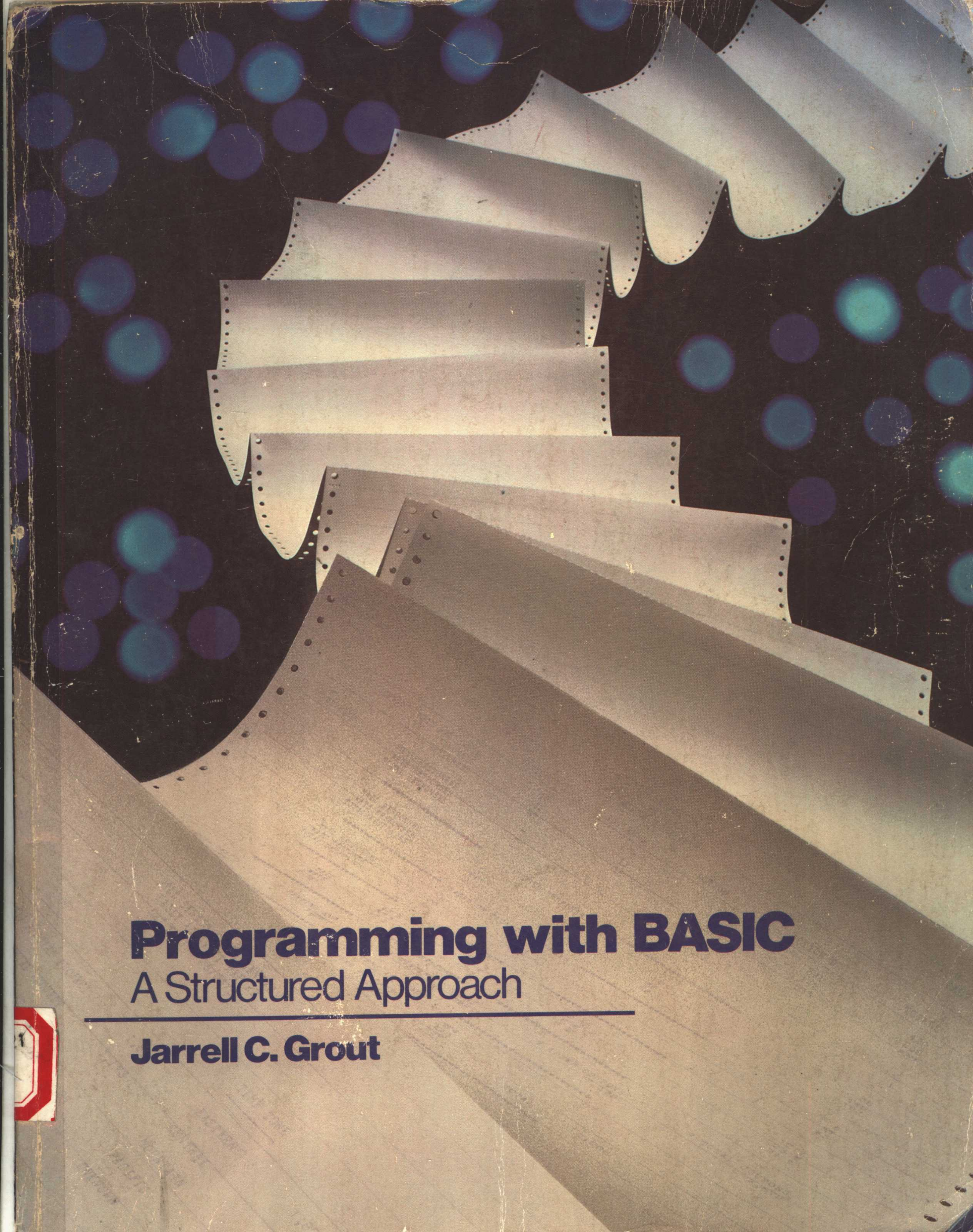


Programming with BASIC

A Structured Approach

Jarrell C. Grout

Programming with BASIC
A Structured Approach

Jarrell C. Grout
Stephen F. Austin State University

wcb
Wm. C. Brown Publishers
Dubuque, Iowa

Cover photograph © Steve Hix Photography

Library of Congress Catalog Card Number 84-73396

ISBN 0-697-00077-X

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

Preface

This book presents the BASIC language in the context of modern computer programming. It is written for high-school graduates who have had no previous experience with computers or BASIC programming.

The proper performance of contemporary computer programming requires not only familiarity with a computer and knowledge of a programming language such as BASIC, but also an understanding of problem solving, structured programming, and modular design. Furthermore, it calls for an awareness of the tools and procedures for designing output, the techniques for developing programs that are interactive and menu-driven, and the methods for processing arrays, data files, and strings. This textbook covers all these essentials and much more.

Chapter 1 provides a concise textual and pictorial introduction to computers and programming, pointing out the interrelated roles of computers, programs, data, and people. Chapter 2 covers the fundamentals of modern problem solving in more depth than most other BASIC programming textbooks, beginning with how to define problems and emphasizing the stepwise refinement procedure for designing algorithms to express problem solutions. Chapter 3 explains how to run BASIC programs on a computer after presenting a simple, useful BASIC program that follows directly from an algorithm designed in chapter 2. Students are encouraged to run the program while studying the chapter. Then, at the end of the chapter, they are given the opportunity of running several other uncomplicated programs.

BASIC language instruction begins in chapter 4, where the fundamental statements necessary for writing a complete program are described and illustrated. Students are expected to start programming individually while studying this chapter.

In chapters 5 and 6 the fundamental and built-in control structures of BASIC are described, and structured programming is stressed. Then, after one-dimensional arrays, subscripted variables, and sorting are covered in chapter 7, modular design is formally introduced in chapter 8, along with the subroutine and function features of BASIC. Data file processes, designed and formatted output, string processing, two-dimensional arrays, and matrix processes are the subjects of chapters 9 through 13.

Ideally, the chapters should be studied in sequence; most of them build on the material of the preceding chapters. However, the following variations from sequential study are practicable: Students who have had a general introduction to computers may begin with chapter 2 or 3; chapter 3 can be studied after chapter 1; and chapter 12 can follow chapter 7. The entire book can be covered in a one-semester or one-quarter BASIC programming course. For shorter courses or courses of which BASIC programming is only a part, chapters 1 through 8 can be used to provide a solid foundational knowledge of BASIC programming; then, time permitting, the study of chapter 9 and/or chapter 12 will add substantially to that foundation.

In addition to those already mentioned, this text also offers the following features:

The use of a program-design language (PDL) for presenting algorithms. The PDL is a succinct, structured pseudocode that is independent of any programming language or computer. It serves as a convenient mechanism for expressing algorithms in their final refined form.

A complete presentation of program flowcharting. Although it is pointed out in the text that BASIC programs can be written from PDL algorithms, standard flowcharting is also completely described and illustrated in chapter appendices because of the importance of the flowchart as the traditional means of expressing algorithms prior to program coding. The flowcharting material corresponds directly with the coverage of the chapter it accompanies and can therefore be used in conjunction with it.

Early coverage (chapter 4) of external input data and the INPUT statement, along with later coverage (chapter 7) of internal data and the READ/DATA statements. This arrangement provides students with a correct view of how input data are actually treated in practice.

Many example algorithms and programs that are designed to solve a wide variety of practical problems. In chapter 2 and chapters 4 through 13, case study examples are provided in individual sections. These examples are completely developed in the text, from problem conception through algorithm and program coding, thereby providing students with thoroughly documented, readable, and structured algorithm and programming models for each major programming topic. They illustrate programming techniques for interactive execution, sorting, searching, table handling, menu-driven processing, file updating, file inquiry handling, report generation, editing, code conversion, curve fitting, and other processes.

Summary and Key Term sections in every chapter. Each Summary section provides a review of the important concepts in the chapter. Each Key Term section consists of a list of the important words defined in the chapter. The definitions are restated, for study purposes, in the Glossary at the back of the book.

BASIC Statement Review Sections toward the end of each chapter, beginning with chapter 4. Included for ready reference and review, each of these sections contains the syntactical form, purpose, and section cross-reference for every BASIC statement covered in the chapter.

Self-Assessment Tests, consisting of review questions and short-answer exercises, in each chapter and in the flowcharting appendices. The test answers are then given in the section Self-Assessment Test Answers, which is located at the back of the book.

Practical assignments in each chapter. These let students apply the concepts presented in that and earlier chapters. Some of the assignments call for computer-store visits, computer center tours, computer magazine reading, and execution of prewritten programs provided in the book. The vast majority, however, require that algorithms be designed and that programs be written. These assignments provide the practice necessary for a person to become adept at programming in BASIC.

Complete coverage of Standard BASIC (American National Standard for minimal BASIC),[1] plus extensions that have widespread use. Programs written with only Standard BASIC, which is presented in chapters 4 through 8 and 12, are transportable; that is, they will run with little or no change on virtually any computer that has a BASIC language processor.

Three appendices for reference and review: "BASIC Command Summary," "BASIC Statement Summary," and "BASIC Function Summary."

Two additional appendices: "Using Microsoft BASIC"[2] and "Using Applesoft BASIC." These present two BASIC Language dialects that are used extensively in microcomputers. The first describes BASIC for the IBM Personal Computer (PC)[3] and other computers that are IBM PC-compatible. The second describes BASIC for Apple[4] computers.

An instructor's manual is available from the publisher. The manual contains the following information for each chapter: teaching and assignment recommendations, solutions for the assignments, and suggested exam questions with answers. It also contains an appropriate selection of transparency masters. I prepared the manual with the same care as the text in order to help instructors teach modern BASIC programming in a complete and convenient manner.

Jarrell C. Grout
Nacogdoches, Texas

1. ANSI X3.60-1978, American National Standards Institute, 1430 Broadway, New York, 1978.
2. Microsoft is a registered trademark of Microsoft Corporation.
3. IBM PC is a registered trademark of International Business Machines Corporation.
4. Apple is a registered trademark of Apple Computer, Inc.

Acknowledgements

This book began as teaching material for the introductory computing course at Stephen F. Austin State University (SFASU), a course taken by a wide variety of majors including entry-level computer science majors with no computer background. Thus, a large number of students have used several versions of the book in manuscript form over the past few years. I appreciate the proofing they performed and the suggestions they made as they studied and applied the subject matter.

I thank Dr. Craig A. Wood, Chairman of the SFASU Department of Computer Science, for scheduling my classes in a way that allowed me to teach from the material and then have the time to develop it into a complete textbook. I appreciate the technical assistance of several SFASU computer center personnel, including Morris Lang, Mary Burton, Glenn Millard, and Gary Earle, as well as Gary's fine staff.

I am grateful to the following reviewers for taking the time to examine as many as three versions of the manuscript in depth and then to offer valuable suggestions and recommendations for improving the work: Norman Lindquist, St. Andrews College; Dr. Biplab K. Dutta, State University of New York, Buffalo; Dr. James Reed III, University of Akron; Pasha A. Rostov, California Polytechnic State University; Arline Sachs, Northern Virginia Community College; Charles M. Williams, Georgia State University.

Contents

Preface ix
Acknowledgments xiii

1 Computers and Programming 1

1.1 Characteristics of Computers 2
1.2 The Nature of Computer Programs 10
1.3 The Parts Played by Data 11
1.4 The Roles People Play 11
1.5 Summary 12
1.6 Key Terms 12
1.7 Self-Assessment Test 13
1.8 Assignments 14

2 Problem-Solving Fundamentals 15

2.1 Defining Problems 16
2.2 Assembling Documentation 17
2.3 Preparing Test Input Data 17
2.4 Devising and Specifying Algorithms 17
2.5 An Exam Average Example 22
2.6 Summary 26
2.7 Key Terms 26
2.8 Self-Assessment Test 26
2.9 Algorithmic Assignments 28
Appendix 2A: Developing Program Flowcharts 30
Appendix 2B: Self-Assessment Test on Flowcharts 35
Appendix 2C: Flowcharting Assignments 36

3 Program Processing Essentials 37

3.1 Building a Program File 39
3.2 Interactive Processing 39
3.3 Batch Processing 42
3.3.1 Batch Execution of Programs Stored on Disk or Tape 43
3.3.2 Batch Execution of Programs Punched Into Cards 43
3.4 Summary 44
3.5 Key Terms 46
3.6 Self-Assessment Test 46
3.7 Program Execution Assignments 47

4 Primary BASIC Concepts 51

4.1 Characters, Numbers, Strings, and Variables 52
4.2 Six Fundamental Statements 00
4.2.1 REM 55
4.2.2 INPUT 55
4.2.3 LET 56
4.2.4 PRINT 58
4.2.5 STOP 61
4.2.6 END 61
4.3 Input Prompting for Interactive Execution 61
4.4 Program Debugging and Testing 62
4.5 An Interactive Conversion Example 64
4.6 Summary 66
4.7 BASIC Statement Review 67
4.8 Key Terms 67
4.9 Self-Assessment Test 68
4.10 Programming Assignments 69
Appendix 4A: The Interactive Conversion Program Flowchart 70
Appendix 4B: Flowcharting Assignments 70

5 Fundamental Control Structures 73

5.1 Structured Programming 74
5.2 Selection with the IF-THEN and GOTO Statements 77
5.3 Repetition with the IF-THEN and GOTO Statements 81
5.4 Repetition with the DO WHILE and END DO Statements 83
5.5 A Table Production Example 84
5.6 Summary 89
5.7 BASIC Statement Review 89
5.8 Key Terms 90
5.9 Self-Assessment Test 90
5.10 Programming Assignments 92
Appendix 5A: Flowcharting the Fundamental Structures 94
Appendix 5B: Self-Assessment Test on Flowcharting 98
Appendix 5C: Flowcharting Assignments 98

6 Built-In Control Structures 99

6.1 Case: The ON-GOTO Statement 100
6.2 Repetition: FOR/NEXT Loops 102
6.3 A Summarization Example 105
6.4 Summary 110
6.5 BASIC Statement Review 110
6.6 Key Terms 110
6.7 Self-Assessment Test 110
6.8 Programming Assignments 112
Appendix 6A: Flowcharting the Built-In Structures 113
Appendix 6B: Self-Assessment Test on Flowcharting 113
Appendix 6C: Flowcharting Assignments 113

7 Arrays and Subscripted Variables 117

7.1 Why Arrays Are Necessary 118
7.2 One-Dimensional Arrays 119
7.3 Defining Lower Bounds: The OPTION Statement 123
7.4 Loading and Searching Arrays 124
7.5 A Sorting Example 128
7.6 Summary 132
7.7 BASIC Statement Review 134
7.8 Key Terms 134
7.9 Self-Assessment Test 134
7.10 Programming Assignments 136

Appendix 7A: Flowcharting for Arrays 143
Appendix 7B: Self-Assessment Test on Flowcharting 143
Appendix 7C: Flowcharting Assignments 143

8 Modular Design, Subroutines, and Functions 145

8.1 Modular Design 146
8.2 Subroutines: The GOSUB and RETURN Statements 149
8.3 Implementation-Supplied Functions 152
8.4 User-Defined Functions: The DEF Statement 158
8.5 A Menu-Driven Example 159
8.6 Summary 171
8.7 BASIC Statement Review 172
8.8 Key Terms 172
8.9 Self-Assessment Test 172
8.10 Programming Assignments 173
8.11 Conclusion of Standard BASIC 175
Appendix 8A: Flowcharting Modular Programs 176
Appendix 8B: Self-Assessment Test on Flowcharting 178
Appendix 8C: Flowcharting Assignments 178

9 Data File Processes 179

9.1 Data File Concepts 180
9.2 Sequential File Processes 182
9.3 A File Update Example 184
9.4 Random Access File Processes 201
9.5 A File Query Example 205
9.6 Guidelines 207
9.7 Summary 208
9.8 BASIC Statement Review 209
9.9 Key Terms 209
9.10 Self-Assessment Test 209
9.11 Programming Assignments 210
Appendix 9A: Flowcharting for File Processing 212
Appendix 9B: Self-Assessment Test on Flowcharting 212
Appendix 9C: Flowcharting Assignments 212

10 Printed Output by Design 215

10.1 Design Tools and Procedures 217
10.2 Controlled Printing 220
10.3 A Report Production Example 223
10.4 Summary 228
10.5 BASIC Statement Review 228
10.6 Key Terms 229
10.7 Self-Assessment Test 229
10.8 Programming Assignments 230

11 Additional String Processing 233

11.1 Input 234
11.2 Editing 235
11.3 An Editing Example 237
11.4 Conversions 241
11.5 A Code Conversion Example 244
11.6 Summary 245
11.7 BASIC Statement Review 246
11.8 Key Terms 246
11.9 Self-Assessment Test 247
11.10 Programming Assignments 248

12 Two-Dimensional Arrays 251

12.1 Declarations 252
12.2 Processing 253
12.3 Another Table Production Example 256
12.4 Alternative Designations for Arrays 260
12.5 Summary 261
12.6 BASIC Statement Review 261
12.7 Key Terms 261
12.8 Self-Assessment Test 261
12.9 Programming Assignments 263

13 Matrix Processes 267

13.1 Input/Output 268
13.2 Computations 272
13.3 A Curve-Fitting Example 277
13.4 Summary 281
13.5 BASIC Statement Review 282
13.6 Key Terms 282
13.7 Self-Assessment Test 282
13.8 Programming Assignments 284

Appendixes 287

A: BASIC Command Summary 289
B: BASIC Statement Summary 291
C: BASIC Function Summary 297
D: Using Microsoft BASIC 299
E: Using Applesoft BASIC 309

Self-Assessment Test Answers 317
Glossary 351
Index 359

Computers and Programming 1

1.1 Characteristics of Computers
1.2 The Nature of Computer Programs
1.3 The Parts Played by Data
1.4 The Roles People Play
1.5 Summary
1.6 Key Terms
1.7 Self-Assessment Test
1.8 Assignments

The purpose of this book is to acquaint you with computer programming and the programming language known as **BASIC** (Beginner's All-purpose Symbolic Instruction Code). BASIC is one of several languages that people can use to program computers. Relative to the others, it is one of the simplest to learn; at the same time, it is a very capable language. Furthermore, BASIC is usable with virtually every computer and is the language that has the most widespread use with personal computers—the small computers employed in offices and homes.

This first chapter provides essential background information. It describes the relevant attributes, interactions, and roles of computers, programs, data, and people.

1.1 Characteristics of Computers

A **computer** is an electronic machine that performs certain actions in accordance with a set of specially written instructions. It works extremely fast—at electronic speeds that cannot even be imagined by most people. Furthermore, it is capable of accomplishing repetitive, boring tasks exactly as instructed for as long as necessary.

The actions performed include accepting and storing instructions, numeric **data,** and other types of data, such as character sequences (letters, digits, and other symbols) called **strings;** adding, subtracting, multiplying, dividing, comparing, and replacing numbers; comparing and modifying strings; displaying the results in a form that people can use; and saving the results in a form that the computer can use again. The specially written instructions constitute a **computer program.** They are prepared in a special type of language, known as a **programming language,** that has been designed for the computer to accept and use. BASIC is one of these languages.

As shown in Figure 1.1, a computer consists of five primary components: **input, memory, control, arithmetic-logic,** and **output.** The input component receives instructions and data from outside the computer and transmits them to memory. These activities—receiving from outside and transmitting to memory—are collectively referred to as **inputting** or **reading.**

Memory, or **primary storage,** is designed to hold the data and instructions temporarily until they are needed by one of the other components. Control directs the activities of the other components by fetching the instructions from memory in the proper sequence, interpreting each instruction to determine which action is to be performed, and then initiating the performance by the proper component. The performance of an action is called **execution.**

Arithmetic-logic executes instructions that contain arithmetic operations or comparisons. The arithmetic operations produce computed results; the comparisons are used in making decisions.

Output accepts data (i.e., results) from memory and either displays them for use by people or saves them for later use by the computer. The output process is known as **printing** or **writing.**

The items of equipment that physically constitute these components are referred to as **computer hardware.** They all require electrical power. Control, arithmetic-logic, and memory are completely electronic. The control and arithmetic-logic components are placed together and designated as the **central processing unit (CPU).**

The input and output components virtually always consist of several different items of equipment and, in fact, usually share some devices that have dual **input/output (I/O)** capabilities. All I/O devices are constructed of electronic and mechanical parts. The most common dual capability devices—**computer terminals** with typewriter-like keyboards, **magnetic disk** drives, and **magnetic tape** drives—are pictured in Figures 1.2 through 1.4. The most widely used input-only device (**punched card reader**) and output-only device (**line printer**) are shown in Figures 1.5 and 1.6. Figures 1.3 through 1.5 also contain photographs of the data recording medium used with each device: **disk pack** with disk drive, **tape reel** with tape drive, and **punched card** with card reader.

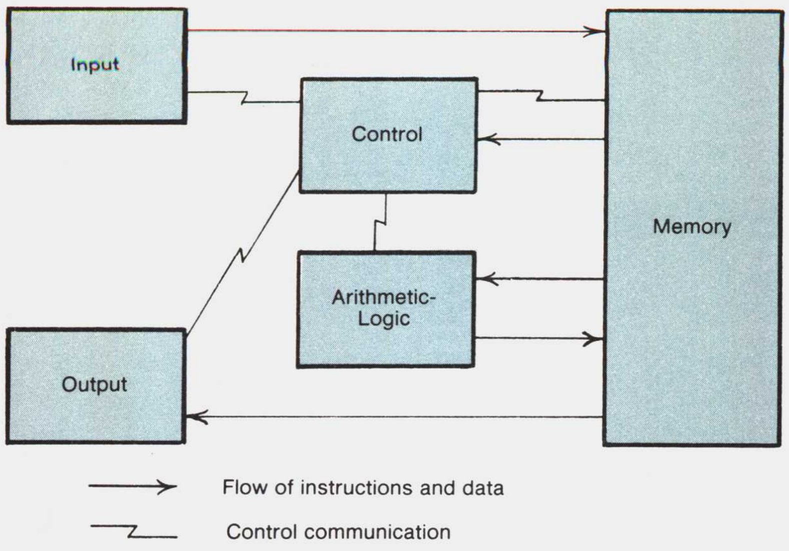
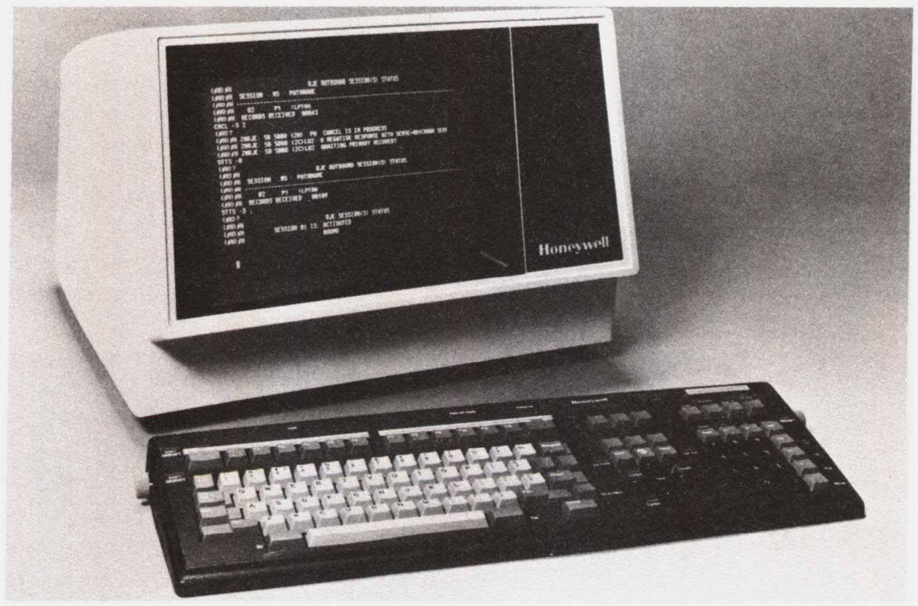


Figure 1.1
The primary components of a computer

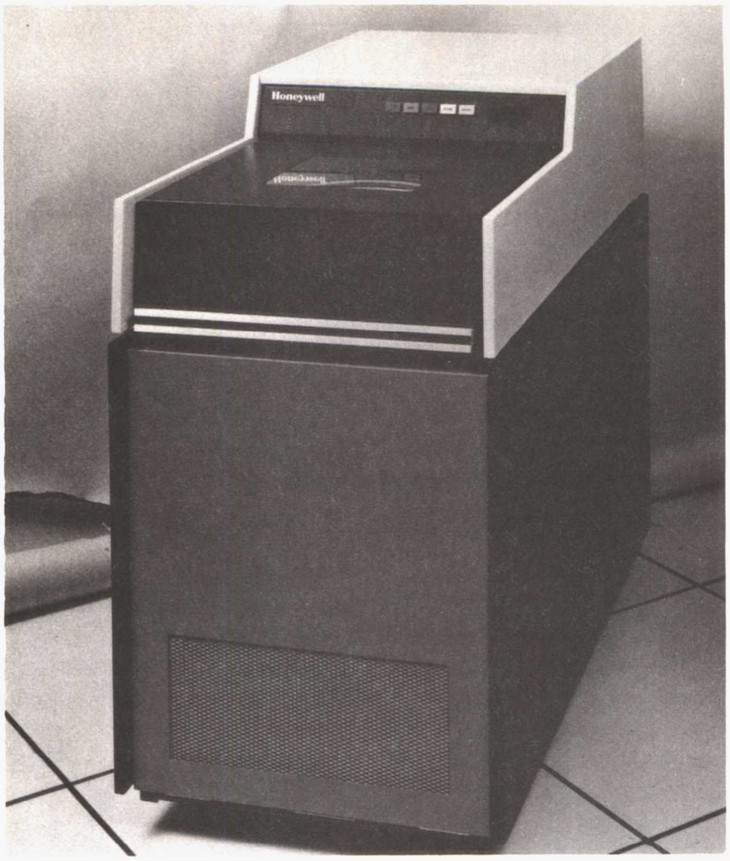
Figure 1.2
Computer terminals with keyboards: dual I/O capability



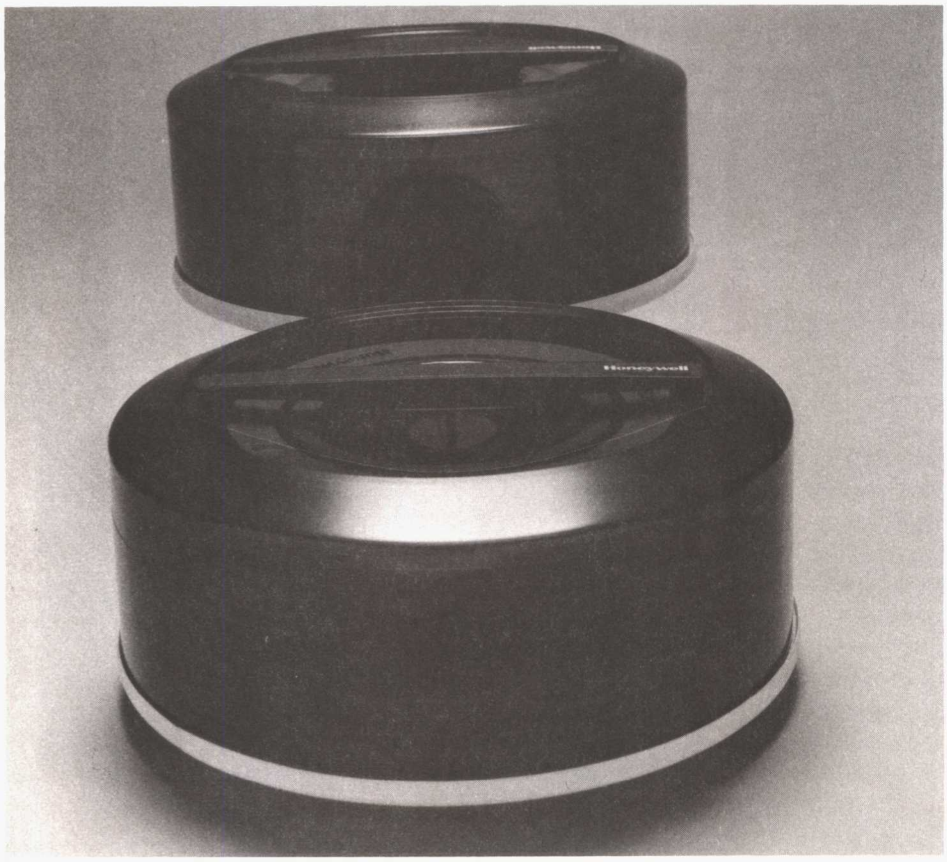
a. Video Display Unit (Courtesy of Honeywell Information Systems, Inc.)

b. Printing Terminal (Photo courtesy of Digital Equipment Corporation)

Figure 1.3
Magnetic disk drive and packs: dual I/O as well as secondary storage capability

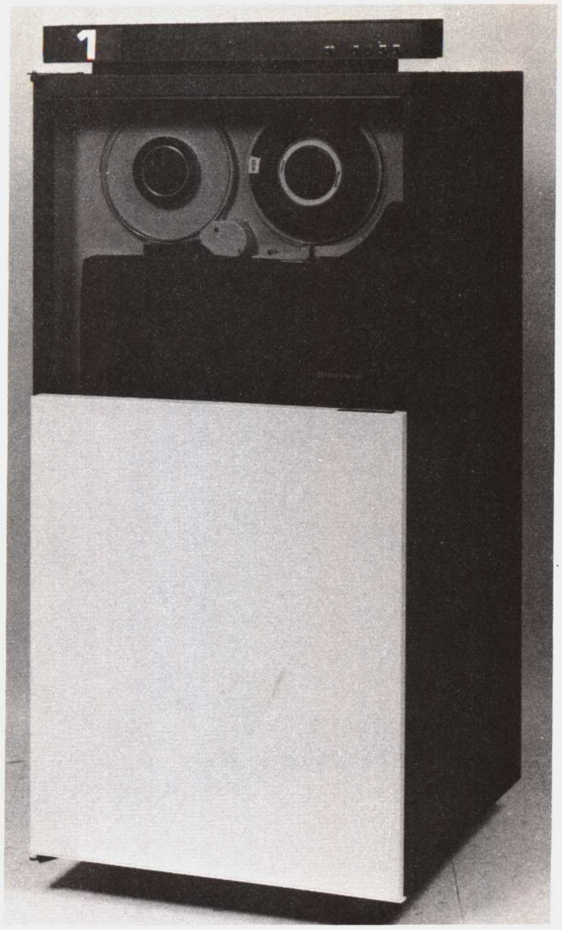


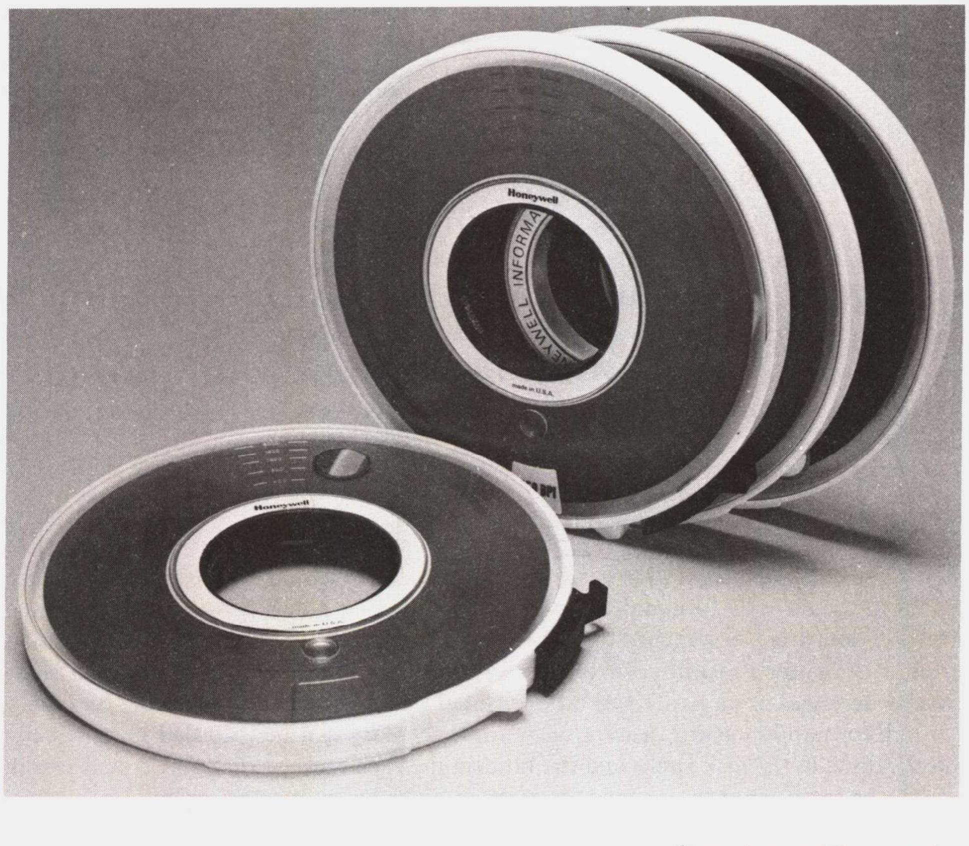
a. Drive (Courtesy of Honeywell Information Systems, Inc.) A Device for I/O Using Disk Packs



b. Packs (Courtesy of Honeywell Information Systems, Inc.) Media for Information Storage and I/O via Disk Drives

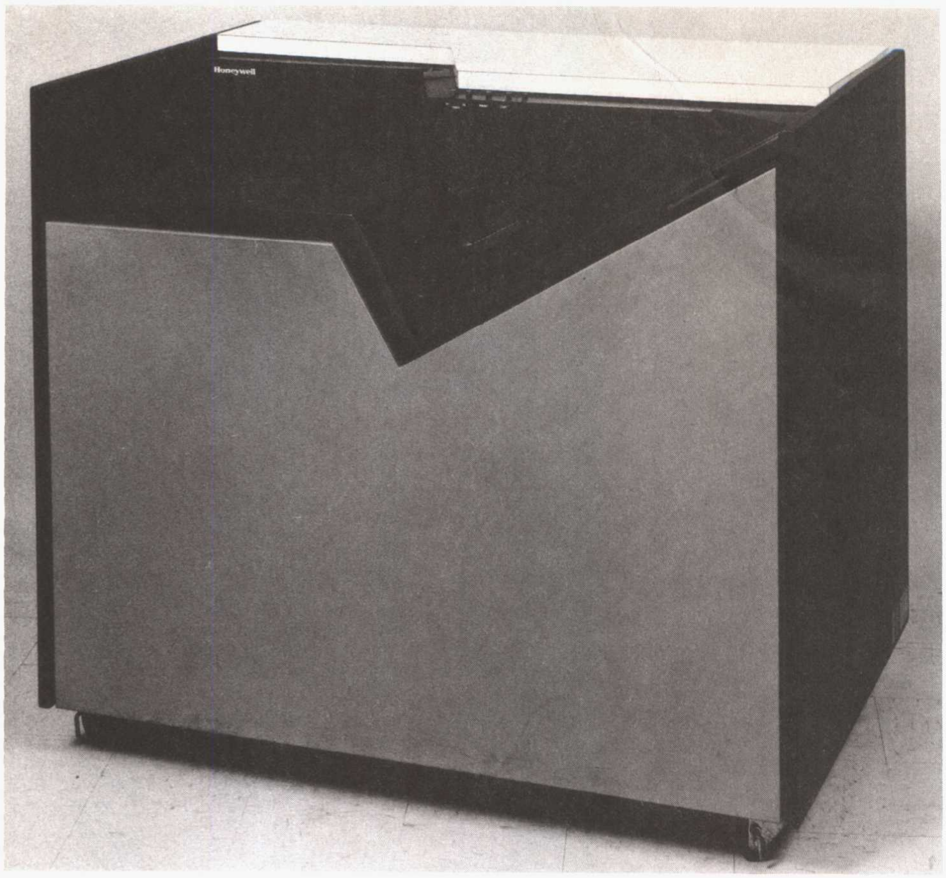
Figure 1.4
Magnetic tape drive and reels: dual I/O as well as secondary storage capability


a. Drive (Courtesy of Honeywell Information Systems, Inc.)
A Device for I/O Using Tape Reels

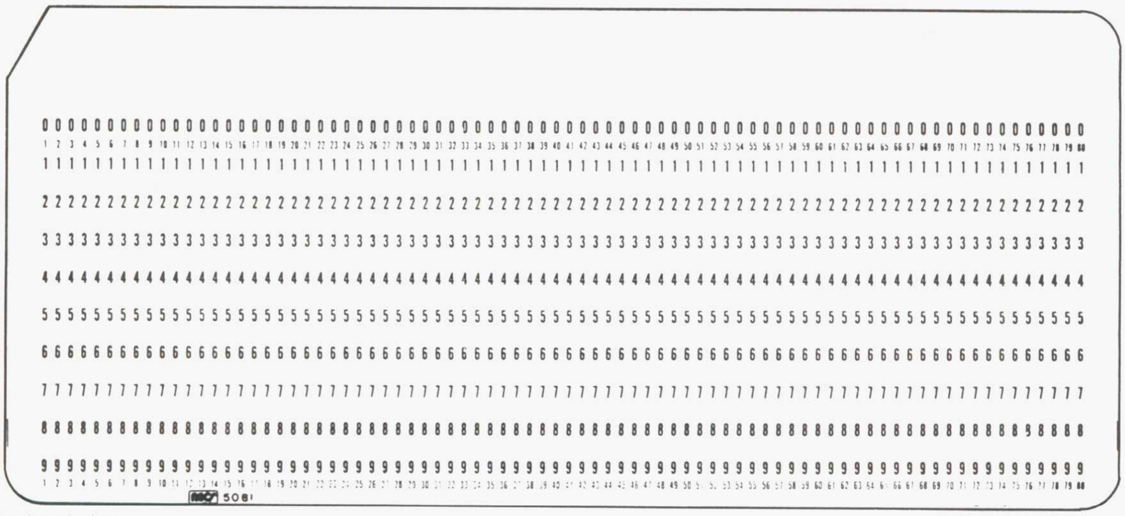


b. Reels (Courtesy of Honeywell Information Systems, Inc.)
Media for Information Storage and I/O via Tape Drives

Figure 1.5
Card reader and punched card: input only



a. Reader (Courtesy of Honeywell Information Systems, Inc.)
A Device for Input Using Punched Cards



b. Card A Medium for Input via a Card Reader

The entry of information into a computer's memory is accomplished with the use of the keyboard on a terminal, a magnetic disk pack in a disk drive, a magnetic tape reel in a tape drive, or punched cards through a card reader. The display of information from a computer's memory for direct use by people occurs on the screen of a **video display terminal** or on paper, the latter printed by a line printer or **printing terminal.**

Information coming from a computer's memory can be recorded magnetically on the disks in the disk packs and the tape on the tape reels by disk and tape drives in much the same manner as sound is recorded by ordinary tape recorders. Such infor-

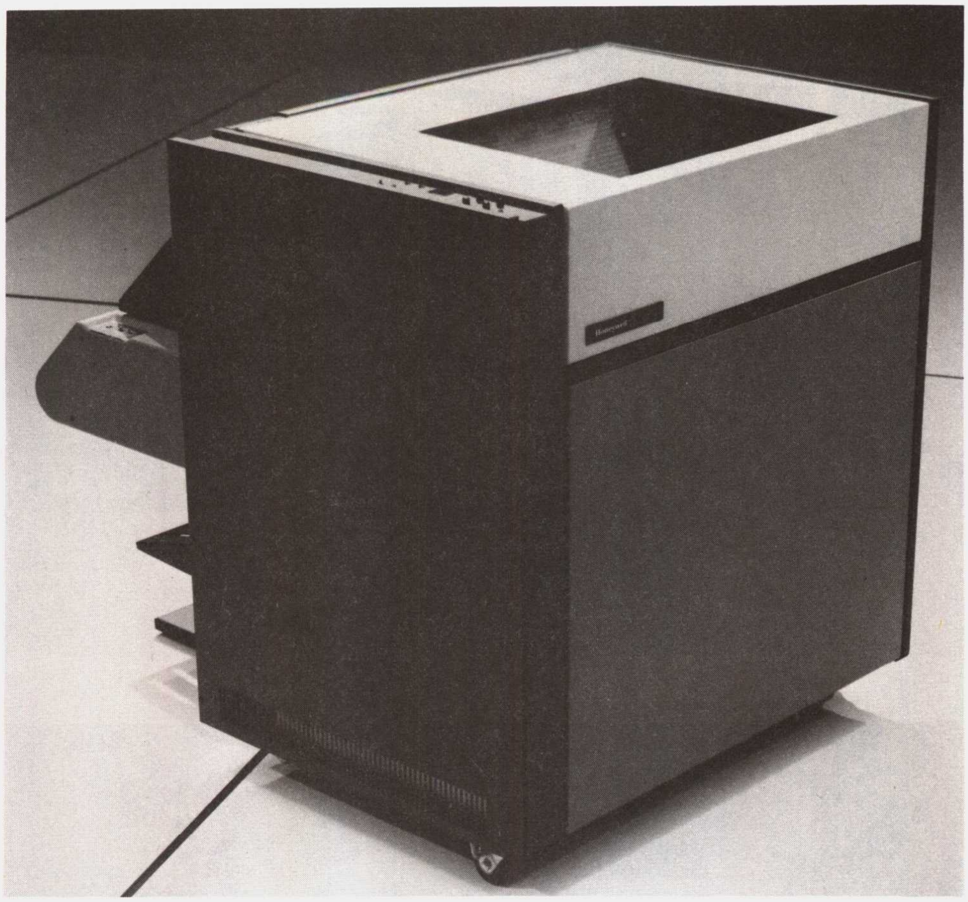


Figure 1.6
Line printer with continuous form paper: output only (Courtesy of Honeywell Information Systems, Inc.)

mation is said to be stored for later use by the computer. Thus magnetic disk and tape devices are also called **secondary storage** devices.

The hardware devices of a computer are connected by electrical cables and other linking mechanisms designed for transmitting information. Some of the devices are located near the CPU; others are remote. Terminals of the type shown in Figure 1.2a, for instance, are often **remote.** Furthermore, several of them may be connected to the same CPU in such a way as to allow a number of people located in different places to use the same computer during the same time period—that is, to engage in computer **time sharing.** The computer operates so fast, though, that time sharing does not seem like sharing at all; it usually seems like having your own computer.

Time sharing is the operational mode by which you may use the computer as you do the programming assignments of this book. Even if you do your work on a small computer without time-sharing capability, you will probably use a terminal that has a keyboard and display screen; your working environment, then, will be similar to that of time sharing. On the other hand, if you do not have access to a terminal, you will use punched cards and enter your information through a card reader. In this case, the operational mode is **batch processing,** so named because similar programs are collected and processed in groups.

Figure 1.7 pictures a **large-scale computer** that may have many remotely located terminals and card readers in operation at the same time. The smaller scale computer in Figure 1.8, a **minicomputer,** can also have a number of remote terminals in operation at the same time. Figure 1.9 shows two **microcomputers,** i.e., **small-scale** or **personal computers.** Microcomputers ordinarily have only one keyboard and one video display screen each. They usually are not designed for remote terminal operations; rather, they handle the work of one person at a time as that person enters information through the single available keyboard.

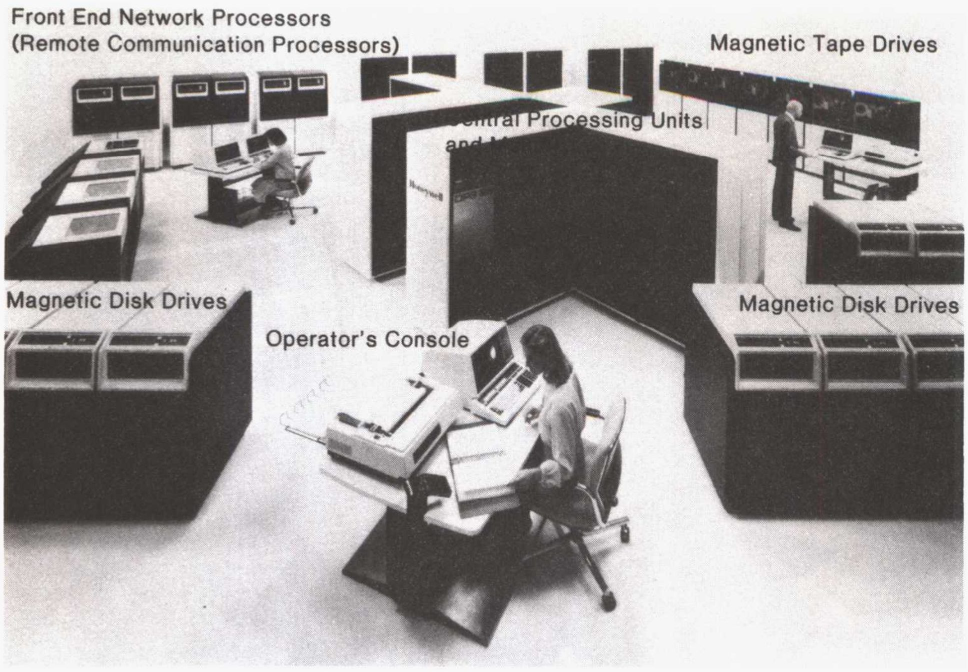


Figure 1.7
A large-scale computer system with time-sharing and batch-processing capabilities (Courtesy of Honeywell Information Systems, Inc.)

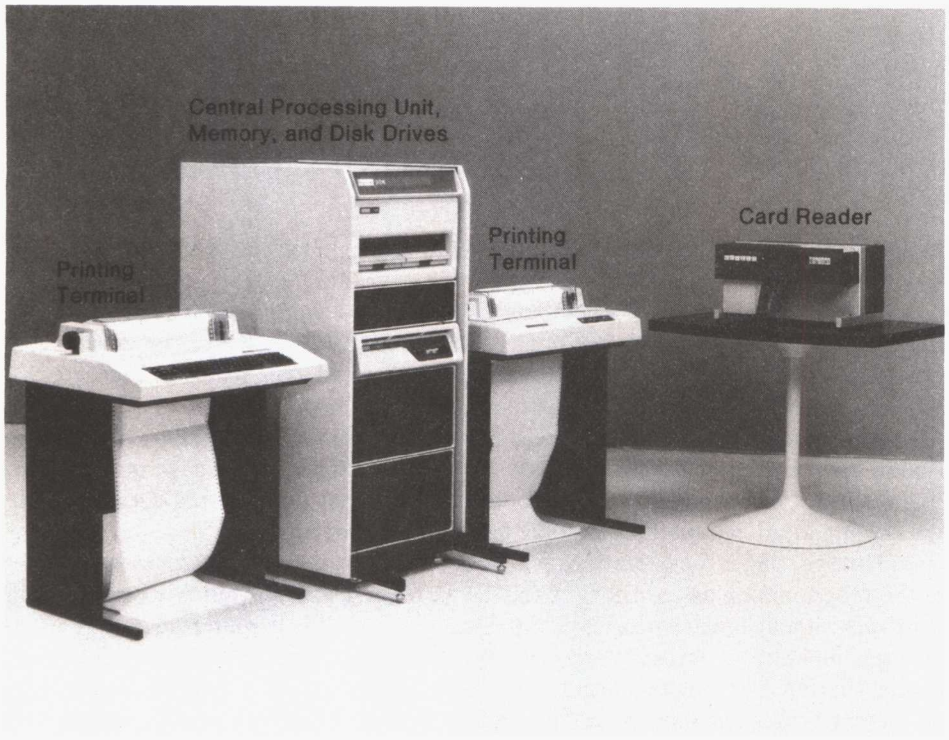


Figure 1.8
A minicomputer with time-sharing capability (Photo courtesy of Digital Equipment Corporation)