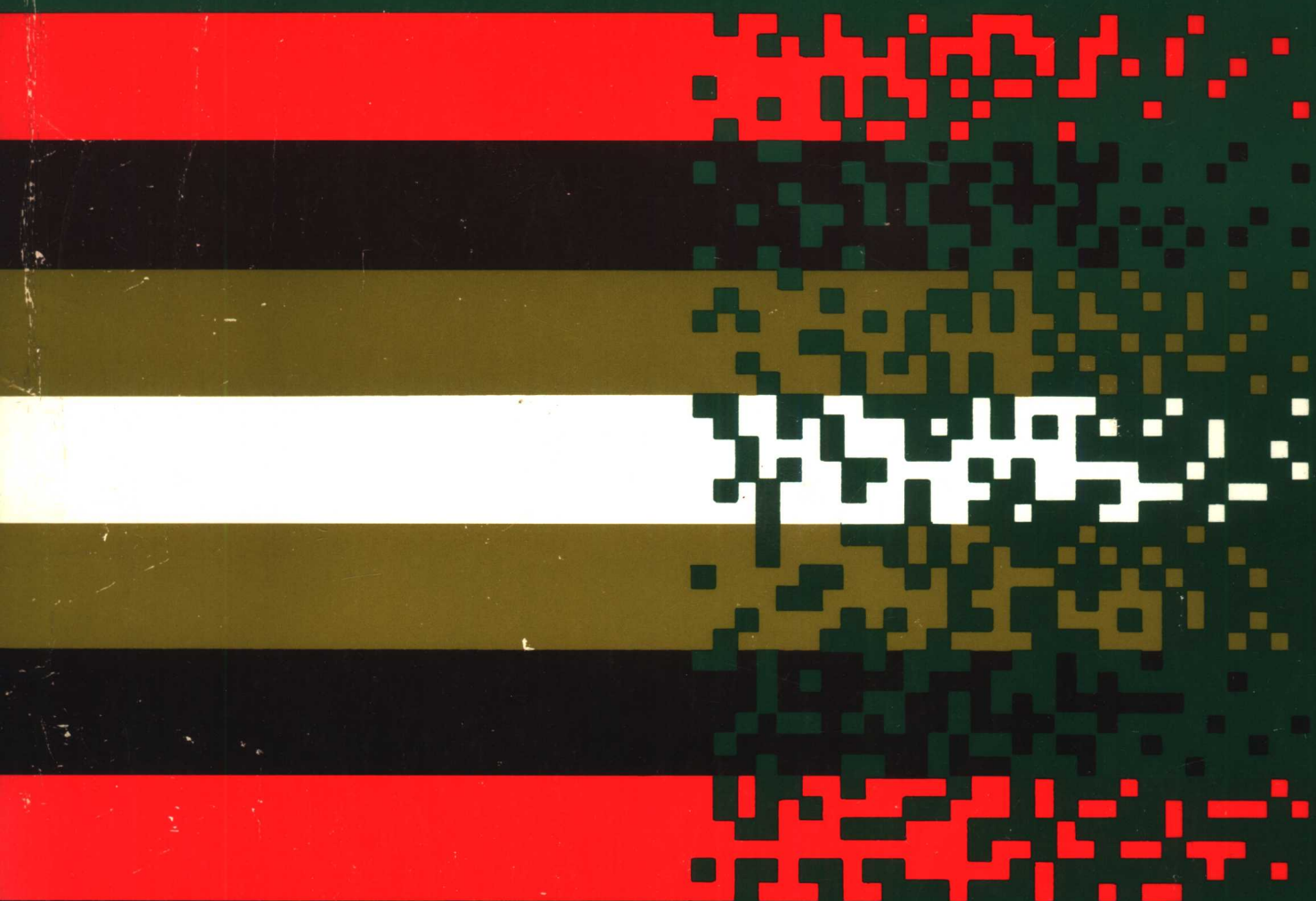
# Fundamentals of Structured COBOL

ROBERT C. NICKERSON



# Fundamentals of Structured COBOL

**Robert C. Nickerson**

San Francisco State University

**Little, Brown and Company**

Boston Toronto

**Library of Congress Cataloging in Publication Data**

Nickerson, Robert C., 1946-
Fundamentals of structured COBOL.

Includes index.
1. COBOL (Computer program language) 2. Structured programming. I. Title.
QA76.73.C25N524 1984 001.64'24 83-19528
ISBN 0-316-60648-0



Library of Congress Catalog Card No. 83-19528

ISBN 0-316-60648-0

9 8 7 6 5 4 3 2 1

MU

Published simultaneously in Canada by Little, Brown & Company (Canada) Limited

Printed in the United States of America

The previous edition of this book was entitled *COBOL Programming: A Structured Approach*

*Acknowledgment*

# Preface

The objective of this book is to provide a thorough, carefully paced introduction to computer programming in the COBOL language with an emphasis on structured programming. To accomplish this objective, the book systematically introduces the elements of COBOL as they are needed for various processing situations. Structured programming concepts are developed along with, not separate from, language features. As a result, the reader not only learns the COBOL language but also gains an understanding of the need for each language element and a grasp of how to develop well-structured programs in COBOL.

The version of COBOL that is described is 1974 ANS COBOL. The most widely used features of this version of the language are covered in detail. Although some of the less common features are not explained in the book, Appendix A gives the syntax of the complete language. In addition, Appendix B describes some of the changes that are included in the proposed new ANSI standard COBOL. The book can be used with any computer that supports 1974 ANS COBOL, including most minicomputers and microcomputers.

Structured programming concepts are incorporated into the book from the beginning. Starting with the first example in Chapter 1, all sample programs follow a structured approach. Basic control structures are discussed early in the book; Chapter 4 is devoted to decision logic and Chapter 5 covers loop control. Modular programming is also introduced early (Chapter 5). Top-down program design is described in detail in Chapter 8 and a complete example is given. This chapter also covers other aspects of program development, including program testing and documentation. Program structure, style, and understandability are emphasized throughout the book. Structure charts and pseudocode are discussed at the appropriate points to help explain program organization and logic. Upon completion of the book, the reader should be able to develop COBOL programs that are well structured, understandable, and correct.

The book is organized into two parts. The first part, consisting of Chapters 1 through 8, introduces the basic features of COBOL, develops fundamental programming methodology, and describes essential program logic. These chapters discuss COBOL elements and program logic for basic input and output, numeric data processing, program control, data organization, and report output. In addition, the main ideas of structured programming are presented. Chapter 8 can be thought of as a capstone for this part of the book because it brings together and explains in detail many concepts about program development. Chapter 8 is also a transition to the second part of the book, which consists of Chapters 9 through 14. These chapters discuss advanced COBOL features and program logic. Among the advanced topics covered are data validation, interactive input and output, control break logic, one-, two-, and three-level table processing including the use of the SEARCH statement, sequential file processing including file updating and sorting, and indexed file processing. With some exceptions, the material in Chapters 9 through 14 can be read in any order.

A large number of programming problems are provided at the end of the book. The problems are arranged into three application groups (sales analysis, inventory processing, and customer order processing). Problems may be selected entirely from one group or from different groups. The first approach makes it easier to complete

more problems because many of them involve modification of previous programs. This approach also emphasizes the importance of good program design for ease of program maintenance. The second approach results in a greater variety of applications being explored. The chapter prerequisite is given for each problem. Test data is provided for most problems.*

The book covers all of the important topics in the Data Processing Management Association's (DPMA) course CIS-2, Application Program Development I, plus additional topics. The book can be used in a one- or two-semester (quarter) course at two- and four-year colleges. In addition, the book can be used in courses at specialized computer schools, in company training programs, and for self-study.

Many features make the book especially useful. These include the following:

- The first section of Chapter 1 contains a brief discussion of essential computer concepts. This can be used as a review by readers who have had previous exposure to these topics, or as an introduction for those readers who have no prerequisite background.
- Both terminal (interactive) and punched card program processing are explained in Chapter 1. This facilitates the use of the book with all types of computers, including microcomputers.
- The book is designed so that programming can begin as early as possible. After finishing Chapter 2, the reader can write simple but complete programs of his or her own design. After each succeeding chapter, increasingly complex programs can be prepared.
- Many examples are provided throughout the book. There is at least one complete program in each chapter, and many chapters have several complete examples. All complete programs were tested on a computer using a 1974 ANS COBOL compiler, and sample input and output are shown for each program. Many partial programs are also used to help explain the concepts.
- One common application (sales analysis) is used throughout the book. Because of this, the reader does not have to learn a new application with each example. Alternative applications are introduced where they are most useful.
- The book discusses a number of system topics that the programmer needs to know. Among these topics are record layout, report design, file organization, interactive input/output design, table usage, system flowcharts, file usage, and system controls. These topics are integrated into the book at the points where they are most appropriate.
- Questions at the end of each chapter review the material covered in the chapter. The answers to approximately half of the review questions are found in Appendix G.
- Flowcharts are discussed in Appendix C. This location allows the topic to be covered at the most appropriate time. All flowcharts are keyed to examples in the book.

A separate student workbook is available. The workbook contains additional material to assist the student in understanding and applying the chapter topics. Included are format summaries, a glossary of terms, additional review questions and answers, and short coding problems with solutions.

An instructor's manual is available that contains teaching suggestions, course schedules, chapter objectives, lists of terms, a complete set of lecture notes, answers to end-of-chapter review questions, and test questions and answers. Also available is a set of transparency masters. Approximately half of the transparency masters are figures from the book; the other half are new illustrations. By using the lecture

* All names of persons, companies, and organizations in examples, problems, and questions in this book are fictitious and are used for illustrative purposes only.

notes and transparencies made from the masters, a complete course in COBOL programming can be taught.

The ideas for a book such as this always come from numerous sources. I am grateful to the many professors, writers, colleagues, and students who have contributed in some way to this book. My special appreciation goes to Gary Hammerstrom who read much of the manuscript and provided many excellent suggestions. I also appreciate the time spent by Sultan Bhimjee in helping me to clarify my thinking on some topics. The manuscript reviewers did an excellent job, and their comments were especially useful. I would like to thank Henry Etlinger, Jerome Myers, David Presser, Kathy Timmer, Murray Berkowitz, William Moloney, and Victor Shtern for their participation in the reviewing. Many of their suggestions have been incorporated into the book.

Finally, I would like to thank my family for their support and help during the sometimes trying writing process.

# Contents

# Chapter 1

# Introduction to COBOL programming

A computer is a device that is used to solve problems. The process that a person goes through in instructing a computer how to solve a problem is called programming. Programming involves combining words and symbols that are part of a special language. COBOL is a language that is commonly used for programming solutions to business problems.

This book is about programming in the COBOL language. The book describes the main rules of COBOL and explains the general process of computer programming. It also presents and explains many programming examples for common business problems. As a result, you will not only learn the fundamentals of the COBOL language but also gain an understanding of the programming process and an insight into common business computer applications.

In this chapter we introduce the basic concepts necessary to begin studying COBOL. The first section reviews elementary computer concepts. We then present the basics of the COBOL language and describe the general process of programming in COBOL. After completing this chapter you will have the background needed to begin learning to program in COBOL. Later chapters will go into detail about the COBOL language, the programming process, and business computer applications.

## 1–1 Computer concepts

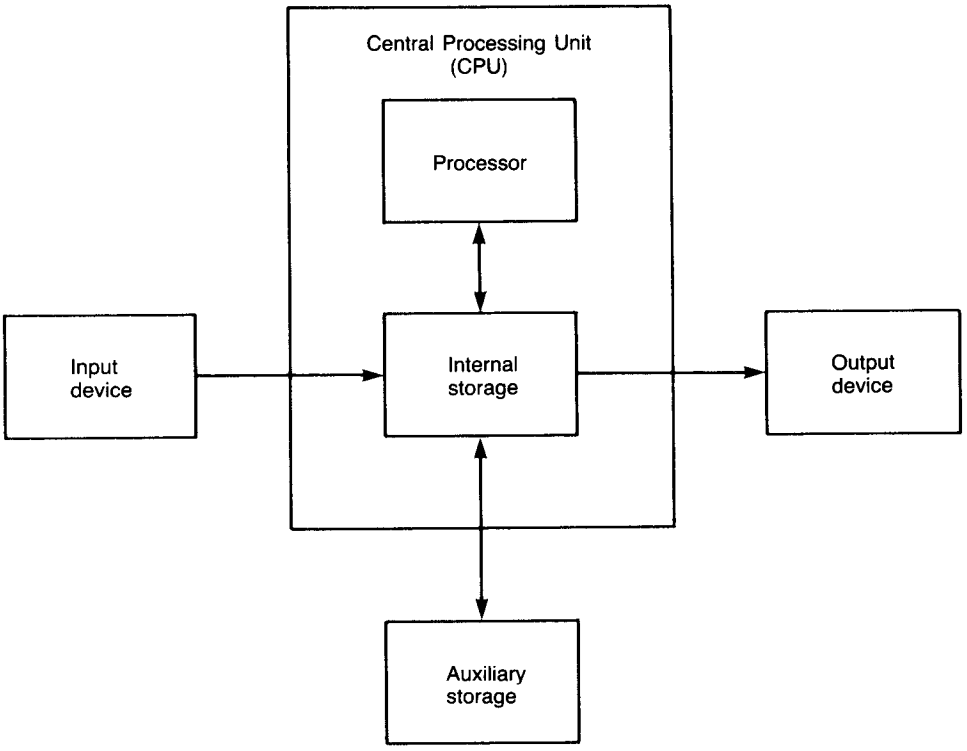
Three topics that should be reviewed before studying COBOL are computers, programs, and data. Basically, a *computer* is an electronic device that processes data by following the instructions in a program. A *program* is a set of instructions that is stored in the computer and performed or executed automatically by the computer. *Data** is facts, figures, numbers, and words that are stored in the computer and processed according to the program's instructions.

### Computers

A computer consists of several interconnected devices or components. One way of viewing the organization of a computer is shown in Figure 1–1. In this diagram,

* The word "data" is most correctly used as a plural noun. The singular of data is "datum." The usual practice, however, is to use the word data in a singular rather than plural sense. We will follow that practice in this book.

Figure 1–1. The organization of a computer




boxes represent the different components of the computer and lines with arrowheads show the paths taken within the computer by data and program instructions. There are five basic components: the input device, the output device, the internal storage, the processor, and the auxiliary storage. Sometimes the internal storage and processor together are called the central processing unit or CPU.

In this subsection we describe each of the components diagrammed in Figure 1–1. Figures 1–2, 1–3, and 1–4 show actual computers with many of the components discussed here.

**Input and output devices.** An *input device* is a mechanism that accepts data from outside the computer and converts it into an electronic form understandable to the computer. The data that is accepted is called *input data,* or simply *input.* For example, one common way of entering input into a computer is to type it with a typewriter-like *keyboard.* This keyboard is an input device. Each time a key is pressed, the electronic form of the symbol on the key is sent to the computer.

Another way of entering input into a computer is to use a device that reads the data from *punched cards* ("IBM" cards). Figure 1–5 shows an example of punched card input. The patterns of holes in the card represent different data. An input device for punched cards recognizes this data and transforms it into an electronic form understandable to the computer. Such a device is called a *card reader.*

An *output device* performs the opposite function of an input device. An output device converts data from its electronic form inside the computer to a form that can be used outside. The converted data is called *output data,* or simply the *output.* For example, one of the most common forms of output is a printed document or *report.* We often call this a computer "printout." Figure 1–6 shows an example of printed report output.

**Figure 1–2.** A typical computer. This is a Digital VAX 11/780. (Courtesy of Digital Equipment Corp.)

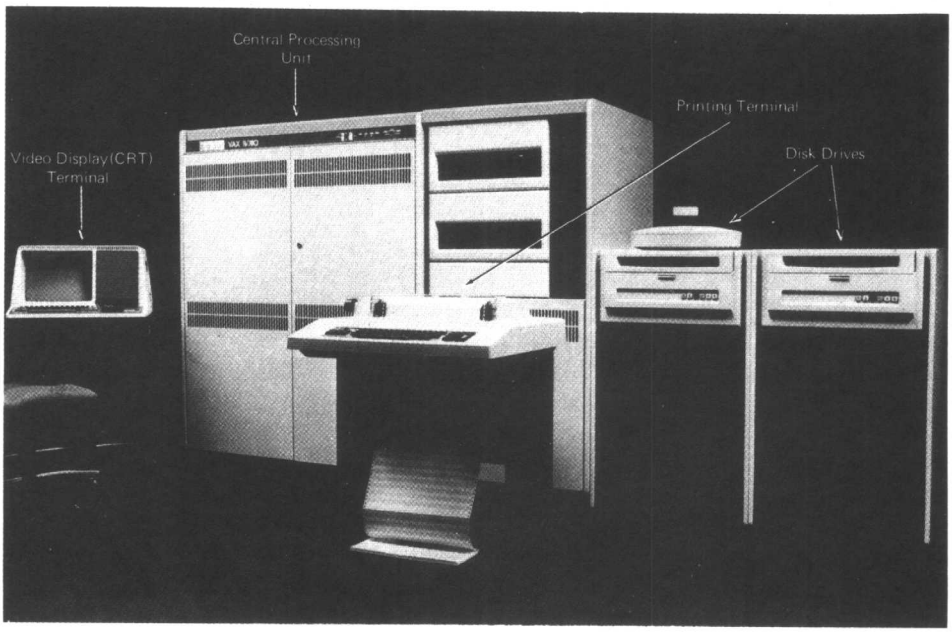



**Figure 1–3.** An older-style computer. This is an IBM System/370. (Courtesy of IBM Corp.)

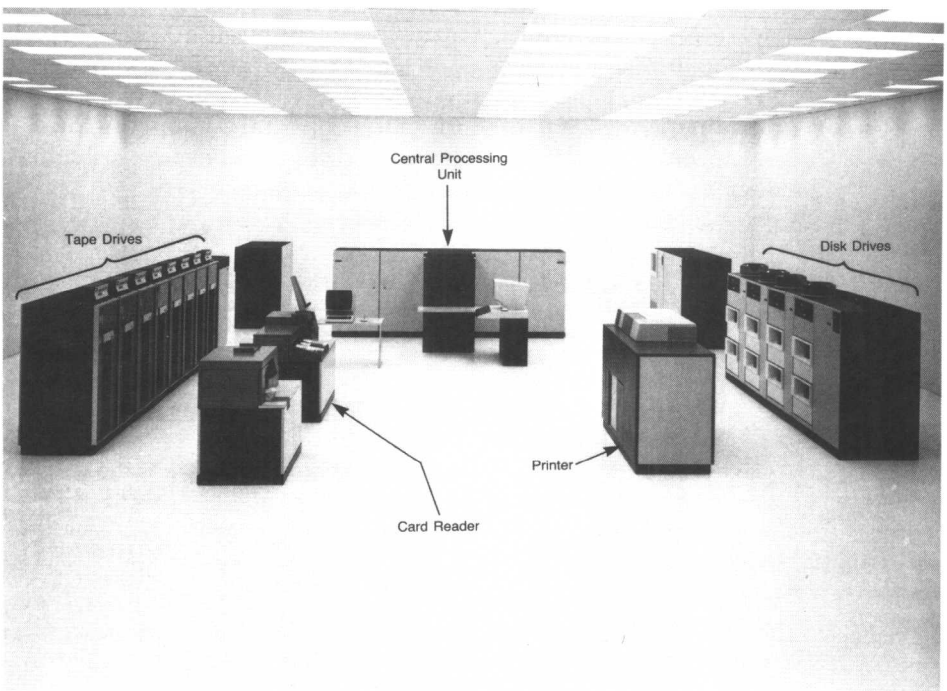
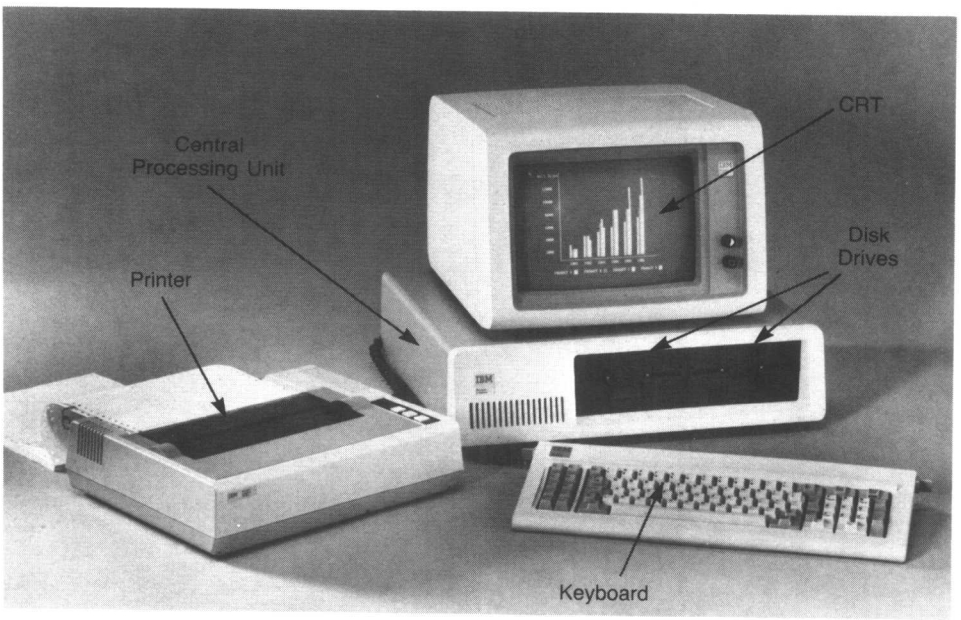


**Figure 1–4.** A microcomputer. This is an IBM Personal Computer. (Courtesy of IBM Corp.)




Printed output is produced by a device called a *printer*. This device converts data from the computer into printed symbols to produce a paper copy of the output. Instead of being printed on paper, output is sometimes displayed on a TV-like screen. Such a video display device is called a *CRT* for *c*athode *r*ay *t*ube (another name for a TV tube).

When keyboard input is used with printer or CRT output, the devices are often combined to form a unit called a *terminal*. Figure 1–2 shows a computer with two types of terminals: a printing terminal and a video display (CRT) terminal.

Input and output devices are often referred to together as input/output or *I/O* devices. Most computers have several I/O devices attached at one time. For example, a medium-sized computer may have many terminals plus a card reader and a printer. Some small computers, however, have only one input device and one output device (such as a keyboard and a CRT).

**Figure 1–5.** Punched card input

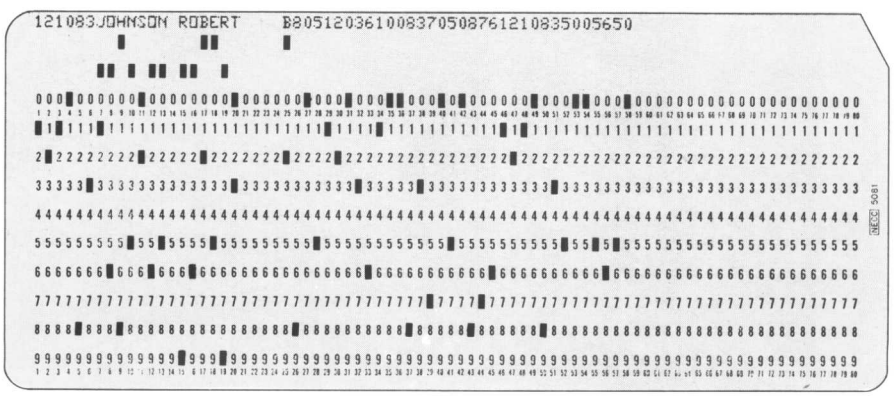


Figure 1–6. Printed report output

YEAR-TO-DATE SALES REPORT

| SALESPERSON NUMBER | SALESPERSON NAME | SALES | RETURNS | NET |
|---|---|---|---|---|
| 0005 | BENNETT ROBERT | 2,850.35 | 38.00 | 2,812.35 |
| 0016 | LOCK ANDREW S | 382.72 | 95.35 | 287.37 |
| 0080 | PARKER JAMES E | 90,700.14 | 555.00 | 90,145.14 |
| 0239 | HAINES CYNTHIA L | 101,000.00 | 2,200.00 | 98,800.00 |
| 0401 | REDDING OLIVIA | 116,159.15 | 24,052.64 | 92,106.51 |
| 0477 | SMITH RICHARD A | 35,450.00 | 510.00 | 34,940.00 |
| 0912 | EMERY ELIZABETH G | 36,200.35 | 1,730.15 | 34,470.20 |
| 1060 | ROBINSON WILLIAM L | 60,350.00 | 25.00 | 60,325.00 |
| 1083 | JOHNSON ROBERT | 63,311.96 | 893.55 | 62,418.41 |
| 1111 | FREDERICKS RICHARD | 52,600.00 | 483.50 | 52,116.50 |
| 1133 | MARSHALL M S | 55,000.00 | .00 | 55,000.00 |
| 1205 | HOLT BENTLEY | 14,881.74 | 413.52 | 14,468.22 |
| 1374 | BENTON ALEX J | 55,600.13 | 6,267.50 | 49,332.63 |
| 1375 | TAYLOR EVERETT | 32,250.00 | 1,125.00 | 31,125.00 |
| 1420 | EHRHARDT ELISE | 4,890.64 | 981.00 | 3,909.64 |
| 1442 | ADAMS JUNE R | 96,771.46 | 1,572.36 | 95,199.10 |
| 1612 | LOCATELLI FRANK | 14,750.00 | 1,505.00 | 13,245.00 |
| 1698 | GUZMAN JOSE | 32,460.00 | 183.00 | 32,277.00 |
| 1842 | COLE ROBERT N | 106,650.39 | 21,637.92 | 85,012.47 |
| | TOTALS | 972,259.03 | 64,268.49 | 907,990.54 * |

**The central processing unit.** Between the input devices and the output devices is the component of the computer that does the actual computing or processing. This is the *central processing unit,* or *CPU.* (See Figure 1–1.) Input data is converted into an electronic form by an input device and sent to the central processing unit where the data is stored. In the CPU the data is used in calculations or other types of processing to produce the solution to the desired problem. After processing is completed, the results are sent to an output device where the data is converted into the final output.

The central processing unit contains two basic units: the internal storage and the processor. The *internal storage* is the "memory" of the computer. Data currently being processed is stored in this part of the CPU. Instructions in the program being executed are also stored in the internal storage.

The *processor* is the unit that executes instructions in the program. Among other things, the processor contains electronic circuits that do arithmetic and perform logical operations. A computer can do the basic arithmetic tasks that a human can do; that is, a computer can add, subtract, multiply, and divide. The logical operations that a computer can do are usually limited to comparing two numbers to determine whether they are equal or whether one is greater than or less than the other. Complex data processing is accomplished by long sequences of these basic operations.

The processor also contains electronic circuits that control the processing in the other parts of the computer. The control circuits perform their function by following the instructions in the program. The program is stored in the computer's internal storage. During processing, each instruction in the program is brought from the internal storage to the processor. The control circuits in the processor analyze the instruction and send signals to the other units based on what the instruction tells the computer to do. The execution of one instruction may involve actions in any of the other parts of the computer. After one instruction in the programmed sequence is executed, the next is brought from internal storage to the processor and executed. This continues until all the instructions in the program have been executed.

**Auxiliary storage.** The final component of a computer is the *auxiliary storage.* This component stores data that is not currently being processed by the computer and

programs that are not currently being executed. It differs from internal storage, which stores the data and instructions that are being processed at the time by the computer. Sometimes internal storage is called *primary storage* and auxiliary storage is called *secondary storage* or *mass storage*.

A common type of auxiliary storage is a *magnetic disk,* which resembles a phonograph record. Disks are available in different sizes: some about 5 inches or less in diameter and others as large as 14 inches across. Information is recorded on the surface of the disk by patterns of magnetism. A *magnetic disk drive* is a device for recording data on magnetic disks and for retrieving data from the disks.

Another type of auxiliary storage is *magnetic tape,* which is much like audio recording tape. Magnetic tape comes in reels of various sizes and even in cassettes. Data is recorded on the surface of the tape by patterns of magnetism. A *magnetic tape drive* is a device that records data on magnetic tape and retrieves data from the tape.

Most computers have several auxiliary storage devices attached at one time. For example, a computer may have four disk drives and two tape drives. Other types of auxiliary storage can also be used; however, magnetic disk and magnetic tape are the most common.

**Computer hardware.** The physical equipment that makes up a computer system is called *hardware.* Computer hardware consists of keyboards, printers, CRTs, terminals, card readers, CPUs, disk drives, tape drives, and other pieces of equipment. Figures 1–2, 1–3, and 1–4 show typical computers with the hardware that we have described.

## Programs

A computer program is a set of instructions that tells the computer how to solve a problem. A program is prepared by a person, called a *programmer,* who is familiar with the different things a computer can do. First the programmer must understand the problem to be solved. Then he or she determines the steps the computer has to go through to solve the problem. The programmer then prepares the instructions for the computer program that solves the problem.

To illustrate the idea of a computer program, let us assume that we want to use a computer to solve the problem of finding the sum of two numbers. The computer must go through a sequence of steps, as follows. First the computer must get two numbers from an input device. Then the numbers must be added to find their sum. Finally the sum must be sent to an output device, possibly a printer, so that we can see the result. Thus, a computer program to solve this problem would have three instructions:

1. Get two numbers.
2. Add the numbers.
3. Print the result.

The instructions would be prepared in a form the computer could understand. Once this was done, the program could be executed by the computer.

In the remainder of this subsection we explain how such a program is executed, what languages are used to prepare programs, and what types of programs are needed.

**Program execution.** To execute a computer program the instructions in the program must be entered into the computer using an input device. For example, the instructions can be keyed in using a keyboard or they can be punched in cards and read with a card reader. The program is then stored in the computer's internal storage.

In executing a program the computer goes through the instructions one at a time in the sequence in which they are stored. For example, assume that the program

to find the sum of two numbers has been entered into the computer and is stored in internal storage. The computer would bring the first instruction in the program from internal storage to the processor. Then execution would proceed as follows:

1. Get two numbers. The processor examines this instruction and sends a signal to the input device that causes two numbers (input data) to be transferred to internal storage. The second instruction is then brought to the processor.
2. Add the numbers. To execute this instruction the processor issues a signal to internal storage that causes the two numbers to be sent to the arithmetic circuit in the processor. Then the numbers are added and the result is stored in internal storage. The last instruction is then brought to the processor.
3. Print the result. The processor executes this instruction by sending a signal to internal storage to transfer the result to the output device. Then the output device prints the output data.

Two important concepts are illustrated by this example. The first is that internal storage is used to store both program instructions and data. All instructions in the program are stored in internal storage before the program begins execution. Data is brought into internal storage as the program executes.

The second important concept is that the instructions in the program are executed in the sequence in which they are written. The sequence must be such that, when executed, the problem is correctly solved. If the instructions are out of order, the computer cannot figure out what the right sequence should be. In such a case, the computer would follow the instructions in the order in which they appear and thus produce an incorrect result.

**Computer programming languages.** A program must be written in a form that a computer can understand. Every instruction must be prepared according to specific rules. The rules form a language that we use to instruct the computer. Humans use *natural languages* such as English and Spanish to communicate with each other. When we communicate with a computer we use a *computer programming language.*

To write a sentence in a natural human language, we form words and phrases from letters and other symbols. The construction of the sentence is determined by the grammar rules of the language. The meaning of the sentence depends on what words are used and how they are organized. A computer programming language also has rules that describe how to form valid instructions. These rules are called the *syntax* of the language. The meanings or effects of the instructions are called the *semantics* of the language. For example, the *syntax* of a particular computer language may say that one type of instruction has the following form:

```
ADD field-name, field-name GIVING field-name
```

That is, the instruction consists of the word ADD, followed by two field-names separated by a comma, then the word GIVING, and finally another field-name. (Of course, we must know what a field-name is in order to complete the instruction.) The *semantics* of the language tells us that this instruction means to add the values identified by the first two field-names and to assign the result to the third field-name. (We will study this instruction in detail in Chapter 3.)

In this book we discuss the syntax and semantics of the COBOL computer programming language. COBOL is just one of many programming languages. In fact, there are several groups of languages and many different languages in each group.

One group of languages is called *machine language.* This is the language in which a computer actually does its processing. To a computer it is a series of electronic