In fact, for an assembly language program to be executed, it must be converted to machine code. The assembly language program is referred to as a 'source program' whereas, the machine language program is an 'object program'.

Assembly language code is very similar in form to machine language code. In fact, assembly languages had a one-to-one correspondence which means that 15 assembly statements, would be translated into 15 machine language statements. This one-to-one correspondence was still laborious. However, assembly language instructions (called macro instructions) were devised, which executed batches of one-to-one instructions.

Assembly languages offer several advantages:

- They are more standardized and easier to use than machine languages.
- They operate very efficiently, although not as efficient as the machine languages.
- They are easier to debug.

However, there are still some disadvantages:

- Assembly language programs are usually very long and difficult to debug.
- Though less abstract than machine languages, assembly language programs are still complex.
- Though more standardized than machine languages, assembly languages are still machine dependent.

High Level Languages

High Level Languages helped programmers by reducing further the number of computer operations details they had to specify, so that they could concentrate more on the logic needed to solve the problem.

The following example gives you a code segment in which the same instruction - " Calculate gross pay" - is expressed in three different languages: machine, assembly and COBOL.

Machine Language:

```
11110010   011100011   1101   001000010000   0111   000000101011

11110010   011100011   1101   000100001000   0111   000000101111

11111100   01010010    1101   001000010010   1101   001000011101

11110000   01000101    1101   001000010011   0000   000000111110

11110011   01000011    0111   000001010000   1101   001000010100

10010110   11110000    0111   000001010100
```

Assembly Language

```
PACK   210 (8, 13), 02B (4,7)

PACK   218 (8, 13), 02F (4,7)

MP     212 (G, 13), 21D (3, 13)

SRP    213 (5, 13), 03E (0), 5

UNPK   050 (5, 7), 214 (4, 13)

IO     054 (7), X ‘FO’
```

COBOL

```
MULTIPLY HOURS -WORKED BY PAY-RATE

GIVING GROSS -PAY ROUNDED
```

3.4 Type of High-Level Languages

Languages are often referred to as generations, the idea being that machine languages were the first generation and assembly languages were the second generation. High-level languages are sometimes used to refer all languages above the assembly level. Here we will subdivide high-level languages into three generation.

- Procedural-oriented or third generation
- Problem-oriented or fourth generation
- Natural or fifth generation

Procedural-oriented Languages

High-level languages are often classified according to whether they solve general problems or specific problems. General-purpose programming languages are called procedural languages or third generation languages. They are languages such as Pascal, BASIC, COBOL, and FORTRAN, which are designed to express the logic, the procedure, of a problem. Because of their flexibility, procedural languages are able to solve a variety of problems.

Procedural languages have many advantages over machine and assembly languages:

- The program statements resemble English and hence are easier to work with.
- Because of their English-like nature, less time is required to develop a program for a given problem.
- Once coded, programs are easier to understand and modify.
- The procedural languages are machine-independent.

However, procedure-oriented languages still have some disadvantages compared to machine and assembly languages:

- Programs are executed more slowly.
- The languages use computer resources less efficiently.

Problem-oriented Languages and Application Generators

Third-generation languages, such as BASIC or Pascal, require you to instruct the computer in step-by-step fashion. Fourth-generation languages, also known as problem-oriented languages, are high-level languages designed to solve specific problems or develop specific applications by enabling you to describe what you want rather than step-by-step procedures for getting there.

Fourth- generation languages may be categorized into several kinds of application development tools:

- Personal computer applications software
- Query languages and report generators
- Decision support systems and financial planning languages.
- Application generators

Personal Computer applications software

You may be knowing about various application softwares for PCs, but the ones we are particularly concerned with here are word processors, spreadsheets, database management, business graphics and integrated packages. Knowledge of spread sheet, Foxpro or Power Point can help you develop your own applications.

Query languages and report generators

Query languages allow people who are not programmers to search a database using certain selection commands. Query languages, for example, are used by airline or railway reservation personnel needing ticket information. Report generators are designed for people needing to prepare reports easily and quickly. Examples of query languages and report generators include QBE, SQL, RPG, etc.

Decision support systems and financial planning languages

Decision support systems are interactive software designed to help managers make decisions. Financial planning languages are particular kinds of decision support systems that are employed for mathematical, statistical and forecasting modelling. Both types of languages find applications in developing complicated business model hypothetical representations of management problems.

Application generators

An application generator consists of a software system with number of program modules, preprogrammed for various functions, so that the programmer or user can simply state which function is needed for a particular application, and the system will select the appropriate module and run a program to meet the user's needs.

Table 3.1 summarizes some of the major difference between third-generation languages (3GLs) and fourth-generation languages (4GLs.).

Table 3.1

Difference between 3 GLs and 4 GLs

Third-generation Languages	Fourth-generation Languages
Intended for use by professional programmers.	May be used by a non-programming personnel as well as a professional programmer.
Requires specification of how to perform task.	Requires specification of what task is to be performed (system determines how to perform the task).
All alternatives must be specified.	Default alternatives are built in; an end user need not specify these alternatives
Require large number of procedural instructions.	Require fewer instructions.
Code may be difficult to read, understand and maintain.	Code is easy to understand and maintain because of English-like commands.
Language developed for batch operations.	Language developed primarily for on-line use.
Can be difficult to learn.	Easy to learn.
Difficult to debug.	Easy to debug.
Typically file-oriented.	Typically database-oriented.

Natural Languages

Natural languages are still in the developmental stages, but they promise to have profound effect, particularly in the areas of artificial intelligence and export systems. Natural languages have two characteristics:

- They are designed to make the connections that humans have with computers more natural - more humanlike.
- They are designed to allow the computer to become "smarter" - to actually simulate the learning process by remembering and improving upon earlier information.

Two popular natural languages are LISP and PROLOG.

3.5 Compilers and Interpreters

For a high-level language to work on the computer it must be translated into machine language. There are two kinds of translators - compilers and interpreters. High-level languages are called either compiled languages or interpreted languages.

In a compiled language, a translation program is run to convert the high-level language program, which is called the source code, into a machine language code. This translation process is called compilation.

The machine language code is called the object code and can be saved and either run (executed) immediately or later. Some of the most widely used compiled languages are COBOL, C, C ++, FORTRAN, etc.

In an interpreted language, a translation program converts each program statement into machine code just before the program statement is to be executed. Translation and execution occur immediately, one after another, one statement at a time.

Unlike the compiled language, no object code is stored and there is no compilation. The most frequently used interpreted language is BASIC.

Compiled languages are better than interpreted languages as they can be executed faster and more efficiently once the object code has been obtained. On the other hand, interpreted languages do not need to create object code and so are usually easier to develop and test.

Intext Questions 3.1

1. What is machine language?
2. Describe briefly the advantages of high level language.
3. Write True or False for the following statement:
   (a) Machine language is machine-dependent.
   (b) Assembly language is second generation language.
   (c) A program written in a high level language is called a object program.
   (d) Portability is one of the features of high level language.
   (e) Query languages allow people who are not programmers to search a database using certain selection commands.

3.6 What is Software

As we know what computer is a machine which cannot do anything without instructions from the user. In order to do any specific job you have to give a sequence of instructions to the computer. This set of instructions in a proper sequence is called a computer program. Software refers to the set of computer programs that cause the hardware (computer system) to function in the desired manner. Hardware means physical components of the computer system. We can say that hardware which cannot perform any calculation, comparison or manipulation without being instructed to do so. These instructions play a vital role in the performance of a computer. A complete set of instructions written to solve a problem on a computer is called software.

3.7 Types of Software

Computer software is normally classified into two broad categories:

(i) System Software

(ii) Application Software

3.7.1 System Software

System software includes general programs written for a computer. It consists of pre-written programs and documentation supplied by the manufacturer along with the computer. These programs are held permanently in the machine. The primary objectives of this software are to:

(i) enhance the efficiency of hardware utilization, and

(ii) make computers simple to use.

The first objective is realized by that component of system software which is commonly known as operating system.

System Software is a set of instruction to the machine to interpret and execute application software, for example, language translators (called compilers and interpreters), operating systems, utilities and special purpose software.

Language Translator

A language translator is a system software which translates a computer program written by a user into a machine understandable form.

The most elemental form of programming uses only the binary digits 0 and 1, which is directly understood by the electronic circuits. A program that have only binary digits is called a machine language program. It is difficult to write or understand machine language programs, as these consist of 0s and 1s.

Assembly language provides a significant improvement over machine language, Assembly language programs are written using memonic codes like ADD, STORE, etc., rather than their machine language representations in binary digits. Therefore, programming in assembly language is easier. For its execution, it needs to be translated into machine language code. This translation is done by an assembler. Both machine language and assembly language programs are machine dependant.

A program written in a high level language needs to be translated into machine language code before execution. This translation is done either through a complier or through an interpreter. A compiler is a translator program which reads an entire program written in a high level language and converts it into machine language code.

An interpreter on the other hand, is a translator which converts one statement of the program into machine code, execute it and then goes on to perform the same for the next statement, and it continues doing so till the end of the program or on error occurrence.

Operating System

An operating system is the most important system software and is a must to operate a computer system. An operating system manages a computer's resources very effectively, takes care of scheduling multiple jobs for execution and manages the flow of data and instructions between the input/output units and the main memory.

The first operating system, called batch processing operating system was developed for the second generation computers. This operating system executes jobs serially one after another from a batch of jobs submitted for execution. The central processing unit is kept busy only during the processing cycle of a job and remains idle during the input and output operations. A multi programming opening system handle multiple jobs simultaneously by overlapping the input/ output and processing cycles of various jobs.

Other types of operating system which are popular today are ‘multi-processing operating systems’ and ‘real time operating systems’. A multi-processing operating system uses multiple CPUs to process multiple jobs. A real time operating system is a type of interactive operating system which strict time limitation. Receiving and processing data is done quickly enough to produce output, to control, direct or effect the outcome of ongoing activity. The reservation system used by railway, airlines, hotel are examples of real time applications.

The function of an operating system can be compared to the functions of a principal in a school. The principal of the school provides an environment in which the other employees can do useful work. The principal will do this by allocating resources available such as allocating time for each period, allocating classes to teachers and providing rooms for different classes etc. Similarly, an operating system governs the working of the computer and input/output devices. The operating system programs act as an interface between the user's programs and the computer's components and help in the execution of user's programs. The major functions of an operating system are:

(i) User identification and keeping of the resources used by the users. Thus un-authorized users cannot use the computer.

(ii) Sharing of computer resources among many users. The sharing is achieved by permitting simultaneous executions of more than one user program. This is usually called multi-programming. A mix of programs can keep the whole memory occupied, all devices active, and the control unit and ALU constantly busy, thus increasing utilization of hardware.

(iii) Executive batches of programs, one after another, without human intervention.

(iv) Protection of user's data and programs.

(v) Controlling the transfer of data and programs between the main memory and secondary storage and other I/O devices.

(vi) Providing programs to select appropriate translators.

(vii) Providing facilities to detect and correct errors in a user's program.

An operating system understands a fixed set of commands. This set of commands is often called job control language (JCL). The JCL commands are used by the computer users to indicate their requirements to the operating system. The operating system which is used with a micro-computer is called CP/M (control program for microprocessor). Another operating system which is gaining popularity and which is available on a variety of different machines is UNIX (UNIX was developed by Bell laboratories). DOS is an operating system commonly used in PCs.

The second objective of simplifying computer usage is achieved by enabling the users to write their own programs in languages other than the machine language.

The only language understood by computers is machine language. If any other language is used, the programs must first be translated into the machine language before they can be executed. One of the most significant aspect of computers is that such translations can be made automatically using other programs. A computer manufacturer provides many programs for the translation of many languages into the machine language. These programs are called translators or compliers and form part of the system software.

We rarely talk about computer hardware alone. It is the hardware and software both which make up a computer system. There is definitely some substance in the argument that system software is as important, if not more, as its hardware.

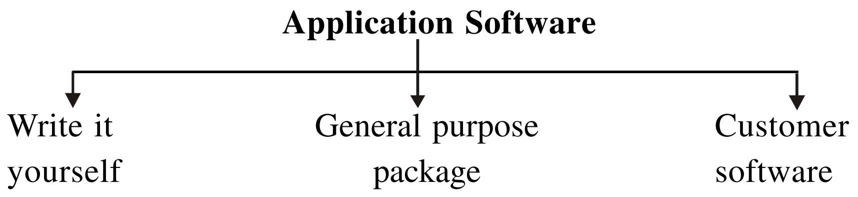
Utility Software

Utility software may be considered as a system software which is used quite often in the development of a program. Sort merge programs are used to sort records in a particular manner or sequence. Such programs are normally provided by the manufacturers. The programmer can also develop his own utility software and keep it in the secondary memory of the computer.

3.7.2 Application Software

Application software is written to perform a specific task or process, such as accounting, payroll, mailing list, result preparation and so on. The majority of application software is written in high-level languages.

Assuming that the task to be carried out has been correctly identified, carefully defined, the prospective user will come across the following alternative sources for this application software.



(a) **Write it yourself** : The program written by the user can be the most satisfactory solution. It will be an exact match to the needs of the business. The program can grow with the business.

(b) **General purpose application packages** : Application packages refer to a set of computer programs, which have been written to perform specific, commonly required tasks. Each program is written in such a way that it is applicable to a large number of users. The main advantage is that it is relatively cheap as cost of the package is spread over a number of customers. The major disadvantage of application package is that it is not likely to fulfil all the requirements of the prospective users.

(c) **Customer software** : It refers to computer programs specially written to match the exact needs of the user. It is precisely the same as getting one's clothes stitched from a tailor to fit exactly rather than buying a ready-made dress. The most important advantage is that such software fulfils all the needs of the customer. The major disadvantage is that customer software costs much more than general purpose application software, because the package is specially made for one particular customer.

Common Application Packages

Some of the common requirements of the users of personal computers have been identified and common applications packages have been developed to meet their needs. These packages include word processor, database processing, spreadsheet calculations, mail-merge, presentations and communications (email). These packages have been prepared so that they are simple to use. They also provide graphical user interface to make them very user friendly. These packages are readily available in the market and one can purchase them, install it on his/her computer easily and start using the package.

Intext Questions 3.2

1. Define software.

2. Differentiate between system software and application software.

3. Write True or False for the following statements.

   (a) A program written in a high level language needs to be translated into machine language code before execution.

(b) An operating system is a must to operate a computer system.

(c) JCL stands for Job Common Language.

(d) Application software is written to perform a specific task or process.

(e) Utility software cannot be considered as an application software.

3.8 What you have learnt

In this lesson, you learnt about various levels of language and their advantages and disadvantages briefly. Importance of two types of software, namely, system software and application software has been explained in a simplified manner so that it could be easily understood by you.

3.9 Terminal Questions

1. Differentiate between hardware and software?
2. What are the advantages and disadvantages of machine language?
3. What is the difference between source program and object program?
4. Explain assembly language. What are its advantages over machine language?
5. Define operating system. Write down its various functions.
6. Describe briefly about application software.

3.10 Feedback to In-Text Questions

In-Text Questions 3.1

1. Machine language consists of sequence of instructions written in the form of binary numbers (1s and 0s) to which the computer responds directly. Computer understands this language only.
2. Advantages of high level language are:

(a) Readability

(b) Easy learning

(c) Easy debugging

(d) Easy software development

3. (a) True (b) True (c) False (d) True (e) True

In-Text Questions 3.2

1. Software refers to the set of computer programs that cause the hardware to function in the desired manner. Software is a general term that is used to describe any single program or a group of programs.

2. Application software is a set of programs to carry out operations for a specific application. System software is a set of programs written for performing tasks such as controlling all operations required to move data into or out of the computer.

3. (a) True (b) True (c) False (d) True (e) False