MODULE 4 PROGRAMMING THE COMPUTER

UNIT 1 COMPUTER LANGUAGES

CONTENTS

7.0 INTRODUCTION

In this unit, we shall take a look at computer programming with emphasis on:

a. The overview of computer programming languages.
b. Evolutionary trends of computer programming languages.
c. Programming computers in a Beginner All-Purpose Symbolic Instruction Code (BASIC) language environment.

8.0 OBJECTIVES

At the end of this unit you should be able to provide background information about programming the computer.

3.0 MAIN CONTENT

3.1 An Overview of Computer Programming Languages

Basically, human beings cannot speak or write in computer language, and since computers cannot speak or write in human language, an intermediate language had to be devised to allow people to communicate with the computers. These intermediate languages, known as programming languages, allow a computer programmer to direct the activities of the computer. These languages are structured around a unique set of rules that dictate exactly how a programmer should direct the computer to perform a specific task. With the powers of reasoning and logic of human beings, there is the capability to accept an instruction and understand it in many different forms. Since a computer must be programmed to respond to specific instructions, instructions cannot be given in just any form. Programming languages standardise the instruction process. The rules of a particular language tell the programmer how the individual instructions must be structured and what sequence of words and symbols must be used to form an instruction.

- An operation code
- Some operands.

The operation code tells the computer what to do such as add, subtract, multiply and divide. The operands tell the computer the data items involved in the operations. The operands in an instruction may consist of the actual data that the computer may use to perform an operation, or the storage address of data. Consider for example the instruction: a = b
+ 5. The '=' and '+' are operation codes while 'a', 'b' and '5' are operands. The 'a' and 'b' are storage addresses of actual data while '5' is an actual data.

Some computers use many types of operation codes in their instruction format and may provide several methods for doing the same thing. Other computers use fewer operation codes, but have the capacity to perform more than one operation with a single instruction. There are four basic types of instructions, namely:

(a) input-output instructions;
(b) arithmetic instructions; (c) branching instructions; and (d) logic instructions.

An input instruction directs the computer to accept data from a specific input device and store it in a specific location in the store. An output instruction tells the computer to move a piece of data from a computer storage location and record it on the output medium.

All of the basic arithmetic operations can be performed by the computer. Since arithmetic operations involve at least two numbers, an arithmetic operation must include at least two operands.

Branch instructions cause the computer to alter the sequence of execution of instruction within the program. There are two basic types of branch instructions; namely unconditional branch instruction and conditional branch instruction. An unconditional branch instruction or statement will cause the computer to branch to a statement regardless of the existing conditions. A conditional branch statement will cause the computer to branch to a statement only when certain conditions exist.

Logic instructions allow the computer to change the sequence of execution of instruction, depending on conditions built into the program by the programmer. Typical logic operations include: shift, compare and test.

3.2 Types of Programming Language

The effective utilisation and control of a computer system is primarily through the software of the system. We note that there are different types of software that can be used to direct the computer system. System software directs the internal operations of the computer, and applications software allows the programmer to use the computer to solve user made problems. The development of programming techniques has become as important to the advancement of computer science as the developments in hardware technology. More sophisticated programming techniques and a wider variety of programming languages have enabled computers to be used in an increasing number of applications.

Programming languages, the primary means of human-computer communication, have evolved from early stages where programmers entered instructions into the computer in a language similar to that used in the application. Computer programming languages can be classified into the following categories:

(a) Machine language
(b) Assembly language
(c) High level symbolic language
(d) Very high level symbolic language

3.2.1 Machine Language

The earliest forms of computer programming were carried out by using languages that were structured according to the computer stored data, that is, in a binary number system. Programmers had to construct programs that used instructions written in binary notation 1

and 0. Writing programs in this fashion is tedious, time-consuming and susceptible to errors.

Each instruction in a machine language program consists, as mentioned before, of two parts namely: operation code and operands. An added difficulty in machine language programming is that the operands of an instruction must tell the computer the storage address of the data to be processed. The programmer must designate storage locations for both instructions and data as part of the programming process. Furthermore, the programmer has to know the location of every switch and register that will be used in executing the program, and must control their functions by means of instructions in the program.

A machine language program allows the programmer to take advantage of all the features and capabilities of the computer system for which it was designed. It is also capable of producing the most efficient program as far as storage requirements and operating speeds are concerned. Few programmers today write applications programs in machine language. A machine language is computer dependent. Thus, an IBM machine language will not run on NCR machine, DEC machine or ICL machine. A machine language is the First Generation (computer) Language (IGL).

3.2.2 Assembly (Low Level) Language

Since machine language programming proved to be a difficult and tedious task, a symbolic way of expressing machine language instructions is devised. In assembly language, the operation code is expressed as a combination of letters rather than binary numbers, sometimes called mnemonics. This allows the programmer to remember the operations codes easily than when expressed strictly as binary numbers. The storage address or location of the operands is expressed as a symbol rather than the actual numeric address. After the computer has read the program, operations software are used to establish the actual locations for each piece of data used by the program. The most popular assembly language is the IBM Assembly Language.

Because the computer understands and executes only machine language programs, the assembly language program must be translated into a machine language. This is accomplished by using a system software program called an assembler. The assembler accepts an assembly language program and produces a machine language program that the computer can actually execute. The schematic diagram of the translation process of the assembly language into the machine language is shown in fig.9.1. Although, assembly language programming offers an improvement over machine language programming, it is still an arduous task, requiring the programmer to write programs based on particular computer operation codes. An assembly language program developed and run on IBM computers would fail to run on ICL computers. Consequently, the portability of computer programs in a computer installation to another computer installation which houses different makes or types of computers were not

possible. The low level languages are, generally, described as Second Generation (Computer) Language (2GL).

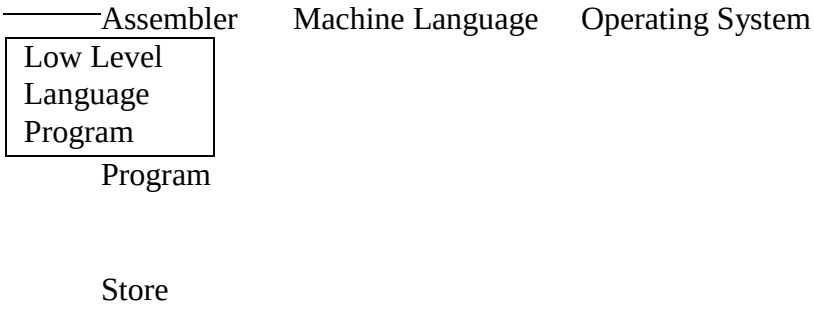

Fig.11: The assembly language program translation process

3.2.3 High Level Language

The difficulty of programming and the time required to program computers in assembly languages and machine languages led to the development of high-level languages. The symbolic languages, sometimes referred to as problem oriented languages reflect the type of problem being solved rather than the computer being used to solve it. Machine and assembly language programming is machine dependent but high level languages are machine independent, that is, a high-level language program can be run on a variety of computers.

While the flexibility of high level languages is greater than that of machine and assembly languages, there are close restrictions in exactly how instructions are to be formulated and written. Only a specific set of numbers, letters, and special characters may be used to write a high level program and special rules must be observed for punctuation. High level language instructions do resemble English language statements and the mathematical symbols used in ordinary mathematics. Among the existing and popular high level programming languages are Fortran, Basic, Cobol, Pascal, Algol, Ada and P1/1. The schematic diagram of the translation process of a high level language into the machine language is shown in fig.9.2. The high level languages are, generally, described as Third Generation (Computer) Language (3GL).

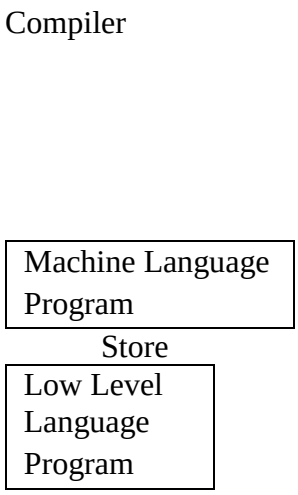

Fig. 12: The high level language program translation process

The general procedure for the compilation of a computer program coded in any high level language is conceptualised in Fig. 13.

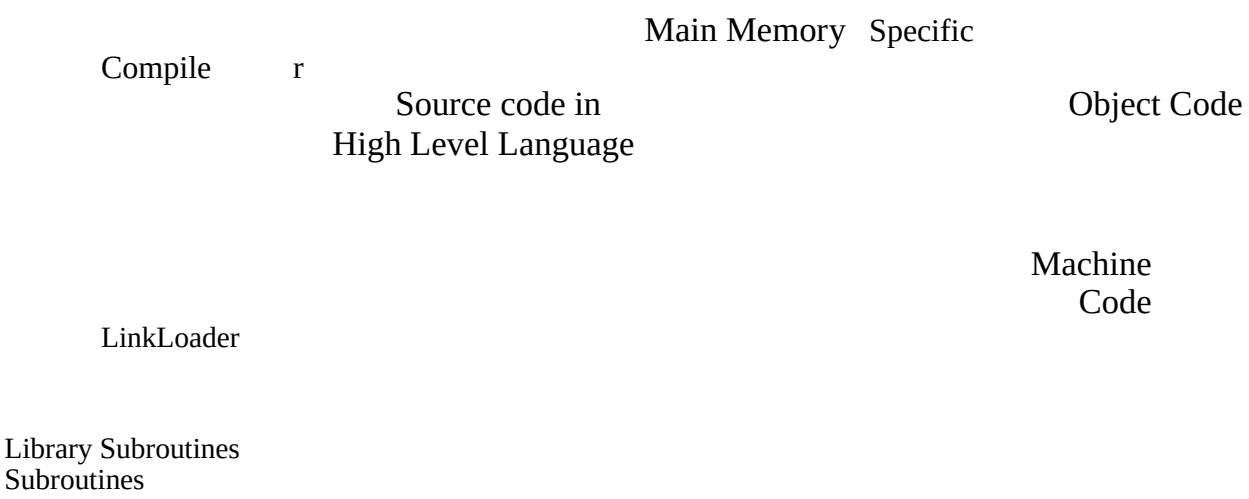

Fig. 13: General procedure for compiling a high level language program

3.2.4 Very High Level Language

Programming aids or programming tools are provided to help programmers do their programming work more easily. Examples of programming tools are:

- Program development systems that help users to learn programming, and to program in a powerful high level language. Using a computer screen (monitor) and keyboard under the direction of an interactive computer program, users are helped to construct application programs.
- A program generator or application generator that assists computer users to write their own programs by expanding simple statements into program code'.
- A database management system
- Debuggers that are programs that help the computer user to locate errors (bugs) in the application programs they write.

The very high level language generally described as the Fourth Generation (computer) Language (4GL), is an ill-defined term that refers to software intended to help computer users or computer programmers to develop their own application programs more quickly and cheaply. A 4GL, by using a menu system for example, allows users to specify what they require, rather than describe the procedures by which these requirements are met. The detailed procedure by which the requirements are met is done by the 4GL software which is transparent to the users.

A 4GL offers the user an English-like set of commands and simple control structures in which to specify general data processing or numerical operations. A program is translated into a conventional highlevel language such as COBOL, which is passed to a compiler. A 4GL is, therefore, a non-procedural language. The program flows are not designed by the programmer but by the fourth generation software itself. Each user request is for a result rather than a procedure to obtain the result. The conceptual diagram of the translation process of very high level language to machine language is given Fig.14.

The 4GL arose partly in response to the applications backlog. A great deal of programming time is spent maintaining and improving old programs rather than building new ones. Many organisations, therefore, have a backlog of applications waiting to be developed. 4GL, by stepping up the process of application design and by making it easier for end-users to build their own programs, helps to reduce the backlog.

Fig.
14

:
The
program translation process

GL
4
Program

Compiler

High Level

Language Program

GL T
4
ranslator

Machine Language

Program

Operating

System

4.0 CONCLUSION

Computer programming languages are means by which programmers manipulate the computer. The programming languages emanate from the need to program the computer in languages that would be easy for nonexperts to understand and to reduce the enormity of tasks involved in writing programs in machine code. Programming languages have

evolved from the machine language to assembly language, high level language and very high level programming language.

5.0 SUMMARY

We summarise the study of computer programming language as follows:

- Machine language is the binary language and it is made up of only 0s and 1s which represent the 'off' and 'on' stages of a computer's electrical circuits.

- Assembly language has a one-to-one relationship with machine language, but uses symbols and mnemonics for particular items. Assembly language, like machine language, is hardware specific, and is translated into machine language by an assembler.

- High level languages are usable on different machines and are designed for similar applications rather than similar hardware. They are procedural in that they describe the logical procedures needed to achieve a particular result. High level languages are translated into machine language by a compiler or an interpreter.

- In a high level language one specifies the logical procedures that have to be performed to achieve a result. In a fourth generation language, one needs to simply define the result one wants, and the requisite program instructions will be generated by the fourth generation software. Fourth generation languages are used in fourth generation systems in which a number of development tools are integrated in one environment.

6.0 TUTOR-MARKED ASSIGNMENT

1. What are computer programming languages?
2. Explain the following terms: machine language, source code, assembler, and compiler.

7.0 REFERENCES/FURTHER READING

Akinyokun, O.C. (1999). *Principles and Practice of Computing Technology*. Ibadan: International Publishers Limited.

Balogun, V.F., Daramola, O.A. Obe, O.O. Ojokoh, B.A., and Oluwadare S.A. (2006). *Introduction to Computing: A Practical Approach.* Akure: Tom- Ray Publications.

Francis Scheid (1983). *Schaum's Outline Series: Computers and Programming*. Singapore: McGraw-Hill Book Company.

Chuley, J.C. (1987). *Introduction to Low Level Programming for Microprocessors*. Macmillan Education Ltd.

Holmes, B.J. (1989). *Basic Programming*, (3rd ed.) ELBS.