

**Chapter  
2**

**PROGRAMS AND  
PROGRAMMING  
LANGUAGES**

*Get on the  
Fast Track!*



TM

**SYS-ED/  
Computer  
Education  
Techniques, Inc.**

**Objectives**

You will learn:

- Instructions and data.
- Program components.
- Types of programming languages.
- Language translation and compilation.
- Features and capabilities of machine languages.
- High-level programming languages.

---

## 1 Programs

In data processing a series of instructions comprising a complete procedure is designated as a program.

A program is stored on an electronic magnetic storage medium directly accessible and controlled by the operating system.

A stored program usually consists of at least two parts:

1. An operation component for reading, writing, adding, subtracting, comparing, or moving data.
2. At least one operand which designates the location(s) of the information to be used for the specified operation.

During an instruction cycle, an instruction is selected from storage and analyzed by the central processing unit. The operation part indicates the operation to be performed. This information is stored and named in order that it has specific meaning for the computer.

### **Examples:**

"5A" is for register add

"59" is for compare

"5C" is for multiply

The operand defines or augments the function of the operation. In order to perform an arithmetic operation, the storage locations need to be designated.

For input or output, the unit to be used is specified.

---

## 1.1 Two-address Instructions

On a IBM System/370, instructions have two address portions.

Depending on the function of the instruction, the two addresses can indicate a device to be used and the data to be operated on, or two factors of data to be processed. The output unit to be used would be included in the other address.

---

## 1.2 Instructions and Data

The only distinction between instructions and data in main storage is the point in time when they are brought into an instruction cycle.

- When brought in at the beginning of the cycle, it is interpreted as an instruction.
- When brought in during any other type of cycle, it is considered to be data.

---

## **2 Program Components**

In order to prepare even a simple program for execution, some or all of the following resources will need to be allocated along with programming language-specific decisions that will need to be made:

- Allocation of storage locations to data, instructions, and related information.
- Availability of reference data, such as tables, files, or constant factors.
- Requirements for accuracy, and methods of checking and auditing.
- Ability to restart the system in case of unscheduled interruptions or error conditions.
- Automatic monitoring of the system to ascertain that the required input and output devices are connected and available for operation.
- Format of output data with provisions, if required, for later conversion to cards, printed reports, or displayed reports.
- Availability of program routines that have been used and tested in other procedures and that may be used to advantage in the current procedure.
- Editing of data with provision to record exceptions that cannot be processed.

---

### **3 Programming Languages**

A computer will execute instructions that have been presented to it in machine language.

Even with the use of hexadecimal equivalents, coding a program in machine language can be difficult, time consuming, and costly. The initial coding of a program will typically contain numerous errors. Subsequent corrections may well introduce other errors.

---

#### **3.1 Symbolic Language**

Symbolic languages permit the programmer to write convenient equivalents of machine instructions using mnemonics to represent them.

Symbolic instruction representations include:

- A for add
- S for subtract
- D for divide
- ST for store
- B for branch

The first generation of programming languages utilized a one-for-one translation. Macro instructions were then developed. A single program language instruction was used to produce a series of machine instructions. This development greatly increased the power of programming languages.

Each final machine language instruction is assigned a particular location in main storage. The location of each instruction must be known precisely. It is, in effect, the "name" of the instruction.

For example, if the L instruction is to be assigned a location of 1000, which is its precise location in main storage, and the add instruction is to immediately follow it, then the location of the add instruction would need to be 1004. This is because the load instruction is a four-byte instruction.

If an additional instruction is to be inserted in a program of many instructions, every instruction from the point of insertion must have its previously assigned location changed. Since most programs undergo changing or updating, instruction location assignment is a tedious and necessary part of programming.

The solution was the development of higher level programming languages in which the translating program selects the desired location of the first instruction, and succeeding instructions are assigned sequentially ascending locations

---

### **3.2 Language Translation or Compiler**

There are two major components comprising a programming language:

1. The language itself, with the associated rules of grammar.
2. A machine language program which translates the language into machine language.

The input to a translator is called the source program.

The output from the processor is the object program; which is also known as the object module.

The object program will not be in executable format; it will need to be processed by a program called the Linkage Editor which will produce a load program or load module, together with tables and other constant factors. There may also be requirements specific to the language where the computer operating system provides resources for execution of the program.

---

**4 Programming Languages - Legacy****COBOL**

Common Business Oriented Language

The language written by a COBOL programmer bears little resemblance to machine language, and the programmer will in most situations not have to be directly concerned with the method by which the COBOL language program is translated into machine language.

COBOL utilizes business English.

**FORTRAN**

Formula Translator

FORTRAN is based upon a mathematical business language.

**PL/1**

PL/1 was developed to meet the need for a broad-based language for utilization in both business and scientific applications.

**RPG**

Report Program Generator

RPG utilizes simplified control forms to streamline the file definition and input/output control required by other programming languages.

---

## **5 Machine Language - First Generation**

Machine languages were developed in the modern era of information technology and were closely integrated with the computer hardware and proprietary operating systems.

A machine language instruction includes an operation code and operand. Since the computer stores each instruction in binary form, in absolute terms, a machine-language instruction consists of a string of zeros and ones. Octal, hexadecimal, or binary coded decimal equivalents are commonly used when coding programs.

A one-to-one relationship exists between the instructions coded by the programmer and the performance of operations by the computer. A complete list of instructions is known as an object program.

An object program can be executed by a computer without being translated; it is machine-understandable. The instructions are keyed, loaded into main storage, and will control subsequent processing steps.

From a programmer's perspective, there are disadvantages in using a machine language.

- The coding of a machine-language program will be tedious.
- It will be necessary to account for and keep track of both addresses and numeric operation codes.
- When an error is discovered in the results produced by the program, it will be necessary to keep examine each instruction in order to ascertain where the error occurred.
- Existing previously written programs will not readily available for reuse when coding a new program.
- It will be necessary to have a comprehensive knowledge of the computer and operating system on which the program is to be executed.

---

## **5.1 Assembly Language**

An Assembler language provides an easier way for generating machine commands. Each operator and data item can have a symbolic name. Most assembler commands are translated into a single machine operation code. For example, an 'A' can reference an add operator.

Advantages associated with Assembler include its efficiency and the access it has to computer resources. The shortcomings include a challenging learning curve, higher maintenance cost, and longer development time.

Assembler is sometimes referred to ALC: Assembler Language Coding or BAL: Basic Assembler Language.

---

**6 High-level Programming Languages**

High level programming languages are not closely related to machine instructions.

Depending on the high-level language used, the assembler is replaced by a compiler or generator.

The advantages associated with a high-level programming language include:

- Easier to learn than machine or assembly languages.
- Reduced amount of time required for coding programs.
- The languages are machine-independent; programs written in a high-level language can be executed on a wide range of computers with minimal alteration.
- The structure of the language makes documentation easier and there typically will be documentation tools and facilities included in the operating environment.
- Maintenance and updating of code can be performed without extensive rewriting of affected code.

There are two major shortcomings with a high-level language.

High-level-language programs usually take longer to execute than a comparable low-level language programs. The translation process is extensive and generates many machine-language instructions.

Some machine operations that a computer can not be implemented in a high-level language.

---

## **6.1 FORTRAN**

In the mid 1950's, a group was organized to create a computer language that would lend itself to arithmetic computation. The group was headed by John Backus of IBM. By 1956, they had developed a language known as FORTRAN.

The original FORTRAN language was designed to address a variety of numerical computation requirements and complex formulas with many variables. It permitted a variable to have up to three independent subscripts.

FORTRAN has evolved into an algebraic and algorithmic scientific-mathematical language.

In situations where large volumes of input and output must be dealt with or extensive file maintenance is required, FORTRAN would not be the optimal high level programming language.

---

## 6.2 COBOL

Up through the late 1950's, no computer language well suited to business applications had been developed. The federal government of the United States, through the Department of Defense, decided to sponsor a committee to develop a language which would address the operational challenges of a business environment.

A committee was formed to study the needs of business and develop an appropriate computer language. The committee became known as CODASYL: Conference of Data Systems Languages. By December 1959, the original specifications for COBOL were completed and distributed by the Government Printing Office in 1960.

In order to encourage full compliance with the project, the Department of Defense announced that no lease or purchase contracts would be made for computers that did not possess meet the standards for the COBOL compiler.

In 1968, the American National Standard - ANS - COBOL was released. In 1974 and 1985 revised ANS COBOL standards were released.

COBOL is considered to be machine-independent. It utilizes a business world terminology and is written in Englishlike terms. Accordingly, it tends to be self-documenting.

The COBOL compiler provides diagnostics and error messages that makes program debugging easier; it is well suited for the input and output operations commonly associated with business operations. In relation to machine and Assembler languages, COBOL is relatively easy to learn.

The structure of the COBOL language is comparable to a physical book. The basic element of a COBOL program is the sentence. The next level in the structural hierarchy of the COBOL language is the paragraph. It is followed by sections and divisions.

The four divisions in a COBOL program are:

- IDENTIFICATION
- ENVIRONMENT
- DATA DIVISION
- PROCEDURE DIVISION