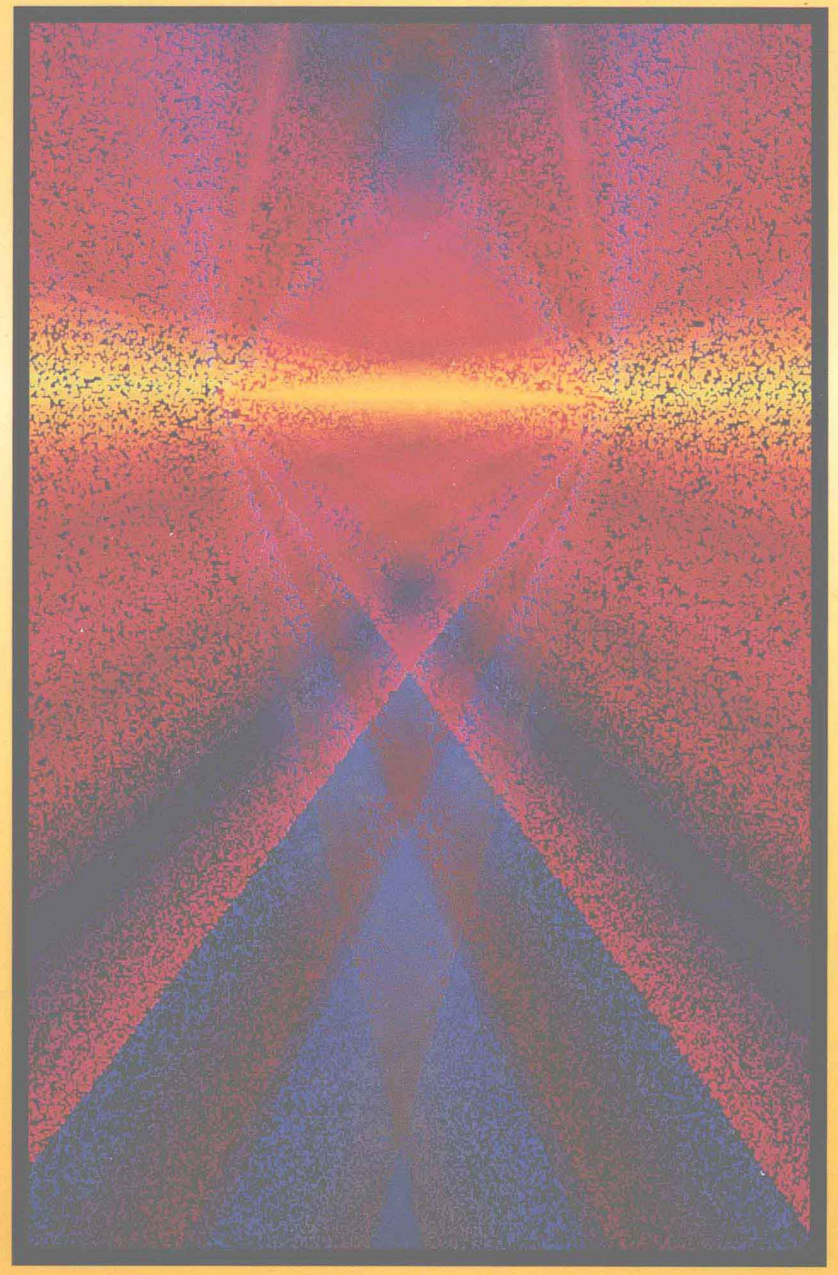
Microprocessor Architecture, Programming, and Applications with the 8085

Fourth Edition



Ramesh S. Gaonkar

Microprocessor Architecture, Programming, and Applications with the 8085

FOURTH EDITION

Ramesh S. Gaonkar
STATE UNIVERSITY OF NEW YORK,
O.C.C. CAMPUS AT SYRACUSE

Prentice Hall
Upper Saddle River, New Jersey *Columbus, Ohio*

Library of Congress Cataloging-in-Publication Data

Gaonkar, Ramesh S.
Microprocessor architecture, programming, and applications with the 8085 / Ramesh S. Gaonkar. — 4th ed.
p. cm.
First-2nd eds. published under title: Microprocessor architecture, programming, and applications with the 8085/8080A.
Includes index.
ISBN 0–13–901257–5 (alk. paper).
1. Microprocessors. 2. Computer architecture. I. Gaonkar, Ramesh S. Microprocessor architecture, programming, and applications with the 8085/8080A, II. Title.
TK7895.M5G36 1999 98–45804
004.165—dc21 CIP

Cover photo: © H. Armstrong Roberts
Editor: Charles E. Stewart, Jr.
Production Editor: Alexandrina Benedicto Wolf
Design Coordinator: Karrie M. Converse
Cover Designer: Jason Moore
Production Manager: Deidra M. Schwartz
Marketing Manager: Ben Leonard

This book was set in Times Roman by The Clarinda Company and was printed and bound by R. R. Donnelley & Sons Company. The cover was printed by Phoenix Color Corp.

Printed in the United States of America

10 9 8 7 6 5 4 3

ISBN: 0-13-901257-5

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada, Inc., *Toronto*
Prentice-Hall Hispanoamericana, S. A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Preface

By the time this fourth edition is published, this microprocessor textbook will have been in the field for fifteen years. It is gratifying to see such acceptance of the integrated approach to teaching microprocessor concepts. The text is intended for introductory microprocessor courses at the undergraduate level in technology and engineering. It is a comprehensive treatment of the microprocessor, covering both hardware and software based on the 8085 microprocessor family. The text assumes a course in digital logic as a prerequisite; however, it does not assume any background in programming. At the outset, though, we need to answer the following three critical questions.

1. *In the late 1990s, is an 8-bit microprocessor an appropriate device through which to teach microprocessor concepts when 32-bit microprocessors are readily available?* If we consider the worldwide sales volume of microprocessor chips, the answer is a resounding *yes*: 8-bit microprocessors (including single-chip microcontrollers) account for more than 85 percent of the total. Consider an analogy from the auto industry. For transportation, we need trucks, sports cars, family cars, and compact cars. Each serves a different purpose. The 8-bit microprocessor has already established its market in the areas of industrial control, such as machine control, process control, instrumentation, and consumer appliances; these systems that include a microprocessor are known as embedded systems or microprocessor-based products. The recent 32-bit microprocessors are used primarily in microcomputers and workstations; they are so powerful that their applications are better suited in such areas as high-speed data processing, CAD/CAM, multitasking, and multiuser systems. The 32-bit microprocessors are less likely to replace 8-bit microprocessors in industrial control applications. In many applications, even 8-bit microprocessors are utilized at less than 50 percent of their capacity.

We are interested in teaching the basic concepts underlying a programmable device, such as buses, machine cycles, various processes of data flow (parallel, serial, interrupts, and DMA), internal register architecture, programming, and interfacing. A general-purpose 8-bit microprocessor is an ideal device to teach these concepts, especially in a rapidly changing technological environment. When students master the basic concepts, they will be able to apply those concepts in such an environment, whether it is based on a microcontroller, an 8-bit processor with a different set of instructions, or a 32-bit processor.

2. *Why shouldn't we focus on the Intel 16-bit processor when PCs (personal computers) are commonly available in college laboratories?* To teach basic concepts, we need a simple processor with an adequate instruction set. The Intel 16-bit processors are too complex at the introductory level because of the concepts involved in memory segmentation and a large instruction set, which is suited for high-level languages. These 16-bit processors were used primarily in PCs, and they are being replaced by 32-bit processors. Technologically, they are obsolete as general-purpose processors; they may be revived as 16-bit microcontrollers.

3. *Why teach the 8085 microprocessor?* Any commonly available 8-bit microprocessor will meet the teaching criteria, and the 8085 is one of the most widely used microprocessors in college laboratories. It has simple architecture and an adequate instruction set, which enable instructors to teach necessary programming concepts. It is inconsequential which microprocessor is selected as the focus; the concepts are easily transferable from one device to another. Having learned basic concepts with the 8085 microprocessor, students can adapt to the microcontroller (such as the Intel 8051 or Motorola 68HC11) environment or to the PC environment. Furthermore, peripheral devices (such as the 8255A, 8254, and 8259) are used commonly in the PC environment. One can argue for a microcontroller as a basis for an introductory course. However, the experiences of many institutions suggest that a microcontroller is an appropriate device for a higher-level course; at an introductory level, the pedagogy becomes quite cumbersome. Furthermore, general-purpose 8-bit processors are being used in small systems. As an example, some Texas Instruments graphic calculators use the 8-bit Z80 processor.

PEDAGOGICAL APPROACH AND TEXT ORGANIZATION

The microprocessor is a general-purpose programmable logic device. A thorough understanding of the microprocessor demands concepts and skills from two different disciplines: hardware concepts from electronics and programming skills from computer science. Hardware is the physical structure of the microprocessor, and programming makes it come alive; one without the other is meaningless. Therefore, this text presents an integrated approach to hardware and software in the context of the 8085 microprocessor. Part I focuses on microprocessor architecture and interfacing; Part II introduces programming; and Part III integrates hardware and software concepts from the earlier sections in interfacing and designing microprocessor-based products. Each topic is covered in depth from basic concepts to industrial applications and is illustrated by numerous examples with complete schematics. The material is supported with assignments of practical applications.

Part I has four chapters dealing with the hardware aspects of the microcomputer as a system, presented with the spiral approach similar to the view from an airplane that is getting ready to land. As the plane circles around, what one observes is a view without any details. As the plane starts descending, one begins to see the same view but with more details. This approach is preferable because students need to use a microcomputer as a system in their laboratory work in the early stages of a course, without having an understanding of all aspects of the system. Chapter 1 presents an overview of microprocessor-based systems. It presents the 8-bit microprocessor as a programmable device and an embedded controller, rather than a computing device or CPU used in computers. Chapters 2, 3, and 4 examine microprocessor architecture, memory, and I/O, with increasing depth in each chapter, from registers to instruction timing and interfacing.

Part II has seven chapters dealing with 8085 instructions, programming techniques, program development, and software development systems. The contents are presented in a step-by-step format. A few instructions that can perform a simple task are selected. Each instruction is described fully with illustrations of its operations and its effects on the selected flags. Then, these instructions are used in writing programs, accompanied by programming techniques and troubleshooting hints. Each illustrative program begins with a problem statement, provides the analysis of the problem, illustrates the program, and explains the programming steps. Chapters conclude with reviews of all the instructions discussed. The contents of Part II are presented in such a way that, in a course with heavy emphasis on hardware, students can teach themselves assembly language programming if necessary.

Part III synthesizes the hardware concepts of Part I and the software techniques of Part II. It

deals with interfacing of I/Os, with numerous industrial and practical examples. Each illustration analyzes the hardware, includes software, and describes how hardware and software work together to accomplish given objectives. Chapters 12 through 16 include various types of data transfer between the microprocessor and its peripherals such as interrupts, interfacing of data converters, I/O with handshake signals using programmable devices, and serial I/O. Chapter 14 discusses special-purpose programmable devices used primarily with the 8085 systems (such as the 8155), while Chapter 15 discusses general-purpose programmable devices (such as the 8255A, 8254, 8259, and 8237). Chapter 17 deals primarily with the project design of a single-board microcomputer that brings together all the concepts discussed in the text. Chapter 18 focuses on how to extend 8-bit microprocessor concepts to higher level processes and microcontrollers. It also discusses trends in microprocessor technology ranging from recent microcontrollers to the latest general-purpose 32-bit microprocessors.

NEW AND IMPROVED FEATURES IN THE FOURTH EDITION

The fourth edition preserves the focus as described and includes the following changes and additions, suggested by reviewers and by faculty who have used the book in their classrooms:

1. Chapter 1 is revised to include the most recent technological changes.
2. Part II (Chapters 5 through 11) has few changes in the content, except in Chapter 11, which is revised to include technological changes.
3. In Chapter 15, the DMA controller 8257 is replaced by the 8237, which is commonly used in newer systems. The 8237 is thoroughly discussed with illustrations.
4. Chapter 17 includes an additional interfacing application: how to interface an LCD module, which has become a popular display device in industrial systems.
5. Chapter 18 is updated to include the latest technological changes in 32-bit microprocessors.
6. In Appendix B, the Intel SDK-85 system is replaced by the EMAC Primer, a stand-alone single-board microcomputer system with a Hex keyboard and LED displays. Its enhanced version can be used with a PC. Students can write assembly language programs using an editor on a PC, assemble the programs, download the binary code from a PC to the Primer trainer, and execute the programs. EMAC Primer is a complete development system, ideally suited to an educational environment.
7. In Appendix D, complete data sheets for the 8237, 8259, and an LCD panel are included. These data sheets will enable students to perform many experiments outside the scope of this introductory text.

A WORD WITH FACULTY

This text is based on my teaching experience, my course development efforts, and my association with industry engineers and programmers. It is an attempt to share my classroom experiences and my observations in industrial practices. Some of my assumptions and observations of 15 years ago appear still valid today:

1. Software (instructions) is an integral part of the microprocessor and demands emphasis equal to that of the hardware.
2. In industry, for development of microprocessor-based projects, 70 percent of the effort is devoted to software and 30 percent to hardware.
3. Technology and engineering students tend to be hardware oriented and have considerable difficulty in programming.
4. Students have difficulty in understanding mnemonics and realizing the critical importance of flags.

In the last fifteen years, numerous faculty members shared their classroom experiences, concerns, and student difficulties with me through letters and

e-mail messages. I have made every effort to incorporate those concerns and suggestions in the fourth edition. This revised edition can be used flexibly to meet the objectives of various courses at the undergraduate level. If used for a one-semester course with 50 percent hardware and 50 percent software emphasis, the following chapters are recommended: Chapters 1 through 4 for hardware lectures and Chapters 5 through 9 and selected sections of Chapter 10 for software laboratory sessions. For interfacing, the initial sections of Chapters 12 and 16 (introduction to interrupts and serial I/O) are recommended. If the course is heavily oriented toward hardware, Chapters 1 through 4 and Chapters 12 through 17 are recommended, and necessary programs can be selected from Chapters 5 through 9. If the course is heavily oriented toward software, Chapters 1 through 11 and selected portions of Chapters 12 and 16 can be used. For a two-semester course, it is best to use the entire text.

A WORD WITH STUDENTS

Microprocessor technology is an exciting, challenging, and growing field; it will pervade industry for decades to come. To meet the challenges of this growing technology, you will have to be conversant with the programmable aspect of the microprocessor. Programming is a process of problem solving and communication in a strange language of mnemonics. Most often, hardware-oriented students find this communication process very difficult. One of the questions frequently asked by students is: How do I get started in a given programming assignment? One approach to learning programming is to examine various types of programs and imitate them. You can begin by studying the illustrative program relevant to an assignment, its flowchart, its analysis, program description, and particularly the comments. Read the instructions from Appendix F as necessary and pay attention to the flags. This text is written in such a way that simple programming aspects of the microprocessor can be self-taught. Once you master the elementary programming techniques, then interfacing and design become exciting and fun.

ACKNOWLEDGMENTS

My sincere thanks to my family members: my wife, Shaila, for her unwavering support and my daughters, Nelima and Vanita, for their assistance in completing various tasks. Several persons have made valuable contributions to this text. I would like to extend my sincere appreciation to my colleagues Charles Abate, James Delaney, and John Merrill, who offered many suggestions throughout the project, and Chris Conty, who initiated the project. Similarly, I appreciate the efforts and numerous suggestions of my reviewers: John Morgan from DeVry Institute, Peter Holsberg from Mercer Community College, David Hata from Portland Community College, and David Delkar from Kansas State University. If this text reads well, the credit goes to Gnomi Gouldin and to my colleague from the English Department, Dr. Kathy Forrest, who devoted painstaking hours to editing the rough draft of the first edition. For this fourth edition, I would like to express my sincere appreciation to the following reviewers of this edition who provided me with valuable comments and suggestions: Cheryl Schmidt, Florida Community College at Jacksonville; Michael Pelletier, Northern Essex Community College; and Ted Nguyen, San Jose City College. I also thank Alex Wolf, my production editor at Prentice Hall, and copy editor Sheryl Rose for their contributions to the text. I would appreciate any communication about the text from the reader. Please feel free to write or send an e-mail message.

Ramesh Gaonkar
State University of New York
O.C.C. Campus at Syracuse
Syracuse, New York 13215
E-mail: gaonkarr@sunyocc.edu

Contents

I

Microprocessor-Based Systems: Hardware and Interfacing

Part I of this book is concerned primarily with the microprocessor architecture in the context of microprocessor-based products. The microprocessor-based systems are discussed in terms of the three components—microprocessor, memory, and I/O (input/output)—and their communication process. The role of the programming languages, from machine language to higher-level languages, is presented in the context of the system.

The material is presented in a format similar to the view from an airplane that is getting ready to land. As the plane circles around, a passenger observes a view without any details. As the plane starts descending, the passenger begins to see the same view but with more details. In the same way, Chapter 1 presents the microprocessor from two points of view: the microprocessor as a programmable embedded device in a product and as an element of a computer system, and how it communicates with memory and I/O. The chapter also discusses the role of assembly language in microprocessor-based products and presents an overview of various types of computers—from large computers to microcomputers and their applications.

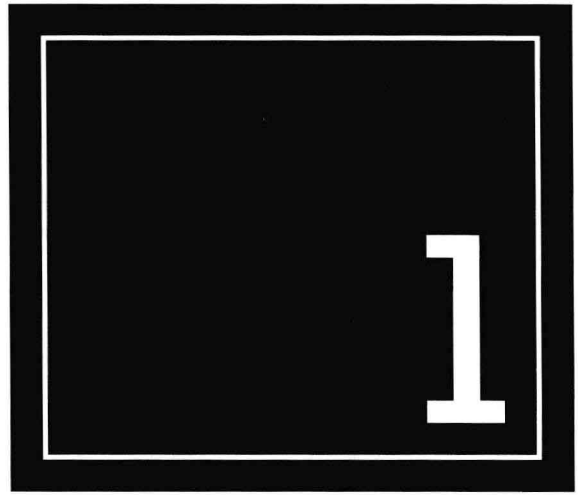
Chapter 2 provides a closer look at a microcomputer system in relation to the 8085 microprocessor. Chapter 3 examines the details of the 8085 microprocessor and memory interfacing. Chapter 4 discusses the interfacing of input/output (I/O) devices.

PREREQUISITES

The reader is expected to know the following concepts:

- ☐ Number systems (binary, octal, and hexadecimal) and their conversions.
- ☐ Boolean algebra, logic gates, flip-flops, and registers.
- ☐ Concepts in combinational and sequential logic.

Microprocessors, Microcomputers, and Assembly Language



The microprocessor plays a significant role in the everyday functioning of industrialized societies. The microprocessor can be viewed as a programmable logic device that can be used to control processes or to turn on/off devices. On the other hand, the microprocessor can be viewed as a data processing unit or a computing unit of a computer. The **microprocessor** is a programmable integrated device that has computing and decision-making capability similar to that of the central processing unit (CPU) of a computer. Nowadays, the microprocessor is being used in a wide range of products called microprocessor-based products or systems. The microprocessor can be embedded in a larger system, can be a stand alone unit controlling processes, or it can function as the CPU of a computer called a **microcomputer.** This chapter introduces the basic structure of a microprocessor-based product and shows how the same structure is applicable to microcomputers and other large (mini- and mainframe) computers. Later in the chapter, microprocessor applications are presented in the context of the entire spectrum of various computer applications.

The microprocessor communicates and operates in the binary numbers 0 and 1, called **bits.** Each microprocessor has a fixed set of instructions in the form of binary patterns called a **machine language.** However, it is difficult for humans to communicate in the language of 0s and 1s. Therefore, the binary instructions are given abbreviated names, called **mnenomics,** which form the **assembly language** for a given microprocessor. This chapter explains both the machine language and the assembly language of the microprocessor, known as the 8085. The advantages of assembly language are compared with high-level languages (such as BASIC, FORTRAN, C, and C++).

OBJECTIVES

- ☐ Draw a block diagram of a microprocessor-based system and explain the functions of each component: microprocessor, memory, and I/O, and their lines of communication (the bus).
- ☐ Explain the terms *SSI, MSI,* and *LSI.*
- ☐ Define the terms *bit, byte, word, instruction, software,* and *hardware.*
- ☐ Explain the difference between the machine language and the assembly language of a computer.
- ☐ Explain the terms *low-level* and *high-level languages.*
- ☐ Explain the advantages of an assembly language over high-level languages.
- ☐ Define the term *ASCII* code and explain the relationship between the binary code and alphanumeric characters.
- ☐ Define the term *operating system.*
- ☐ List components and peripherals of a typical personal computer (PC).

1.1 MICROPROCESSORS

A microprocessor is a multipurpose, programmable, clock-driven, register-based electronic device that reads binary instructions from a storage device called *memory,* accepts binary data as input and processes data according to those instructions, and provides results as output. A typical programmable machine can be represented with three components: microprocessor, memory, and I/O as shown in Figure 1.1. These three components work together or interact with each other to perform a given task; thus, they comprise a system. The physical components of this system are called **hardware.** A set of instructions written for the microprocessor to perform a task is called a **program,** and a group of programs is called **software.** The machine (system) represented in Figure 1.1 can be programmed to turn traffic lights on and off, compute mathematical functions, or keep track of a guidance system. This system may be simple or sophisticated, depending on its applications, and it is recognized by various names depending upon the purpose for which it is designed. The microprocessor applications are classified primarily in two categories: reprogrammable systems and embedded systems. In reprogrammable systems, such as microcomputers, the microprocessor is used for computing and data processing. These systems include general-purpose microprocessors capable of handling large data, mass storage devices (such as disks and CD-ROMs), and peripherals such as printers; a personal computer (PC) is a typical illustration. In embedded systems, the microprocessor is a part of a final product and is not available for reprogramming to the end user. A copy-

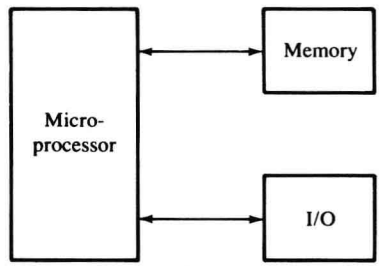


FIGURE 1.1
A Programmable Machine

ing machine is a typical example of an embedded system. The microprocessors used in these systems are generally categorized as: (1) **microcontrollers** that include all the components shown in Figure 1.1 on one chip, and (2) general-purpose microprocessors with discrete components shown in Figure 1.1. Embedded systems can also be viewed as products that use microprocessors to perform their operations; they are known as microprocessor-based products. Examples include a wide range of products such as washing machines, dishwashers, automobile dashboard controls, traffic light controllers, and automatic testing instruments.

BINARY DIGITS

The microprocessor operates in binary digits, 0 and 1, also known as bits. **Bit** is an abbreviation for the term *binary digit.* These digits are represented in terms of electrical voltages in the machine: Generally, 0 represents one voltage level, and 1 represents another. The digits 0 and 1 are also synonymous with low and high, respectively.

Each microprocessor recognizes and processes a group of bits called the *word,* and microprocessors are classified according to their word length. For example, a processor with an 8-bit word is known as an 8-bit microprocessor, and a processor with a 32-bit word is known as a 32-bit microprocessor.

A MICROPROCESSOR AS A PROGRAMMABLE DEVICE

The fact that the microprocessor is programmable means it can be instructed to perform given tasks within its capability. A piano is a programmable machine; it is capable of generating various kinds of tones based on the number of keys it has. A musician selects keys depending upon the musical score printed on a sheet. Similarly, today's microprocessor is designed to understand and execute many binary instructions. It is a multipurpose machine: It can be used to perform various sophisticated computing functions, as well as simple tasks such as turning devices on or off. A programmer can select appropriate instructions and ask the microprocessor to perform various tasks on a given set of data.

The person who designs a piano determines the frequency (tone) for a given key and the scope of the piano music. Similarly, the engineers designing a microprocessor determine a set of tasks the microprocessor should perform and design the necessary logic circuits, and provide the user with a list of the instructions the processor will understand. For example, an instruction for adding two numbers may look like a group of eight binary digits, such as 1000 0000. These instructions are simply a pattern of 0s and 1s. The user (programmer) selects instructions from the list and determines the sequence of execution for a given task. These instructions are entered or stored in storage, called *memory,* which can be read by the microprocessor.

MEMORY

Memory is like the pages of a notebook with space for a fixed number of binary numbers on each line. However, these pages are generally made of semiconductor material. Typically, each line is an 8-bit register that can store eight binary bits, and several of these registers are arranged in a sequence called memory. These registers are always grouped together in powers of two. For example, a group of $1024(2^{10})$ 8-bit registers on a semi-

conductor chip is known as 1K byte of memory; 1K is the closest approximation in thousands.* The user writes the necessary instructions and data in memory through an input device (described below), and asks the microprocessor to perform the given task and find an answer. The answer is generally displayed at an output device (described below) or stored in memory.

INPUT/OUTPUT

The user can enter instructions and data into memory through devices such as a keyboard or simple switches. These devices are called **input devices.** The microprocessor reads the instructions from the memory and processes the data according to those instructions. The result can be displayed by a device such as seven-segment LEDs (Light Emitting Diodes) or printed by a printer. These devices are called **output devices.**

MICROPROCESSOR AS A CPU (MPU)

We can also view the microprocessor as a primary component of a computer. Traditionally, the computer is represented in block diagram as shown in Figure 1.2(a). The block diagram shows that the computer has four components: Memory, Input, Output, and the central processing unit (CPU), which consists of the Arithmetic/Logic Unit (ALU) and Control Unit. The CPU contains various registers to store data, the ALU to perform arithmetic and logical operations, instruction decoders, counters, and control lines. The CPU reads instructions from the memory and performs the tasks specified. It communicates with input/output devices either to accept or to send data. These devices are also known as peripherals. The CPU is the primary and central player in communicating with devices such as memory, input, and output. However, the timing of the communication process is controlled by the group of circuits called the **control unit.**

In the late 1960s, the CPU was designed with discrete components on various boards. With the advent of the integrated circuit technology, it became possible to build the CPU on a single chip; this came to be known as a microprocessor, and the traditional block diagram shown in Figure 1.2(a) can be replaced by the block diagram shown in Figure 1.2(b). It is also known as an MPU (microprocessor unit).

1.11 Advances in Semiconductor Technology

In the last forty years, semiconductor technology has undergone unprecedented changes. After the invention of the transistor, integrated circuits (ICs) appeared on the scene at the end of the 1950s; an entire circuit consisting of several transistors, diodes, and resistors could be designed on a single chip. In the early 1960s, logic gates known as the 7400 series were commonly available as ICs, and the technology of integrating the circuits of a logic gate on a single chip became known as small-scale integration (SSI). As semiconductor technology advanced, more than 100 gates were fabricated on one chip; this was called medium-scale integration (MSI). A typical example of MSI is a decade counter (7490). Within a few years, it was possible to fabricate more than 1000 gates on a single chip; this came to be known as large-scale integration (LSI). Now we are in the era of

*In computer terminology, 1K is equal to 1024. In scientific terminology, 1k is equal to 1000.