THE 8051 MICROCONTROLLER

Second Edition

I. Scott MacKenzie

SECOND EDITION

THE 8051 MICROCONTROLLER

I. Scott MacKenzie
University of Guelph
Guelph, Ontario

Prentice Hall, Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in- Publication Data
MacKenzie, I. Scott

The 8051 microcontroller / I. Scott MacKenzie—2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-02-373660-7
1. Intel 8051 (Computer)—Programming 2. Digital control systems.
I. Title. II. Title: Eight thousand fifty-one
QA76.8.127M23 1995
004.165—-dc20 94-8278
CIP

Cover Photo: **Lester Lefkowitz/Tony Stone Worldwide**
Editor: **Charles E. Stewart, Jr.**
Production Editor: **Stephen C. Robb**
Cover Designer: **Julia Z. Van Hook**
Production Manager: **Pamela D. Bennett**

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

ISBN 0-02-373660-7

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

This book examines the hardware and software features of the MCS-51 family of microcontrollers. The intended audience is college or university students of electronics or computer technology, electrical or computer engineering, or practicing technicians or engineers interested in learning about microcontrollers

The means to effectively fulfill that audience's informational needs were tested and refined in the development of this book. In its prototype form, *The 8051 Microcontroller* has been the basis of a fifth semester course for college students in computer engineering. As detailed in Chapter 10, students build an 8051 single-board computer as part of this course. That computer, in turn, has been used as the target system for a final, sixth semester "project" course in which students design, implement, and document a "product" controlled by the 8051 microcontroller and incorporating original software and hardware.

Since the 8051—like all microcontrollers—contains a high degree of functionality, the book emphasizes architecture and programming rather than electrical details. The software topics are delivered in the context of Intel's assembler (ASM51) and linker/locator (RL51).

It is my view that courses on microprocessors or microcontrollers are inherently more difficult to deliver than courses in, for example, digital systems, because a linear sequence of topics is hard to devise. The very first program that is demonstrated to students brings with it significant assumptions, such as a knowledge of the CPU's programming model and addressing modes, the distinction between an address and the content of an address, and so on. For this reason, a course based on this book should not attempt to follow strictly the sequence presented. Chapter 1 is a good starting point, however. It serves as a general introduction to microcontrollers, with particular emphasis on the distinctions between microcontrollers and microprocessors.

Chapter 2 introduces the hardware architecture of the 8051 microcontroller, and its counterparts that form the MCS-51 family. Concise examples are presented using short sequences of instructions. Instructors should be prepared at this point to introduce, in parallel, topics from Chapters 3 and 7 and Appendices A and C to support the requisite software knowledge in these examples. Appendix A is particularly valuable, since it contains in a single figure the entire 8051 instruction set.

Chapter 3 introduces the instruction set, beginning with definitions of the 8051's addressing modes. The instruction set has convenient categories of instructions (data transfer, branch, etc.) which facilitate a step-wise presentation. Numerous brief examples demonstrate each addressing mode and each type of instruction.

Chapters 4, 5, and 6 progress through the 8051's on-chip features, beginning with the timers, advancing to the serial port (which requires a timer as a baud rate generator), and concluding with interrupts. The examples in these chapters are longer and more complex than those presented earlier. Instructors are wise not to rush into these chapters: it is essential that students gain solid understanding of the 8051's hardware architecture and instruction set before advancing to these topics.

Many of the topics in Chapter 7 will be covered, by necessity, in progressing through the first six chapters. Nevertheless, this chapter is perhaps the most important for developing in students the potential to undertake large-scale projects. Advanced topics such as assemble-time expression evaluation, modular programming, linking and locating, and macro programming will be a significant challenge for many students. At this point the importance of hands-on experience cannot be over-emphasized. Students should be encouraged to experiment by entering the examples in the chapter into the computer and observing the output and error messages provided by ASM51, RL51, and the object-to-hex conversion utility (OH).

Some advanced topics relating to programming methods, style, and the development environment are presented in Chapters 8 and 9. These chapters address larger, more conceptual topics important in professional development environments.

Chapter 10 presents several design examples incorporating selected hardware with supporting software. The software is fully annotated and is the real focus in these examples. The second edition includes two additional interfaces; a digital-to-analog output interface using an MC1408 8-bit DAC, and an analog-to-digital input interface using an ADC0804 8-bit ADC. One of the designs in Chapter 10 is the SBC-51—the 8051 single-board computer. The SBC-51 can form the basis of a course on the 8051 microcontroller. A short monitor program is included (see Appendix G) which is sufficient to get "up and running." A development environment also requires a host computer which doubles as a dumb terminal for controlling the SBC-51 after programs have been downloaded for execution.

Many dozens of students have wire-wrapped prototype versions of the SBC-51 during the years that I have taught 8051-based courses to computer engineering students. Shortly after the release of the first edition of this text, URDA, Inc. (Pittsburgh, Pennsylvania) began manufacturing and marketing a PC-board version of the SBC-51. This has proven to be a cost-effective solution to implementing a complete lecture-plus-lab package for teaching the 8051 microcontroller to technology students. Contact URDA at 1-800-338-0517 for more information.

Finally, each chapter contains questions further exploring the concepts presented. This new edition includes 128 end-of-chapter questions—almost double the number in the first edition. A solutions manual is available to instructors from the publisher.

The book makes extensive use of, and builds on, Intel's literature on the MCS-51 devices. In particular, Appendix C contains the definitions of all 8051 instructions and Appendix E contains the 8051 data sheet. Intel's cooperation is gratefully acknowledged. I also thank the following persons who reviewed the manuscript and offered invaluable comments, criticism, and suggestions: Antony Alumkal, Austin Community College; Omer Farook, Purdue University—Calumet; David Jones, Lenoir Community College; Roy Seigel, DeVry Institute; and Chandra Sekhar, Purdue University—Calumet.

I. Scott MacKenzie

CONTENTS

1 INTRODUCTION TO MICROCONTROLLERS

1.1 INTRODUCTION

Although computers have only been with us for a few decades, their impact has been profound, rivaling that of the telephone, automobile, or television. Their presence is felt by us all, whether computer programmers or recipients of monthly bills printed by a large computer system and delivered by mail. Our notion of computers usually categorizes them as "data processors," performing numeric operations with inexhaustible competence.

We confront computers of a vastly different breed in a more subtle context performing tasks in a quiet, efficient, and even humble manner, their presence often unnoticed. As a central component in many industrial and consumer products, we find computers at the supermarket inside cash registers and scales; at home in ovens, washing machines, alarm clocks, and thermostats; at play in toys, VCRs, stereo equipment, and musical instruments; at the office in typewriters and photocopiers; and in industrial equipment such as drill presses and phototypesetters. In these settings computers are performing "control" functions by interfacing with the "real world" to turn devices on and off and to monitor conditions. **Microcontrollers** (as opposed to microcomputers or microprocessors) are often found in applications such as these.

It's hard to imagine the present world of electronic tools and toys without the microprocessor. Yet this single-chip wonder has barely reached its twentieth birthday. In 1971 Intel Corporation introduced the 8080, the first successful microprocessor. Shortly thereafter, Motorola, RCA, and then MOS Technology and Zilog introduced similar devices: the 6800, 1801, 6502, and Z80, respectively. Alone these integrated circuits (ICs) were rather helpless (and they remain so); but as part of a single-board computer (SBC) they became the central component in useful products for learning about and designing with microprocessors. These SBCs, of which the *D2* by Motorola, *KIM-1* by MOS Technology, and *SDK-85* by Intel are the most memorable, quickly found their way into design labs at colleges, universities, and electronics companies.

A device similar to the microprocessor is the microcontroller. In 1976 Intel introduced the 8748, the first device in the MCS-48™ family of microcontrollers. Within a single integrated circuit containing over 17,000 transistors, the 8748 delivered a CPU, 1K byte of EPROM, 64 bytes of RAM, 27 I/O pins, and an 8-bit timer. This IC, and other MCS-48™ devices that followed, soon became an industry standard in control-oriented applications. Replacement of electromechanical components in products such as washing machines and traffic light controllers was a popular application initially, and remains so. Other products where microcontrollers can be found include automobiles, industrial equipment, consumer entertainment products, and computer peripherals. (Owners of an IBM *PC* need only look inside the keyboard for an example of a microcontroller in a minimum-component design.)

The power, size, and complexity of microcontrollers advanced an order of magnitude in 1980 with Intel's announcement of the 8051, the first device in the MCS-51™ family of microcontrollers. In comparison to the 8048, this device contains over 60,000 transistors, 4K bytes ROM, 128 bytes of RAM, 32 I/O lines, a serial port, and two 16-bit timers—a remarkable amount of circuity for a single IC (see Figure 1–1). New members have been added to the MCS-51™ family, and today variations exist virtually doubling these specifications. Siemens Corporation, a second source for MCS-51™ components, offers the SAB80515, an enhanced 8051 in a 68-pin package with six 8-bit I/O ports, 13 interrupt sources, and an 8-bit A/D converter with 8 input channels. The 8051 family is well established as one of the most versatile and powerful of the 8-bit microcontrollers, its position as a leading microcontroller entrenched for years to come.

This book is about the MCS-51™ family of microcontrollers. The following chapters introduce the hardware and software architecture of the MCS-51™ family, and demonstrate through numerous design examples how this family of devices can participate in electronic designs with a minimum of additional components.

In the following sections, through a brief introduction to computer architecture, we shall develop a working vocabulary of the many acronyms and buzz words that prevail

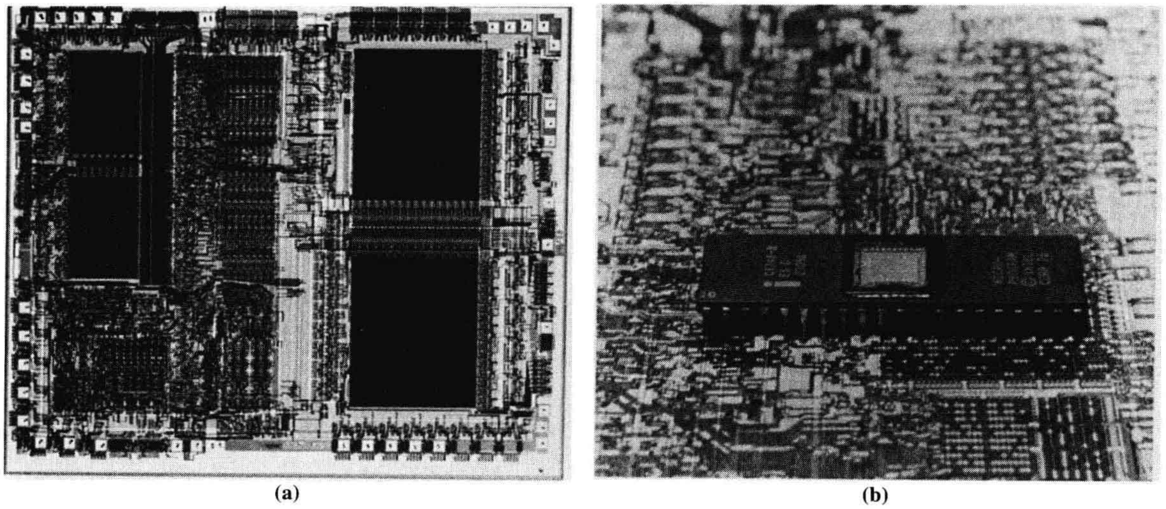
(a) (b)

FIGURE 1–1
The 8051 microcontroller. (a) An 8051 die. (b) An 8751 EPROM. (Courtesy Intel Corp. Copyright 1991.)

(and often confound) in this field. Since many terms have vague and overlapping definitions subject to the prejudices of large corporations and the whims of various authors, our treatment is practical rather than academic. Each term is presented in its most common setting with a straightforward explanation.

1.2 TERMINOLOGY

To begin, a **computer** is defined by two key traits: (1) the ability to be programmed to operate on data without human intervention, and (2) the ability to store and retrieve data. More generally, a **computer system** also includes the **peripheral devices** for communicating with humans, as well as **programs** that process data. The equipment is **hardware,** the programs are **software.** Let's begin with computer hardware by examining Figure 1–2.

The absence of detail in the figure is deliberate, making it representative of all sizes of computers. As shown, a computer system contains a **central processing unit** (CPU) connected to **random access memory** (RAM) and **read-only memory** (ROM) via the **address bus, data bus,** and **control bus. Interface circuits** connect the system buses to **peripheral devices.** Let's discuss each of these in detail.

1.3 THE CENTRAL PROCESSING UNIT

The CPU, as the "brain" of the computer system, administers all activity in the system and performs all operations on data. Most of the CPU's mystique is undeserved, since it is just a collection of logic circuits that continuously performs two operations: fetching

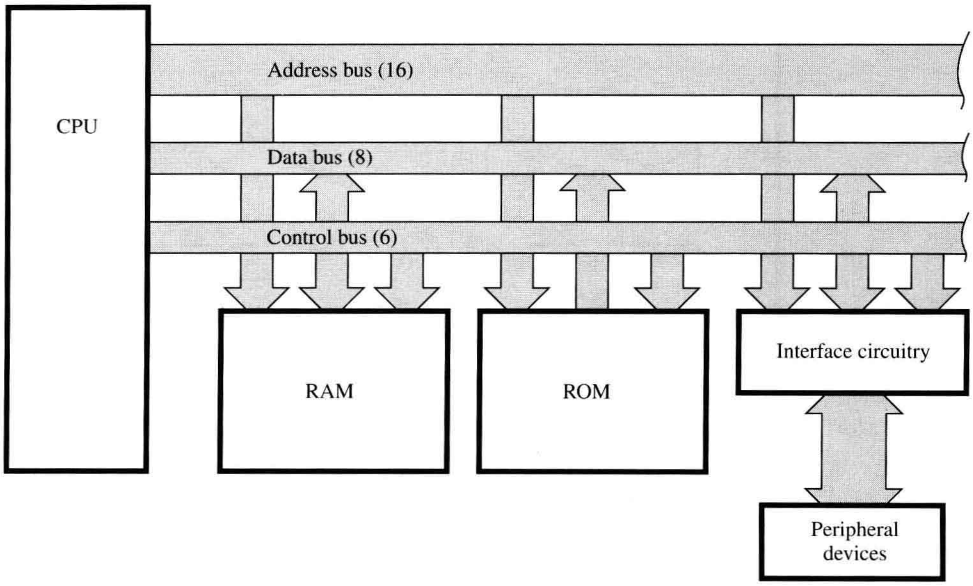

FIGURE 1–2
Block diagram of a microcomputer system

instructions and executing instructions. The CPU has the ability to understand and execute instructions based on a set of binary codes, each representing a simple operation. These instructions are usually arithmetic (add, subtract, multiply, divide), logic (AND, OR, NOT, etc.), data movement, or branch operations, and are represented by a set of binary codes called the **instruction set.**

Figure 1–3 is an extremely simplified view of the inside of a CPU. It shows a set of **registers** for the temporary storage of information, an **arithmetic and logic unit** (ALU) for performing operations on this information, an **instruction decode and control unit** that determines the operation to perform and sets in motion the necessary actions to perform it, and two additional registers. The **instruction register** (IR) holds the binary code for each instruction as it is executed, and the **program counter** (PC) holds the memory address of the next instruction to be executed.

Fetching an instruction from the system RAM is one of the most fundamental operations performed by the CPU. It involves the following steps: (a) the contents of the program counter are placed on the address bus, (b) a READ control signal is activated, (c) data (the instruction opcode) are read from RAM and placed on the data bus, (d) the opcode is latched into the CPU's internal instruction register, and (e) the program counter is incremented to prepare for the next fetch from memory. Figure 1–4 illustrates the flow of information for an instruction fetch.

The execution stage involves decoding (or deciphering) the opcode and generating control signals to gate internal registers in and out of the ALU and to signal the ALU to perform the specified operation. Due to the wide variety of possible operations, this explanation is somewhat limited in scope. It applies to a simple operation such as "incre-

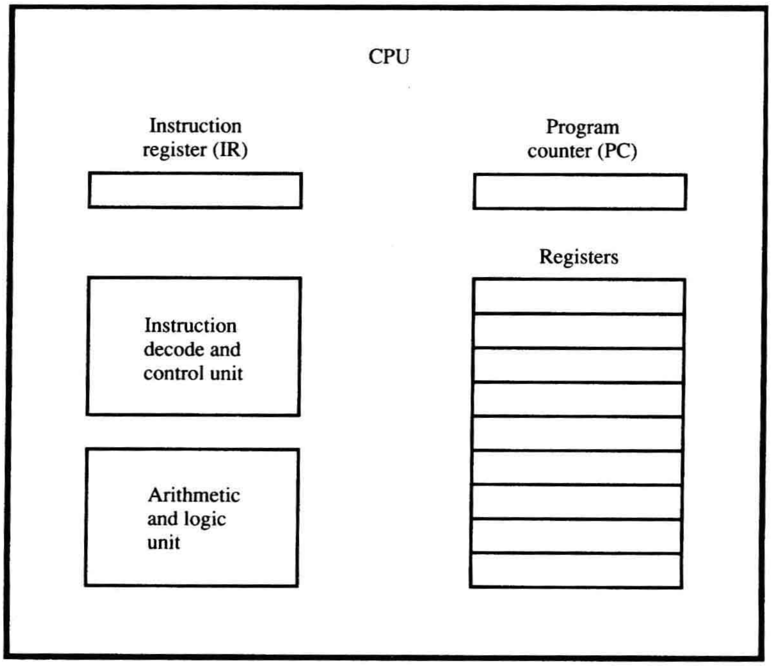
CPU
Instruction register (IR)
Program counter (PC)
Registers
Instruction decode and control unit
Arithmetic and logic unit

FIGURE 1–3
The central processing unit (CPU)

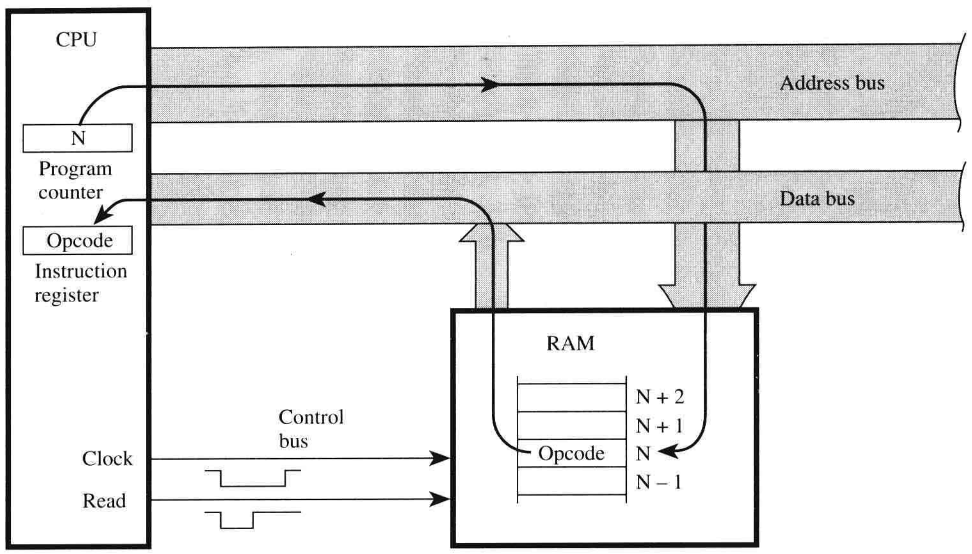

FIGURE 1–4
Bus activity for an opcode fetch cycle

ment register." More complex instructions require more steps, such as reading a second and third byte as data for the operation.

A series of instructions combined to perform a meaningful task is called a **program,** or **software,** and herein is the real mystique. The degree to which tasks are efficiently and correctly carried out is determined for the most part by the quality of software, not by the sophistication of the CPU. Programs, then, "drive" the CPU, and in doing so they occasionally go amiss, mimicking the frailties of their authors. Phrases such as "The computer made a mistake" are misguided. Although equipment breakdowns are inevitable, mistakes in results are usually a sign of poor programs or operator error.

1.4 SEMICONDUCTOR MEMORY: RAM AND ROM

Programs and data are stored in memory. The variations of computer memory are so vast, their accompanying terms so plentiful, and technology breakthroughs so frequent, that extensive and continual study is required to keep abreast of the latest developments. The memory devices directly accessible by the CPU consist of semiconductor ICs (integrated circuits) called RAM and ROM. There are two features that distinguish RAM and ROM: first, RAM is read/write memory while ROM is read-only memory; and second, RAM is volatile (the contents are lost when power is removed), while ROM is nonvolatile.

Most computer systems have a disk drive and a small amount of ROM, just enough to hold the short, frequently used software routines that perform input/output operations. User programs and data are stored on disk and are loaded into RAM for execution. With