8080 8085 Software Design

Book 1

Christopher A. Titus, Peter R. Rony,
David G. Larsen, and Jonathan A. Titus

BLACKSBURG CONTINUING EDUCATION SERIES™
Edited by Titus, Titus & Larson

The Blacksburg Continuing Education™ Series

The Blacksburg Continuing Education Series™ of books provide a Laboratory—or experiment-oriented approach to electronic topics. Present and forthcoming titles in this series include:

- DBUG: An 8080 Interpretive Debugger
- Design of Active Filters, With Experiments
- Design of Op-Amp Circuits, With Experiments
- Design of Phase-Locked Loop Circuits, With Experiments
- Design of Transistor Circuits, With Experiments
- Design of VMOS Circuits, With Experiments
- The 8080A Bugbook®: Microcomputer Interfacing and Programming
- 8080/8085 Software Design (2 Volumes)
- 555 Timer Applications Sourcebook, With Experiments
- Guide to CMOS Design Basics: Circuits and Experiments
- Interfacing and Scientific Data Communications Experiments
- Introductory Experiments in Digital Electronics and 8080A Microcomputer Programming and Interfacing (2 Volumes)
- Logic & Memory Experiments Using TTL Integrated Circuits (2 Volumes)
- Microcomputer—Analog Converter Software and Hardware Interfacing
- Microcomputer Interfacing With the 8255 PPI Chip
- NCR Basic Electronics Course, With Experiments
- NCR Data Communications Concepts
- NCR Data Processing Concepts Course
- NCR EDP Concepts Course
- Programming and Interfacing the 6502
- 6800 Microcomputer Interfacing and Programming, With Experiments
- 6502 Software Design
- TEA: An 8080/8085 Co-Resident Editor/Assembler
- TRS-80 Interfacing
- Z-80 Microprocessor Programming & Interfacing (2 Volumes)

In most cases, these books provide both text material and experiments, which permit one to demonstrate and explore the concepts that are covered in the book. These books remain among the very few that provide step-by-step instructions concerning how to learn basic electronic concepts, wire actual circuits, test microcomputer interfaces, and program computers based on popular microprocessor chips. We have found that the books are very useful to the electronic novice who desires to join the "electronics revolution," with minimum time and effort.

Additional information about the "Blacksburg Group" is presented inside the rear cover.

Jonathan A. Titus, Christopher A. Titus, and David G. Larsen
"The Blacksburg Group"

8080/8085 Software Design

Book 1

by

Christopher A. Titus, Peter R. Rony,
David G. Larsen, and Jonathan A. Titus

Also published as

8080/8085 Software Design
by E & L Instruments, Inc.

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

FIRST EDITION
THIRD PRINTING–1980

International Standard Book Number: 0-672-21541-1
Library of Congress Catalog Card Number: 78-57207

Printed in the United States of America.

Preface

Although it did not look like a revolution at the time, the Intel Corporation introduced the world to the concept of microprocessors and microcomputers with its four-bit 4004 microprocessor "chip." The microcomputer revolution gained momentum as improvements were made in earlier integrated circuits, and more and more manufacturers began to produce microprocessor integrated circuits and microcomputers. Now these devices, made from one of the chemical components of purified *sand*, have begun to truly change the way in which we live.

In the supermarket, the electronic cash register not only performs all of the tasks of its mechanical predecessor, but it can also perform inventory control. Some sewing machines no longer require complex gears and cams to do special stitches. Instead, a microcomputer controls the position of the needle when a particular stitch is used. In the near future, you will probably be able to even "teach" your sewing machine the latest stitch.

Instead of using a slide rule to perform complex calculations, calculators can be used that have more "power" than state-of-the-art computers produced 10 or 20 years ago. Watches no longer have to be wound, and, in fact, the terms *clockwise* and *counterclockwise* may soon be archaic. Today, you can even use a checkbook that electronically keeps track of how much money you have in your checking account. No longer do you have to balance your checkbook every month. In the not-too-distant future, microwave ovens may "know" more about cooking than you do.

Of course, to use a microcomputer to solve a problem requires both *hardware* and *software*. The hardware consists of the microprocessor integrated circuit(s) and possibly some memory inte-

grated circuits, light-emitting diodes, and *interface* electronics so that the microcomputer can control *peripheral devices* (lights, motors, valves, solenoids, or gauges). The software consists of the sequence of instructions that the microcomputer executes so that it can process data or control peripheral devices. Unfortunately, the design and implementation of software is not as well defined as the design and implementation of hardware.

Some current microcomputer users program their microcomputers in the BASIC* programming language. To do this, a BASIC interpreter, an assembly language program, must be stored in from 5000 to 8000 memory locations. This assembly language program enables you to enter a BASIC program into the microcomputer and actually execute it. Like everything else in the world, there are advantages and disadvantages to programming a microcomputer in the BASIC language. Due to the disadvantages of using the BASIC language, *assembly language* is often used when programming microcomputers. This is the type of programming language that we will discuss in this book.

There is no doubt in our minds that assembly language is the more difficult of the two to teach and to learn. You cannot write an assembly language program in 10 or 15 minutes that will calculate the cube roots of all numbers between 1 and 1000 to six significant digits. Using the BASIC language, you probably could. However, there are advantages in using assembly language. In fact, many tasks can be performed using assembly language that cannot be performed using the BASIC language, particularly in the areas of process control, peripheral control, high-speed calculations, and real-time data acquisition. In fact, most consumer products that incorporate microcomputers are programmed in assembly language; video (television) games, microwave ovens, sewing machines, electronic cash registers (point-of-sale terminals), gasoline pumps, and blood-pressure monitors. Therefore, assembly language is a very powerful and useful language to learn.

In the first chapter of this book, we do not discuss any assembly language programming, but rather the characteristics of the 8080 and 8085 microprocessor integrated circuits. This includes a discussion of the various *registers* contained within these two integrated circuits. These registers are important because you will use them time and time again in your assembly language programs. The next three chapters deal with the assembly language instructions that the 8080 and 8085 can actually execute. You will not find long, detailed programs in these first few chapters, since you will not have enough familiarity with all of the important instructions to understand long, complex programs. When these instructions are understood, you can begin to program the microcomputer in assembly language so that it performs useful tasks. The remaining chapters of the book deal with the application of the 8080's assembly language instructions to mathematical operations, number-base conversions, and input/output device (peripheral device) control.

There are some unusual features of this book which we believe that you will appreciate. We have not included a chapter on binary, octal, hexadecimal, and decimal numbering systems. In some of the chapters, we do show you how some of these operations are performed using paper and pencil. However, we are personally more interested in the instruction sequence that the microcomputer must execute to perform these same operations. If you need more information on basic mathematical operations, you can find it in just about any other book on computer programming. You will also find that the programs in this book can be executed on just about any 8080-based microcomputer. We do not have hardware or software examples that feature one or another manufacturer's hardware. The program examples will execute equally well on a MITS, Processor Technology, Intel Corporation, National Semiconductor Corporation, Digital Group, Control Logic, IMS Associates, or E & L Instruments 8080-based microcomputer. Of course, peripheral devices may vary from system to system just as device addresses may vary from system to system.

In the Input/Output chapter (Chapter 7), we have included the electrical schematic diagrams for the devices that we are controlling with software. A software listing of a program that controls a peripheral device is *worthless* unless you can see how the peripheral device is actually wired (interfaced) to the microcomputer. Also, if you want to use a particular program on your own microcomputer, you will probably need the schematic diagram of the peripheral device in order to duplicate the hardware.

In the first three chapters of the book, we have included both octal and hexadecimal op codes for the instructions being discussed. We prefer the octal numbering system. We feel strongly that the op codes for the 8080's assembly language instructions are easier to remember in octal, and when you examine an octal op code, it is easy to determine the registers or memory locations that are affected by the instruction. If the hexadecimal numbering system is used, this process is very difficult. On the other hand, many programmers prefer the hexadecimal numbering system because they only have to remember a two-digit op code rather than a three-digit octal op code. The only conclusion that we could reach was to use both numbering systems in our examples. We recommend that a beginning 8080 or 8085 programmer use octal, simply because

* BASIC is a registered trademark of the trustees of Dartmouth College.

it is easier to remember. At a later time, it may be more convenient to use hexadecimal. If you are unsure of which numbering system to use, try both and stay with the numbering system that you are the most comfortable with. Hexadecimal numbers are usually found in the text bracketed with parentheses; e.g., (5F).

Another goal that we had was to provide detailed descriptive explanations of how the program examples operate. We do not say, "Here it is, you figure out how it works." You would learn very little if this approach were used. Instead, we *develop* programs, starting with the simplest sequence of instructions that can accomplish a particular task. These programs often have limitations, and if they do, we add instructions so that the microcomputer can better perform either a more general-purpose task or a specific task. Therefore, we *design* the best solution to a problem, learning from previous examples.

As you go through this book, you may notice that the computer printouts in the program examples are in a format that is different than the formats used by other authors. The reason that we have chosen the format that we have is so that newcomers to assembly language programming can grasp the *concepts* of assembly language programming. One of the most important concepts is the fact that multibyte instructions *must* be stored in consecutive memory locations. This concept is difficult to grasp if you look at the computer printouts used as examples in other books. For this reason, we have written a coresident editor/assembler (TEA) that produces the listings that we felt would be most appropriate to a book on assembly language programming. It is important to emphasize that once you learn the principles of good assembly language programming, it really does not matter what editor/assembler software package you use for assembly language program development. Remember, editors and assemblers are only the means to the end. They are just tools that are used to aid you in the development of a properly operating assembly language program.

One of the nice features of software is that it is easy to print and distribute. Because of this, a number of professional and hobbyist computer magazines provide listings and source and object paper tapes or audio cassettes for the programs in their software libraries. Software listings often appear in *Electronic Design, Computer Design,* and *Electronics*. The hobbyist magazines, *Interface Age* (McPheters, Wolfe & Jones, Cerritos, CA 90701), *SCCS Interface* (SCCS, Santa Monica, CA 90405), *Kilobaud* and *Byte* (both of Peterborough, NH 03458), and *Dr. Dobbs Journal of Computer Calisthenics and Orthodontia* (Menlo Park, CA 94025) also have numerous program listings. There are also a number of software libraries, including the Intel User's Library, 3065 Bowers Ave., Santa Clara, CA 95051, and the Microcomputer Software Depository, 2361 E. Foothill Blvd., Pasadena, CA 91107. Entries in the Depository are listed in *Interface Age* every month.

Tychon, Inc., is dedicated to educating the scientist, engineer, and electronics/computer hobbyist in the areas of digital electronics, microcomputer hardware, and microcomputer software. We currently have monthly columns that appear in *American Laboratory* (International Scientific Communications, Inc., 808 Kings Highway, Fairfield, CT 06430), the Thai publication, *Semiconductor Electronics Journal* (Science, Engineering and Education Co., Ltd., Bangkok, Thailand), and the German publication, *Elektroniker* (Burgdorf, Switzerland). Our columns also appear in *Electronic News* (IPC Business Press Pty. Ltd., Sydney, Australia), *Ham Radio* (Greenville, NH), *Computer Design* (Concord, MA), and *Radio-Electronics* (New York City, NY). Our books are currently being translated into Italian, German, French, Spanish, and Thai.

If you are interested in learning basic digital electronics, a two-volume set of books in the *Blacksburg Continuing Education Series*™, entitled *Logic & Memory Experiments Using TTL Integrated Circuits,* will introduce you to the subject and give you the ability to try student-proven experiments. Another book in the series, *Interfacing and Scientific Data Communications Experiments,* has been specifically written for those engineers, scientists, and hobbyists interested in asynchronous communications techniques using UARTs. This book includes experiments that will permit you to use asynchronous communications techniques to interface instrumentation to computers or other UART-based devices. Three books in the series have been written about the 8080 microprocessor/microcomputer: *The 8080A Bugbook®: Microcomputer Interfacing and Programming,* and a two-volume set, *Introductory Experiments in Digital Electronics and 8080A Microcomputer Programming and Interfacing.* These self-teaching texts describe the design of elementary interface hardware and software and how they are applied to probblems, the substitution of software for hardware, and many other interesting topics. One of our latest books, *Microcomputer–Analog Converter Software and Hardware Interfacing,* deals entirely with interfacing digital-to-analog and analog-to-digital converters to microcomputers. The number of titles in the *Blacksburg Continuing Education Series*™ grows constantly. They are listed on the inside front cover.

Many of the programming concepts that are presented in this book have been incorporated into the material taught at seminars

Bugbook is a registered trademark of E & L Instruments, Inc., Derby, CT 06418.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.