7

COMPUTER LANGUAGES

Language is a means of communication. Computer languages enable the end user to communicate with the computer system. With the help of computer languages, the programmer can tell a computer what he wants to do. Programming language is the combination of all the characters, symbols and the usage rules that permit people to communicate with the computer. Just as a natural language has a systematic method of using symbols (e.g., grammar in English, Malayalam, Hindi, etc.), certain rules are there in computer languages also. These rules are called syntax rules, which tells us what is to be done and how it is to be done. Computer, being an artificial creation, is not able to deduce the correct meaning from an incorrect instruction. As such, the programmer must use the exact syntax rules and even the punctuation marks must be correct. On the basis of syntax and usage of character, computer languages are divided into five categories. They are:

1. Machine Language

Machine language is the language understood by the computer. It is the fundamental language of a computer and is called the machine code of the computer. It consists of strings of binary numbers. It is the only one language, which is understood by the central processor. The instruction prepared in the machine language involves two parts such as command or operation code or 'opcode' and the 'operand'.

The opcode tells the computer what function it is to perform. The operand tells the processor where to find or store the data or other instructions, which are to be manipulated. Each instruction inform the central processor, of the function to be performed and the length and location of data fields in the operation. A program to add two numbers can be expressed in the machine language as follows:

0010000001100111001

0011000001000010001

2. Assembly Language

This computer language was popularized during 1959-69, and is the low-level first generation computer language. In assembly language, operation codes are represented by mnemonic codes. Mnemonics refers to any type of mental trick, which helps us to remember. Mnemonic instruction to add two numbers can be represented as:

AP WKGRDTOT, MPLYANS

Here AP mnemonic names the machine operations, and WKGRDTOT names a particular register in the CPU and MPLYANS represents a storage location containing the result of manipulation.

Symbolic addresses are used to represent storage location. The mnemonic codes and symbolic address are translated into machine operation codes (opcodes) and machine storage addresses (operands) by the assembler.

A program of instruction written by the programmer in assembly language is called source program. When it is translated into machine language, it is called object program. Adding two numbers and printing the result could be written in assembler language as:

```
CLA  A
ADD  B
STA  C
TYP  C
HLT
```

It means: take A from memory, add B to it, and store it in C, and then type C and halt. The translation process of an assembler can be represented as follows:

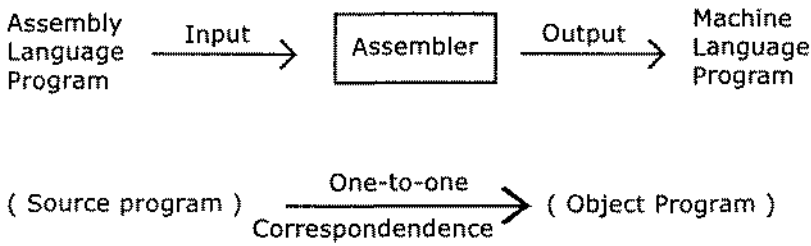

Fig. 7.1: Conversion of assembly language to machine language

The language which substitutes the letter and symbols for number in machine language is called assembly language or symbolic language. A program written in assembly language is known as assembly codes or symbolic program. Each brand of computer has its own unique assembler language suited to the machine architecture. The following example shows certain opcodes and their corresponding mnemonics in assembler language.

Mnemonic	*Opcode*	*Meaning*
HLT	00	Halt, (to stop the program)
CLA	10	Clear and add to A register
ADD	14	Add the contents of A register
SUB	15	Subtract from contents of A register
STA	30	Store A register

Initially, one assembly language instruction will be translated into exactly one machine language instruction. But, in actual practice, to perform a specific operation, more than one instruction in the machine language is required. In order to solve this problem, a new development was made in the assembly language, which came to be known as macroinstruction. In Macroinstruction, a particular assembly language instruction is translated into a number of machine language instructions so that the complete set of instructions for the performance of that particular activity is available to the central processor. The development of macroinstructions paved the way for designing machine-independent higher level languages.

3. High Level Language (HLL)

A high level language is popularly called procedure-oriented language, which specifies the step-by-step procedure

to be followed by a programmer, while performing an operation. It is oriented towards a specific class of processing problems. High-level languages are developed as a boon to the programmers because they enable them to use the computer without knowing its internal structure in detail. It facilitates the writing of program instructions in English words and mathematical symbols, so that the programmer can concentrate on the logic of the problem to be solved, instead of thinking about the programming details. The advantage is that, with the help of one instruction, the programmer can solve the entire operation, which otherwise requires at least three instructions to complete. If we want to add two numbers ERST and SCND, and the sum is to be stored in ANSR, we can perform these three activities with the help of only one instruction in the high level language. It can be written as:

ANSR = ERST + SCND

Basically, high-level languages are symbolic languages using English words as well as mathematical symbols. In other words, HLL is a symbolic language with macroinstructions. The problem-oriented high level language, when used for business applications, it is called business-oriented language. If such languages applied to perform sophisticated computations, they are called mathematically-oriented languages. High-level languages are of two types; data processing languages to perform the processing of huge volume of data, and algorithmic languages to perform mathematical computations and to make algorithms. The program written in a high level language is translated into machine langauge is (which the computer can understand) by the compilers. Compiler is a translating program, which transform the instructions in HLL into machine language. It is so called because it compiles a set of machine language instructions for each instruction of a high level language. The translation process can be illustrated as follows:

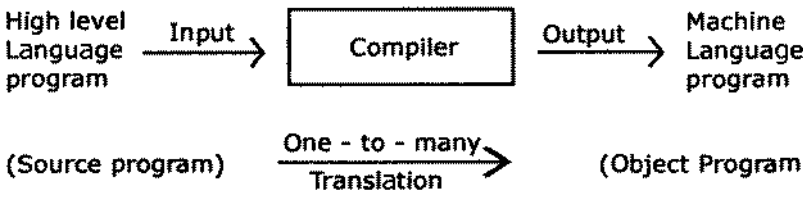

Fig. 7.2: Conversion of highlevel language to machine language.

A separate compiler is required for each high level language. That is, only a COBOL compiler can transform source programs written in COBOL.

Another important translator used for converting instructions in high-level languages to machine language instructions is the interpreter. Interpreter translates one instruction in high-level language into machine code, which is immediately executed. In other words, the interpreter translates one instruction, the control unit executes the machine code of that instruction, and then the next instruction is translated and the control unit executes the corresponding machine code, and the like. The difference between a compiler and interpreter is that, in a compiler, the machine codes corresponding to a high level language instruction are not executed, while in an interpreter, it is executed each time the translation occurs. Interpreters are normally used in microcomputers. Translators like assembler, compiler, and interpreter are also called language processors, since they process a specific language. More than 180 high level languages are currently in use. Some of them are:

1. FORTRAN (FORmula TRANslator): It is the first standardized high-level language, developed in 1957. It is designed for solving scientific and engineering problems and for mathematical computations and algorithms. It is an algebra based programming language. A number of commands like READ, WRITE, DO, STOP, etc. is used in FORTRAN, with direct meaning. It requires a certain part of a statement to be placed in certain columns.

Statement number is optional and is placed in columns 1–5. The comment statement starts with 'C' in the first column. Atypical FORTRAN program to print the sum of 20 numbers is given below:

```
C       FORTRAN PROGRAM TO COMPUTE THE SUM
C       OF 10 NUMBERS
        SUM = 0
        DO 501 = 1, 20
        READ (5, 10) N
10      FORMAT (F6.2)
        SUM = SUM + N
50      CONTINUE
        WRITE (6,20) SUM
20      FORMAT (1x,' THE SUM OF GIVEN NUMBERS = F10.2)
        STOP
        END.
```

2. BASIC (Beginner's All-Purpose Symbolic Instruction Code): BASIC was developed in the USA in 1964 by John Kemny and Thomas Kurtz. It is a popular interactive language useful in the fields of education, personal computing, etc. It is an interpreter-based language since interpreters are used to convert BASIC instructions to machine language. The advantage is that the language is simple, flexible, and powerful and easy to understand.

BASIC can be used in both business and scientific applications. The only problem associated with it is that it has no standardized format. A BASIC program to compute the sum of 20 numbers is:

```
5     REM PROGRAM TO COMPUTE
6     REM THE SUM OF 20 NUMBERS
10    LET S = 0
20    FOR I = 1 to 10
30    READ N
40    LET S = S + N
50    NEXT I
60    PRINT " THE SUM OF GIVEN NUMBERS" =, S
70    DATA 4,20,15,32,48,40,25,30,35,10
80    DATA 12,3,9,14,44,15,10,20,30,35
90        END.
```

3. COBOL (Common Business-Oriented Language): It is a language developed and designed in 1959-60, specifically for business applications. The language has the structure of a business report in English and is constructed from sentences, paragraphs, sections and divisions. Each program in COBOL has four divisions, such as Identification Division, Environment Division, Data Division and Procedure Division. A COBOL program to compute and print the sum of certain number is given below:

```
IDENTIFICATION DIVISION
PROGRAM-ID SUMUP
AUTHOR. PK SINHA
*     THIS PROGRAM COMPUTES AND PRINTS
*     THE SUM OF GIVEN NUMBERS.
ENVIRONMENT DIVISION
CONFIGURATION SECTION.
```

```
SOURCE-COMPUTER.  BURROUGHS-6700
OBJECT-COMPUTER.  BURROUGHS-6700
INPUT - OUTPUT SECTION
FILE CONTROL
     SELECT DATA - FILE ASSIGN TO
     CARD-READER.
     SELECT OUTPUT - FILE ASSIGN TO PRINTER
DATA DIVISION
     FILE SECTION.
     FD DATA-FILE.
        RECORD CONTAINS 80 CHARACTERS
        LABEL RECORD IS OMITTED
DATA RECORD IS INPUT - DATA - RECORD.
01   INPUT - DATA - RECORD
     05 N PICTURE 9 (6) V99.
     0S FILLER PICTURE X (72)
FD   OUTPUT - FILE
     RECORD CONTAINS 132 CHARACTERS
     LABEL RECORD IS OMITTED
     DATA - RECORD IS OUTPUT - RECORD.
01   OUTPUT - RECORD.
     05 FILLER PICTURE X
     05 TITLE PICTURE X (25)
     0S SUM PICTURE 9 (10) V99.
     0S FILLER PICTURE X (94)
WORKING - STORAGE SECTION
77   MESSAGE PICTURE X (25) VALUE IS
     "THE SUM OF GIVEN NUMBERS" =.
PROCEDURE DIVISION
OPEN - FILES.
     OPEN INPUT DATA - FILE.
     OPEN OUTPUT  OUTPUT - FILE
INITIALIZATION.
     MOVE SPACES TO OUTPUT RECORD.
     MOVE ZERO TO SUM.
```

```
PROCESS - LOOP
        READ DATA - FILE AT END GO TO
        PRINT - PARA.
        ADD N TO SUM
        GO TO PROCESS LOOP.
PRINT - PARA
        MOVE MESSAGE TO THE TITLE.
        WRITE OUTPUT - RECORD.
END - OF - JOB.
        CLOSE DATA - FILE
        CLOSE OUTPUT FILE
        STOP RUN.
```

4. RPG (Report Program Generator): It is also a business-oriented programming language, designed to generate reports of business applications. It was developed in 1961 by the IBM. The language is commonly used in small business computers. It cannot be used for scientific applications since it has limited mathematical capability.

5. PASCAL (Named in honor of Blaise Pascal): It was the first language, which introduced the concept of structured programming. In PASCAL, the programs are written in logical modules and inturn transferred to the controlling module. Looping process is carried out automatically by special loop control statements. PASCAL can be used for both scientific and file processing application. A PASCAL program to compute and print the sum of 20 numbers is given below:

```
PROGRAM SUMNUMS (INUT, OUTPUT);
(* PROGRAM TO COMPUTE THE SUM OF 10 NUMBERS *)
(* DECLARATION OF VARIABLES*)
VAR SUM, N : REAL;
VAR1 : INTEGER;
BEGIN
SUM : = 0;
FOR 1 : = 1 TO 20 DO
BEGIN
READ (N)
SUM: = SUM + N;
END;
```

```
WRITELN (THE SUM OF GIVEN NUMBERS = SUM)
END.
```

6. JAVA: It is the most popular object-oriented programming language today.

7. C, C++: It is a general-purpose language allowing the manipulation of internal processor registers.

8. PROLOG (PROgramming LOGic): It is designed to handle complex logical operations and is an artificial intelligence language.

9. SNOBOL (StriNg Oriented SymBOlic Language): This language is designed to manipulate texts and strings of characters and to perform string comparison, splitting of a string, combining two strings, etc.

10. PILOT (Programmed Inquiry Learning or Teaching)

11. ADA (Named in honor of lady Augusta Ada Lovelace)

12. APL (A Programming Language)

13. ALGOL (Algorithmic Language)

14. PL /1 (Programming Language / " One ")

15. LISP (LISt Programming)

16. Logo

17. APT (Automatically Programmed Tooling)

18. FORTH

19. Module – 2

4. Fourth–generation Language (4GL)

The application development tools designed after the development of high level languages (Third generation languages) are together referred to as the Fourth-Generation Languages. They interact with DBMS to store, manipulate and retrieve data. Unlike high-level languages, which are procedural languages, AGLS are non–procedural or declarative languages, which allow the user to specify the output, without explaining the details of data manipulation to produce the result. Generally, the fourth generation languages include two application tools like End–user–oriented 4GLS and 4GLS oriented to DBMS packages. The 4GLS include SQL as their core product. They provide formatting feature to produce

quality reports, and also means of quick application development like Application By–Form Languages like Micro Soft Excel, Visual Basic (VB) for applications, HTML (Hyper Text Markup Languages), etc. are examples of 4 GLS.

Program generators: They prepare application programs from specifications of display terminals, interactive dialogs and processing function to be carried out. With the help of a program generator, the programmer can escape from the task of writing detailed programming procedures for a dialog to be displayed on the Visual Display Terminal: They can use the screen dialog layout to provide specifications to the program generator which automatically generates the program instructions.

Report generators: Report generators help the programmer to describe the format of the report and features of the data. The detailed procedures of formatting a report are generated by the software. All complex activities in report writing, like page breaks, page heading on first and subsequent–pages, page numbering, grand totals etc., are performed by the report generator, by following regular rules. An example of a report generator on small computers is RPG (Report Program Generator).

5. Fifth–Generation Languages (5 GL)

These include Experts Systems and Artificial Intelligence, which are intended to anticipate the needs and requirements of end–users, and to execute the commands accordingly. They are referred to as Knowledge Based System (KBS), which go beyond the Decision Support Systems and use AI and ES tool to deduce decision influences on the basis of codified knowledge. Examples of such Expert Systems are PROSPECTOR [ ] For mineral exploration, xcon [ ] for configuring VAX Computer, DENDRAL [ ] etc. Expert System Shells [ ] help researchers to build just the knowledge base and use the same for various applications.

Documentation

It refers to the process of writing explanation for a program in the form of remarks or comments. It is divided into two categories, such as:

1. Technical documentation: Meant for the programmer who tries to modify the source program and

includes information about the formula used, data and programs copied from other programs, etc.

2. User level documentation: Which helps the user to understand technical information about the program and to use the program.

Debugging and Testing

Debugging refers to the identification, removal and correction of run-time errors, mistakes, or bugs in the execution of a program.

Testing refers to the checking of correctness of a computer solution to a specific problem under consideration. The debugging and testing functions are performed by the compilers.

Conclusion

Spectacular developments are taking place in the computing and technology fields, which leads to a change in the concept of MIS as a passive data support tool to a modern weapon for providing new business opportunities capable of competing in technologically advanced economy. The role of computers in MIS has also changed from an abstract concept to a concrete system, which provides insight competing advantage and avenues to leverage business. The traditional file processing systems, their latest incarnation DBMS, various computer languages, etc. constitute the primary vehicle for the development of MIS.

Exercise

Short Answer Questions

1. Describe machine language.
2. What is an assembly language?
3. Briefly explain high level languages.
4. Explain the significance of computer languages.
5. What is FORTRAN?
6. Describe BASIC language.
7. Describe the significance of COBOL as a business application language.
8. What is RPG?
9. Explain the use of PASCAL.
10. What do you mean by fourth generation languages (4 GL)?

11. Explain the concepts of program generators and report generators.
12. Describe fifth generation languages and its importance.
13. What do you mean by documentation?
14. Explain the concepts of debugging and testing.
15. Write short notes on assemblers and compilers.

Essay Questions

1. Explain various computer languages and their applications.
2. Briefly explain important high level languages in computer.
3. Explain the functions of program generators and report generators.
4. Discuss the recent development in computer languages.
5. Discuss COBOL language and programming in COBOL.
6. Explain the application of FORTRN in scientific and engineering fields.
7. Write a BASIC program to find the product of 10 numbers.
8. Describe JAVA, C++, PROLOG and LISP languages.
9. Explain fourth and fifth generation languages.
10. What are programming languages and explain their applications.