INTRODUCTION TO

Microcomputer Programming

PETER ROB

BASIC

Introduction to Microcomputer Programming

Peter Rob
Professor of Information Systems
Middle Tennessee State University

Wadsworth Publishing Company
A Division of Wadsworth, Inc.
Belmont, California

Computer Science Editor: Frank Ruggirello
Production Editor: Leland Moss
Managing Designer: Detta Penna
Designer: Edith Allgood
Cover Designer: Stephen Rapley
Cover Photograph: Dow, Clement & Simison
Copy Editor: Susan Thornton
Technical Illustrator: Evanell Towne
Cartoons © 1983 Rand Renfroe

ISBN 0-534-03184-6

Printed in the United States of America

3 4 5 6 7 8 9 10 — 88 87 86 85

Library of Congress Cataloging in Publication Data

Rob, Peter.
Introduction to microcomputer programming.

Includes index.
1. Microcomputers—Programming. 2. Basic (Computer program language) I. Title.
QA76.6.R62 1984 001.64'2 83-23267
ISBN 0-534-03184-6

Preface

Unless you are a confirmed masochist, you must have some compelling reasons for undertaking the long and difficult task of writing a textbook. So many selections abound—why not just pick one? However, as I evaluated the available textbooks for an introductory BASIC microcomputer programming course, I found it difficult to find any with the desired attributes.

An introductory programming text should not strive to develop instant professional programmers. Yet, if the text's expectations are too low, students are not likely to gain useful programming skills. Most books I reviewed sacrifice programming proficiency at the altar of simplicity. I have written this text to correct that mistake, without losing sight of the nature of an introductory programming course.

As you examine this book, you will notice several uncommon features. Each feature is designed to help guide the student to greater programming proficiency without posing an unacceptable degree of difficulty. These features include:

1. Programming techniques that are easily adaptable to IBM PC, the APPLE II, and the TRS-80
2. Greater topical coverage
3. Greater depth of coverage
4. Extensive sample quizzes with answers
5. Expandable programming problems
6. Appendixes containing system commands and examples of their use on the IBM PC, the APPLE II, and the TRS-80
7. Detailed summaries and comparisons of sequential and random access file procedures for the IBM PC and the APPLE II

Adaptable Programming Techniques

Novice programmers are frequently frustrated when they discover that their programming techniques do not work very well on different microcomputer systems. This book helps decrease that frustration by introducing the student to three of the most popular microcomputers—the IBM PC, the APPLE II, and the TRS-80. Although the program examples are all done on the IBM PC, notes for APPLE II and TRS-80 programmers are highlighted where appropriate.

System-specific programming techniques are avoided where possible. However, files and high resolution graphics are unavoidably system-specific. Therefore, separate file chapters exist for the APPLE II and the IBM PC. (The TRS-80 files are, for all practical purposes, identical to those of the PC.) In addition, the high resolution graphics chapter is divided into two sections, one each for the IBM PC and the APPLE II.

Greater Topical Coverage

Among other important functions, computers can organize and present numerical information, ranging from teachers' grade summaries to various business reports. Because such reports are often best generated on the basis of **two-dimensional arrays,** this text includes a chapter on that subject.

The ability to create programs that make extensive use of subroutines is a hallmark of good programming technique. Both **subroutines** and **ON . . . GOSUB techniques** are discussed, presented, and used, especially in conjunction with simulations and files. Subroutines are also used in connection with menu-driven programs.

The ability to store and retrieve information is an especially useful aspect of computer use, and both **sequential** and **random access** (direct access) **files** are covered in this book. In order to make file creation and manipulation easier to understand, the nature of files, fields, and records is explored in greater detail than is common in most introductory BASIC programming texts. Numerous file options are covered as well, including the creation of files, deleting or adding records from or to files, changing records in files, and transferring records between files.

Since a picture is often worth a thousand words, the book presents **high resolution graphics** procedures for both the IBM PC and the APPLE II. Coverage goes beyond fundamental graphics commands to show applications like frequency plots. Given the difficulty of combining high resolution graphics with text on the APPLE II screen, the chapter includes a discussion of how to use subroutines that define numbers and letters as "binary" strings.

Although most introductory BASIC programming texts include some discussion of **random numbers generators,** this text covers more application areas. Simulation is covered through elementary Monte Carlo methods, and a simple example of computer-aided instruction is shown, using random numbers to create a computer-scored arithmetic quiz.

Greater Depth of Coverage

This text demonstrates that, even within a simple framework, a useful level of programming proficiency can be attained. Consequently, discussions, programming examples, and problems often probe into more realistic applications. For example, sequential and random access file programs are written to be menu-driven, incorporating in a single program the ability to create, delete, add to, or correct files.

Computer-generated reports tend to be more realistic and show greater detail and capability than is common in introductory BASIC programming texts. The use of two dimensional arrays, coupled with variable TAB functions, allows the user to compute, format, and print row/column totals, subtotals, and grand totals.

Other programming topics are covered in similar detail, using simulation results, graphics-based summaries, and so on. The relatively higher level of programming sophistication is attained partially because the text introduces programming principles, style, and examples in a gradual, unhurried manner. Although this conversational writing style increases the book's length, such an approach offers the freedom to explore topics in greater depth without becoming cumbersome or threatening.

Extensive Sample Quizzes With Answers

At the end of each chapter a multiple choice/true–false quiz provides quick feedback to students. Quiz questions deal primarily with programming techniques and are sufficiently challenging to provide a realistic evaluation of the material learned in each chapter.

Expandable Programming Problems

Problem sets provide a sense of direction and the recognition that the student is gaining significant programming expertise. Many problems done in one chapter reappear in a more sophisticated format in subsequent chapters, designed to create a useful level of programming expertise. For example, a very simple program that only adds two numbers may expand into a program that uses a random number generator to select the values to be added. In a later chapter a scoring device and, finally, a random access file is added to the program to store the results. At this point the programming student has developed a program that can test an elementary school student's arithmetic, providing each student with a unique test, scoring the test, storing the test results, and, if necessary, printing a grade report for each student in the file. If the programming student happens to be an elementary school teacher, the usefulness of the microcomputer has surely been demonstrated.

Similarly, a frequency plot is initially based on given frequency data. The student is then shown how to let the computer generate the frequency distributions. Finally, the high resolution graphics chapter demonstrates yet another use for the frequency plot.

Appendixes Containing System Commands

While system users manuals are very informative, many students either have no access to—or no interest in—the highly detailed examination of specific microcomputer systems. This book, therefore, contains an appendix for each major microcomputer system: IBM PC, APPLE II, and TRS-80. The appendix covers such topics as how to turn the computer on, how to initialize a disk, how to name programs, and how to use system commands such as SAVE, LOAD, LIST, DELETE (KILL), and RUN.

Summaries of Sequential/Random Access File Procedures

Writing file programs can be very frustrating, since such programs contain so many unusual and very specific features. To diminish such frustration, this text provides a summary, with examples, of the most important file procedures, such as opening a file, closing a file, getting data from a file, putting data into a file, and so on.

Acknowledgments

Writing a textbook is an exciting, yet long and often painful task. To start writing a textbook is easy enough; to finish it requires the help and understanding of many (often unsung) heroes. Among the important contributors to the successful completion of this text is Dr. William J. Grasty, who chairs the department of accounting and information systems at Middle Tennessee State University. Bill Grasty was always willing to let me experiment with course materials, often providing the necessary encouragement to make writing a less painful process. He also scheduled my courses and sections in order to help create blocks of writing and programming time. In addition, he helped locate several truly outstanding assistants who transformed my midnight scribbling to the typed manuscript.

My first assistant, Betsy Garmany, helped produce the initial rough draft of this text. Her expertise on the word processor and her constant good cheer are hereby gratefully acknowledged. Ms. Garmany's many outstanding qualities were easily recognized by a very fortunate employer, causing her retirement from this project.

Many thanks to Lisa Northcutt, who produced the remainder of the book's first draft and who helped me through the first (and many other) rewrites. Ms. Northcutt's ability to juggle schedules and her willingness to work overtime

made it possible for me to meet the many deadlines that authors are heir to. Her very agile mind made the word processor perform miracles. Without her, this book would possibly still be a two-foot-high pile of scribbles and computer printouts.

Two additional student assistants provided important help in transforming the book's rough draft into the finished product. Tambra W. Peters assumed the difficult task of typing the computer listings in Chapter 13 and in Appendix D, in addition to doing some of the final editing. Jeanne D. Williams used her considerable expertise in word processing to help make many of the final corrections. Meeting the numerous deadlines would have been impossible without their contributions.

The light touch in this book is provided by Rand Renfrew, whose outstanding cartooning skills drive programming points home or provide the often therapeutic chuckle. Rand's familiarity with programming and its apparently inevitable "bugs" has created the sense of "Yes, I've been there." Many thanks to him for contributing his art.

How can I sufficiently thank the many readers whose critiques were so useful in reshaping the initial efforts? Their contributions range from ferreting out typographical errors to questions dealing with continuity, program logic, and pedagogical worth. Most of these readers will find their suggestions and recommendations incorporated in the text. The contributions of the following are hereby gratefully acknowledged: Professors Michael Cox, Golden West College; Bernard Eisenberg, Kingsborough Community College, New York; Dave Hart, Weber State, Utah; Sister M. K. Keller, Clarke College, Iowa; Norman Licht, S.U.N.Y. at Pottsdam; Leonard Presby, William Patterson College; Leona Roen, Oakton Community College, Illinois; Fred Snyder, Bryant and Stratton Institutes, Powelson, New York; and Richard Tangeman, Arkansas State University.

Surely no author could be more indebted to a publisher. Frank Ruggirello, Senior Editor at Wadsworth, is truly one of the class acts in the publishing business. Frank pushed this project (and its author) to the limit, providing talent and resources to get the job done right and on time. Judith McKibben, developmental editor, provided a sense of direction and numerous writing suggestions. Suzanna Brabant channeled a constant stream of letters and phone calls, each dealing with some important aspect of the writing process. Leland Moss and Detta Penna, as well as others on a fine production staff, transformed the final edited manuscript into an attractive textbook.

Finally, my wife Anne's contributions are hereby gratefully acknowledged. Her numerous comments and suggestions are all reflected in these pages. Her sense of humor has helped keep all events (and especially those many deadlines!) in proper perspective. During the past 22 years she has gracefully endured numerous industrial consulting crises, research deadlines, and, with this text, the many hours of proofreading. Without her, much of what I do would not be worth doing. I therefore dedicate this book to my wife and best friend, Anne.

Peter Rob

Table of Contents

1

About Those Microcomputers

OBJECTIVES

1. To describe the nature and characteristics of microcomputers
2. To define and describe commonly used technical terms
3. To discuss the basic components of a microcomputer
4. To discuss, in general terms, the operation of a microcomputer
5. To describe the nature and function of a computer program
6. To introduce a computer programming language known as BASIC

This chapter answers four main questions: Why study computers? What are the components of a microcomputer? How does a microcomputer work? What is computer programming, and why use BASIC as a computer language?

1.1 What Is a Computer?

In a fundamental sense, a **computer** is a piece of equipment that performs computations. So, of course, does a calculator. What distinguishes a computer from a calculator is that whereas a calculator requires "prompting" from a human being each time a calculation is performed, a computer can perform thousands, hundreds of thousands, or even millions of calculations without direct human intervention. The most important distinction is that a computer can perform logical operations that require *making choices* among alternatives. It is this feature that makes the computer useful in so many different applications.

A computer does not, of course, have an inborn ability to make choices. It must be given a set of detailed instructions, called a **program,** on how to respond under different circumstances. This text will teach you how to write programs. Mastering this skill can be exciting, especially when we can make a computer perform exactly according to our designs and specifications. In addi-

tion, learning to program is a helpful discipline because it encourages us to be logical in approaches to problems. Programming, in fact, may turn out to be one of the most satisfying learning experiences.

Uses of a Computer

Suppose that you want to teach someone a bit of arithmetic known as addition. When this student adds two numbers, the result may be correct or incorrect. If the answer is incorrect, the value may be too high or too low. The student can use a computer to check his or her answer against one that the computer can calculate, and the computer can then evaluate the difference, if any, between the two answers.

The use of the computer as a teaching tool is thus based on its ability to compare a user-supplied answer to its own. For example, if the student were to add the values 2 and 4, the answer should be 6. If, in fact, he or she did answer 6, the computer might respond by giving the message YOU ARE RIGHT. If, on the other hand, the student had answered 5, the computer might respond by printing the message NO, YOUR ANSWER IS TOO LOW. WANT TO TRY AGAIN? Similarly, if the answer had been 8, the response might be NO, YOUR ANSWER IS TOO HIGH. WANT TO TRY AGAIN?

The computer's ability to make choices among alternatives can also be applied to industry. For instance, a computer may be instructed to monitor the operating temperatures of a piece of manufacturing equipment. Is the current operating temperature within defined acceptable limits? If the answer is NO, the computer may shut the equipment down, or it may increase the cooling rates. If the answer is YES, the computer may check to see if there is an upward trend in the operating temperature. If there is such a trend, the computer may take action to correct the problem before it becomes critical. Such applications—and much more complicated ones—are all possible because a computer can make logical choices among alternative courses of action. It is, therefore, the ability to make choices that makes the computer so unique and so powerful.

1.2 Why We Need to Know About Computers

It is important to realize that not only can computers do much but that they are already doing it! Computers and computer applications are all around us. Computers keep our bank balances, monitor jet engine fuel flows, check for unsafe conditions in manufacturing plants, help design products we use, draw pictures, play games, make sculptures, teach concepts, and perform diagnostic checks on our cars as well as our bodies. Perhaps even more important, as the number of computers and computer applications increase rapidly, computer costs decrease dramatically. Small, easy-to-use, yet powerful computers known as *microcomputers* are available in 1984 for less than $2,000. In many respects, these low-cost computers do more than the very large and complex computer systems did in the 1960s at a cost of millions of dollars. And the trend continues: In 1984 lower-powered microcomputers can even be purchased for less than $100.

Apart from the direct computer applications listed above, computers perform tasks that may be even more important to us in the future. Computers, with their incredible ability to store, process, and interpret information, can create information networks; compare, match, or link records; and perform the millions of manipulations that we demand in our quest for meaning in the information. There may be a darker side to this use of computers, as well. Information can be misused, the concepts of science can be misapplied. As the

number of computers multiplies, the potential for abuse increases. But whatever our hopes and fears, they will not be buoyed or alleviated by ignoring the computer revolution. The computer is here to stay, touching virtually all our lives: We really do not have a choice about becoming acquainted with it.

Learning About Computers

There are two approaches to learning about computers. We may choose to study the equipment itself, to see how and why it functions. Or we may concentrate on learning how to use computers. This book, as its title suggests, focuses on the second approach. Yet we cannot remain ignorant of the basic components of the computer. Many of us will probably buy small computers, and we need to be able to evaluate both the equipment and our needs in simple terms. How, for example, would you interpret an advertisement describing a small computer with "64 Kb RAM, 5 Mb internal storage, and 2 disk drives"? What is a *CRT?* How could we determine whether we needed a *disk drive?* Such terms are explored briefly in this chapter to help us become better informed computer consumers.

Programming

While there are many computer programming languages, we will concentrate on a language known as **BASIC** (Beginner's All-purpose Symbolic Instruction Code). BASIC was developed in the mid-sixties by Dr. John Kemeny and Dr. Thomas Kurtz at Dartmouth College. Relatively few demands were placed on the new computer language; its primary use was educational, and its primary advantage was simplicity. Since 1965, BASIC has grown from its modest beginning to become a powerful and very popular computer language. In fact, BASIC is probably the most widely used computer language in the world.

Despite its growth in popularity and power, BASIC has managed to retain its simplicity. It is still relatively easy to learn. Beyond simplicity, however, there are better reasons to study BASIC. First, it has become a language capable of complex manipulations. Second, and probably more important, BASIC is used almost exclusively by microcomputers.

1.3 What Is a Microcomputer?

A **microcomputer** is a "computer on a chip." The **chip,** which may be only a fraction of an inch square, contains all of the electrical circuits that allow the microcomputer to perform both arithmetic (add, subtract, multiply, and divide) and logical operations (compare and conclude). Such a chip is called a microprocessor.

The Microprocessor

Physically, a **microprocessor** is a single integrated circuit etched on a thin wafer (chip) made of silicon, the primary component of beach sand. The silicon used to manufacture the chip is, of course, a highly refined substance. The **integrated circuit** is a complex collection (circuit) of very small electronic components that control on/off switches. The circuit is called *integrated* because all of the necessary components that need to work together (in other words, integrated) are located on a single chip. The position of various combinations of on/off switches allows the computer to process and "remember" information.

The microprocessor's very fine circuits must be connected to an electrical source. Therefore, the microprocessor is embedded in a piece of plastic

with prongs attached to it. The combination of microprocessor and its "bed" is known as a **dual inline package,** or **DIP.** Both microprocessors and DIPs are illustrated in Figure 1.1. A microcomputer may consist of a single chip embedded in a single DIP. Most microcomputers are composed of a collection of DIPs connected to a printed circuit card, or **board.** Technically speaking, then, *a microcomputer is a computer whose central processing unit is a microprocessor.*

The Central Processing Unit

The **central processing unit (CPU)** is, in a sense, the computer's central nervous system, directing its activities. The CPU actually has two functions. The portion of the CPU that provides the arithmetic/logic function, known as the **ALU,** does the actual computing. The second function of the CPU controls activities in the various portions of the computer. The CPU's second portion is, therefore, called the **control unit (CU).** The CU is, in effect, a switching center that directs various programming commands and information flows to appropriate sections of the computer.

However, not all microcomputers use the same brand of microprocessor. For example, the TRS-80 microcomputer uses a Z80 microprocessor, the Commodore 64 uses an MCS6510, the IBM PC uses an 8088, and so on. Various microcomputers use different operating systems, all of which direct the computer's activities.

The collection of computer commands that operates the CU is known as the **operating system.** The operating system that "manages" the computer is generally a collection of programs supplied by a manufacturer. The different microcomputer operating systems may be accessed by using specific BASIC commands. We will discover, therefore, that BASIC is not standard on all systems. Each system may use its own BASIC "dialect."

A Note About System-Specific Programming

Given the existence of different BASIC dialects, we could elect to write computer programs that use certain *commands* that are tailored precisely to the system being used. When we write a program specifically designed for a particular system, we refer to such a program as **system-specific.** Because system-specific programs cannot be modified easily to run on other systems, system-specific programming is not always a good practice. Fortunately, unless we try to access very specialized microcomputer functions such as graphics and file management, we can write programs so that the differences between the BASIC dialects are relatively minor. If we learn to write our programs carefully, the same BASIC codes can usually be used on different systems. In fact, all the programs in Chapters 2–11 were tested on microcomputer systems as radically different as the APPLE II and the IBM PC. Only rarely did the programs require any modification. So, although there is no single standard BASIC language, we can still learn to write programs that are understood by most microcomputers.

Nonetheless, the lack of standard operating systems can occasionally be very bothersome, especially when we are dealing with the files that store and access information. However, many modern microcomputers have adopted a control system known as **CP/M** (control program for microcomputers). The CP/M system was designed by Digital Research for microcomputers that use microprocessors known as the 8080, the Z80, the 8085, and their derivatives. Language differences among the various CP/M-based microcomputers tend to be minor. Although the APPLE II uses a non-CP/M system, it can be modified by inserting a CP/M board containing the appropriate microprocessors. How-

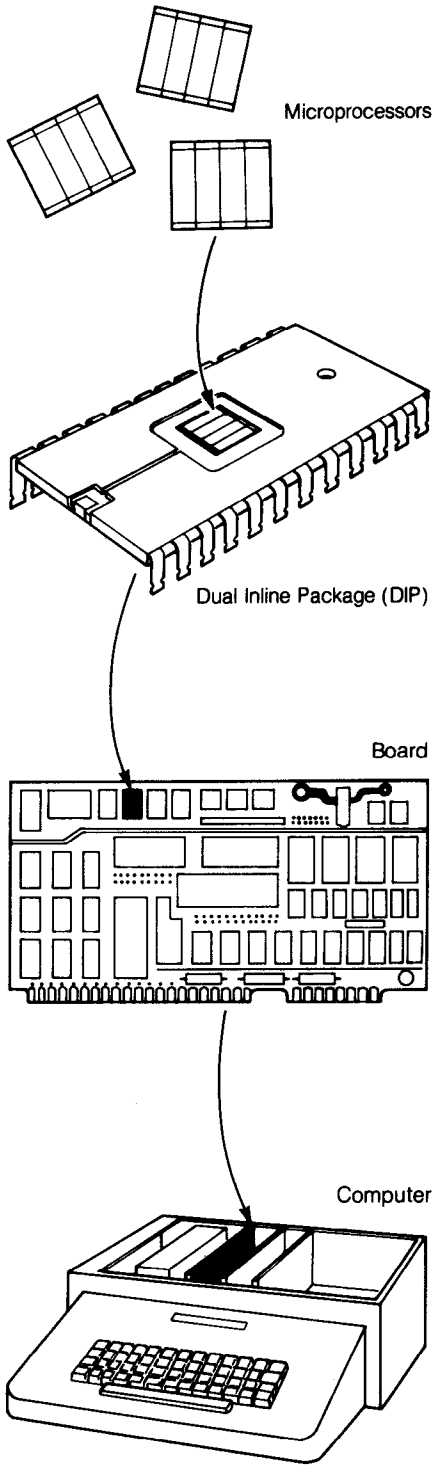

Figure 1.1
MICROPROCESSORS AND DIPs

ever, since most APPLE II microcomputers have not been modified, this text contains occasional programming commands designed for Apple II programmers, as well as a separate chapter on files.

1.4 Gaining Access to a Microcomputer System

The Keyboard

Microcomputers are very small computers; indeed, the word *micro* means "very small." The actual computer itself may be only a fraction of an inch square! The microcomputer looks bigger than that, of course, because humans must be able to access it through a *keyboard,* which must be relatively large because of our big fingers. Since we send information into the computer via the keyboard, the keyboard is also referred to as an **input device.**

The CRT Screen and Printer

The results of the computer's activities are monitored by a TV screen, called a CRT (cathode ray tube) or a VDT (video display terminal). The screen has to be large enough to be read easily. Because the information coming out of the computer is shown on the CRT, we usually refer to a CRT as an **output device.** If we want a permanent copy of the output shown on the CRT, we can attach a printer to the system. The printer, too, is called an output device. When we see the abbreviation **I/O,** we should realize that *I* refers to input, and *O* refers to output. Therefore, the keyboard (input) and the CRT or the printer (output) are often described as **I/O devices.**

1.5 Memory

Thus far, we have described a computer system that consists of these components:

1. Central processing unit (CPU), composed of a control unit (CU) and an arithmetic/logic unit (ALU)
2. Keyboard (input device)
3. CRT (VDT) and/or printer (output device/s)

Unfortunately, we still do not have a functioning microcomputer because we lack several important pieces of equipment, the most significant being the **memory,** where the actual computing work will be done. The memory consists of two parts: the **random access memory (RAM)** and the **read only memory (ROM).** The RAM is so named because its contents may be accessed in any (random) order. We can both read information from and write information into the RAM. The ROM, on the other hand, is not as flexible. We can only read information from the ROM; we cannot write information into it. And because all of the information in the ROM was installed in the computer factory (firm), the ROM is referred to as **firmware.**

Measuring the Memory

The amount of work that can be done by the microcomputer depends on how large the memory is. Therefore, if we want to evaluate the microcomputer's capabilities, we must be able to measure both RAM and ROM capacity, that is, how much information can be stored in the memory at one time. The unit of measurement is determined by the way a computer "codes" information.

In its simplest sense, a computer consists of a large number of switches that may be turned *on* or *off.* The combination of on/off switches represents a value, a letter, or a symbol. Since there are only two switch positions for each switch (on or off), we can represent the off position using the value *zero* (0) and the on position by the value *one* (1). Each of these two digits (0,1) is called a **binary digit,** or **bit.** (The word *bi* means "two.") A unit of eight bits forms one **byte.** Each byte represents a single memory location in the computer's memory that is capable of holding a single character of information, such as the number 5 or the letter T. Each combination of on/off switches, in other words, is a code that represents a letter, number, or symbol. To avoid confusion about which code will represent which character, letter, or number, a code has been standardized, called the American Standard Code for Information Interchange (**ASCII**).

For review purposes, let us summarize the main features of RAM and ROM capacity measurement:

1. Characters are coded by using binary digits (bits).
2. Eight bits together form one byte.
3. Each byte can hold one character in memory.

RAM and/or ROM capacity is measured in thousands of bytes, or **kilobytes (K** or **Kb).** (The word *kilo,* incidentally, means "thousand.") Therefore, a micro with an advertised RAM of 48K would have 48,000 bytes available, allowing storage of up to 48,000 characters in the memory at one time. Actually, 1,000 is only a convenient measuring rod: The computer's RAM really contains 1,024 bytes per kilobyte, so 48K represents $48 \times 1{,}024 = 49{,}152$ bytes. Before celebrating this "extra" memory, we must recognize that sometimes not all of the RAM is available to store information as well as instructions for manipulating that information. The computer's central processing unit (CPU) may use some of the available RAM to perform its various controlling functions. Nevertheless, a RAM value of 48K is a good general estimate of what is available. The RAM capability, incidentally, is growing very rapidly; micros now boast 64K RAM, 128K RAM, and even 256K and higher RAM values. One word of caution, however: When you see an advertisement for "64K memory chips," you should realize that the individual chip capacity is measured in terms of *bits* rather than *bytes;* remember, it will take eight 64K chips to make 64K bytes of RAM.

The definition of bit also allows us to evaluate microprocessor capability. Many of the older microcomputers were of the 8-bit variety. Most newer microcomputers have 16-bit microprocessors, which allow a much higher processing speed than the 8-bit microprocessors. A few microprocessors are even designed with a 32-bit capacity.

1.6 Storage Devices

Our microcomputer system is at this point composed of the following components:

1. Central processing unit (CPU), a microprocessor composed of a control unit (CU) and an arithmetic/logic unit (ALU)
2. Memory (RAM and ROM)
3. Keyboard (input device)
4. CRT and/or printer (output device)

Although these four components will allow us to start computing, one final piece of equipment is indispensable. The microcomputer cannot permanently

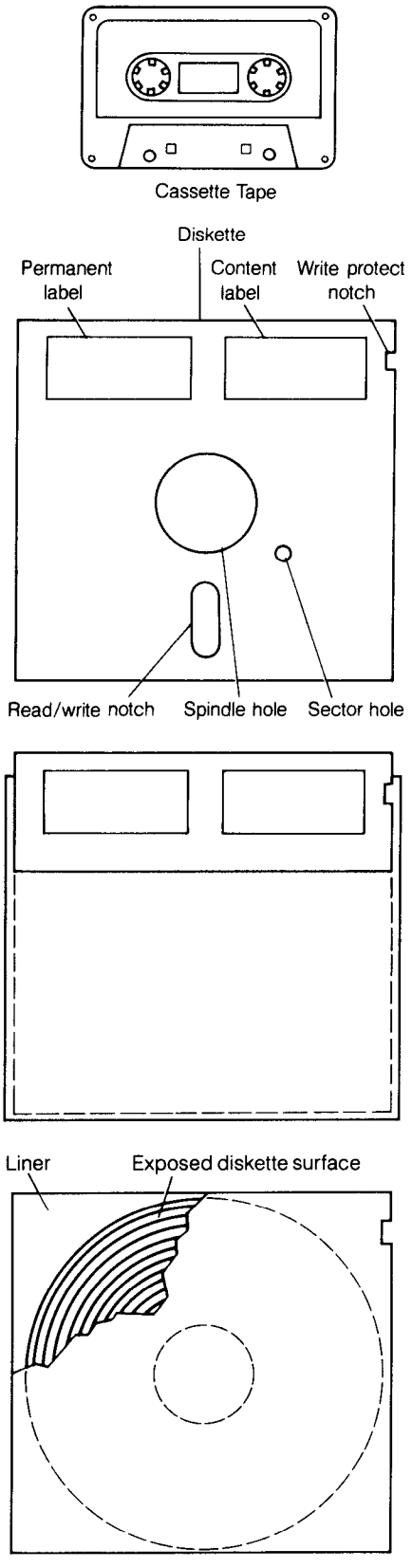

Figure 1.2
MICROCOMPUTER TAPE AND DISKETTE

"save" any information we place in the RAM because the RAM is **volatile**; that is, if we turn the computer off, the information in the RAM is lost forever. (It has evaporated, so to speak.) Quite naturally, if we have spent hours behind the keyboard typing information and instructions, we will quickly lose interest in computing when we discover that all our efforts can be lost that easily. Therefore, we need a storage device that can save the RAM contents permanently. Such storage devices, shown in Figure 1.2, are generally **tapes,** or **disks.**

Cassette Tapes

Microcomputer tapes are usually the same tapes that we use in cassette recorders. In fact, many microcomputers can actually use the same cassette recorders that we use to play recorded music. For several reasons, however, tapes are not very desirable for computing.

1. Information retrieval from tape tends to be very slow. For example, if a specific piece of information is located 15 ft from the current tape position, we must wait until the tape cassette has wound the tape to the desired position. We cannot jump directly to the desired location.
2. The inability to jump back and forth to different tape positions means that many of the microcomputer's best features cannot be used. One of those features involves an information storage and retrieval system known as the direct access file, discussed in Chapters 12 and 13. The term *direct access* implies an ability to go directly to a given information position. Since we cannot jump directly to specific information locations when we are using tape, we cannot use direct access files.
3. Tapes sometimes tend to record information incorrectly because of noise distortion. The tape drive's volume selection position can become a matter of costly experimentation.
4. Tapes can and do break: Splicing broken tapes can be very damaging to the information stored on them.
5. Finally, tapes can hold only a limited amount of information compared to a disk. For these reasons, many modern microcomputers tend to be designed around the ability to use disk rather than tape storage.

Diskettes

The disk system of information storage and retrieval is, when compared to tape, very efficient. We can access information on a disk directly; storage capacity is great and becoming greater; the disk is not subject to noise distortion; and disks do not break as tapes do. On the other hand, disks can be damaged easily if handled roughly. In addition, the cost of the device that spins the disk (the **disk drive**) tends to be high in comparison to the cost of the cassette recorder. Nevertheless, the extra cost is usually offset by the benefits of using the disk drive.

Microcomputers are designed to have special capabilities when harnessed to disk drives. In fact, the disk operating system (DOS) is so important that microcomputers tend to have separate DOS manuals to help the user tap the vast powers of the microcomputer more efficiently.

The most common disk drives use $5\frac{1}{4}$ in. flexible disks known as *floppy diskettes, "floppies,"* or just plain *diskettes.* The diskettes consist of a flexible plastic disk covered with a fine coating of ferrous oxide on which characters such as letters, numbers, or symbols can be deposited magnetically. A typical diskette is shown in Figure 1.2. The diskette's main identifying features include the following: