ASSEMBLY LANGUAGE
FOR THE IBM-PC

2nd edition

KIP R. IRVINE

Assembly Language for the IBM-PC

Second Edition

KIP R. IRVINE

Miami-Dade Community College-Kendall

PRENTICE HALL
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging in Publication Data

Irvine, Kip R.,
Assembly language for the IBM-PC/Kip R. Irvine.—2nd ed.
p. cm.
ISBN 0-02-359651-1
1. IBM Personal Computer—Programming. 2. Assembler language (Computer program language) I. Title.
QA76.8.I259I77 1993
005.265—dc20 92-10002
CIP

Editor(s): John Griffin
Production Supervisor: Ron Harris
Production Manager: Paul Smolenski
Text Designer: Jane Edelstein
Cover Designer: Tom Mack
Cover Art: Tim Alt

This book was set in Times Roman by York Graphics Services.

A Simon & Schuster Company
Englewood Cliffs, New Jersey 07632

Printed in the United States of America

10

ISBN 0-02-359651-1

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*

Assembly Language for the IBM-PC

To Jack and Candy Irvine

Preface

Assembly Language for the IBM-PC directly addresses the needs of a college course in assembly language programming. Of course, there are many assembly languages, so the goal of this book is to give students an understanding of the following:

- The Intel 8086/8088 assembly language instruction set, directives, macros, and data allocation statements.
- How programs interact with the operating system, including memory management, input-output services.
- Programming methodology, using assembly language as a tool.
- Direct programming of computer hardware.

We begin with machine and operating systems concepts. Then the assembler instruction set is gradually introduced, along with short program examples. The latter third of the book contains application programs that take advantage of advanced operating system functions. We show how to link assembly language subroutines to Pascal and C. A structured programming style is used throughout the book, to make the programs both effective and easy to read.

The book is designed to accompany a sophomore-, junior-, or senior-level college course in assembly language programming. Twelve chapters can usually be covered in a single semester or in two quarters. I consider the first nine chapters to be the core text, with the remaining six being individually selectable topics. This book may also be used for self-study, enhanced by the sample program disk available from the publisher.

As a classroom teacher of assembly language, my goal is to help students approach problems with the mindset of an assembly language programmer. Ideally, students learning assembly language acquire the confidence to tackle programming problems at the systems level, free from the restrictions imposed by high-level languages. In this book, I integrate presentations of new instructions with short examples showing how they may be *applied* to programming problems.

In addition to the hundreds of short examples, *Assembly Language for the IBM-PC* contains over 75 ready-to-run programs, each newly written for this book, that demonstrate instructions or ideas as they are presented in the text. Reference materials such as guides to DOS interrupts and instruction mnemonics are available at the end of the book. There is a sizable link library (introduced in Chapter 9), which is expanded throughout the text and available on disk. A short but useful macro library is also included.

Required Background. The reader should know how to program in at least one structured computer language, preferably Pascal, Ada, Modula-2, or C. I have personally used the book with majors in Computer Science, Management Information Systems, Computer Information Systems, and Electrical Engineering as well as professional programmers. All programs have been tested using Borland's Turbo Assembler 2.0 and Microsoft's Macro Assembler 5.1.

Operating System Concepts. We have a great advantage when writing assembly language programs on a microcomputer, because they communicate directly with DOS. System integrity or security are of no concern when only one user is involved. I constantly try to remove the mysteries shrouding high-level languages and DOS. Students acquire more confidence in their programming skills, and the principles learned here carry over to more advanced courses in operating systems and computer organization.

Enhancements to the Second Edition

First and foremost, the second edition was needed to bring the discussions about hardware and software up to date with the ever-changing computer industry. The following improvements were made:

- A stronger introduction to programming using DEBUG in the first two chapters.
- Quick-reference guides to Microsoft CodeView and Borland Turbo Debugger.
- Earlier introduction of loops and conditional processing.
- Less space devoted to numeric conversions and file processing.
- A more complete link library (CONSOLE.LIB).
- A detailed explanation of instruction encoding, relating to op codes, the addressing ModR/M byte, and operands.
- Inline assembly code in Pascal; better coverage of linking assembly routines to Pascal and C.
- A more thorough explanation of memory segmentation and EXE program structure.
- A better explanation of the BIOS keyboard services.
- An include file containing many useful macros.
- 80286 and 80386 instructions, with a sample program.
- Hardware concepts: Differences between the 8088, 8086, 80286, 80386, and 80486 processors. Discussion of *real mode, protected mode,* and *virtual mode* as applied to the 80286 and 80386 processors.

ORGANIZATION AND FORMAT

Presentation Sequence

Chapters 1–8 represent the basic foundation of assembly language, and are most effective if covered in order. A great deal of effort went into making these chapters flow smoothly:

1: **Introduction**
Basic concepts, machine language, numbering systems, elements of an assembly language program.

2: **Hardware and Software Architecture**
Hardware fundamentals and terminology, registers, system software, stack.

3: **Assembly Language Fundamentals**
Data definition, program structure, MOV, XCHG, INC, DEC, ADD, SUB, addressing modes.

4: **The Macro Assembler**
Assembling, linking, operators, expressions

5: **Input-Output Services**
Interrupts, DOS (INT 21h) services, video and keyboard BIOS services.

6: **Conditional Processing**
Boolean and comparison instructions, conditional jumps and loops, high-level logic structures.

7: **Arithmetic**
Shift and rotate instructions, multidigit arithmetic, multiplication and division, ASCII and packed decimal arithmetic.

8: **Numeric Conversions and Libraries**
Character translation (XLAT), binary-to-decimal conversion, creating external subroutines.

Chapters 9 through 15 may be covered in any order, with one minor restriction: The link library developed in Chapter 9 (CONSOLE.LIB) is used by nearly all programs in Chapters 10, 11, 12, and 13. The instructor may wish to supply students with the library on disk even at the beginning of the course.

9: **String Processing**
String storage methods, string primitive instructions, building a library of string routines.

10: **Macros and Structures**
Declaring and calling macros, conditional assembly, using macros to call procedures, STRUC and RECORD directives, advanced operators and directives.

11: **Disk Storage**
Disk storage fundamentals, directory, file allocation table, system-level file and directory access.

12: **File Processing**
Standard DOS file functions, text file applications, fixed-length record processing, applications, random record retrieval and indexing.

13: **High-Level Linking**
Linking to Turbo Pascal, built-in assembler (BASM), inline assembly code, linking to Turbo C.

14: **Advanced Topics—I**
Pointers, indirect jumps and calls, interrupt and processor control, defining segments, COM programs, EXE programs, memory models.

15: **Advanced Topics—II**
Real-time clock, instruction timings, bit encoding of instructions, dynamic memory allocation, interrupt handling, memory-resident programs, 80x87 math co-processor.

FEATURES

Borland TASM and Microsoft MASM. This book takes advantage of the simplified segment directives (.CODE, .DATA, and .STACK) available in both TASM and MASM. These directives greatly simplify program development for beginners.

Complete Program Listings. The book contains over 75 assembled and tested programs. They are on disk, either attached to the book or available from Macmillan. All source code from the link library (CONSOLE.LIB) and all source code for a macro library are included on the disk.

Programming Logic. Chapters 6 and 7 emphasize Boolean arithmetic for comparison and bit manipulation. This includes the AND, OR, XOR, NOT, TEST, shift, and rotate instructions. Chapters 6 and 12 both show how to create and optimize high-level logic structures using assembly language, including WHILE, REPEAT, FOR-NEXT, and IF-ELSE.

Hardware and Operating System Concepts. Chapter 2 introduces basic hardware and DOS concepts, including registers, flags, stack, memory addressing, and memory mapping. Chapter 3 discusses EXE program structure. Chapter 5 shows how assembly language interacts with DOS and the BIOS. Chapter 15 demonstrates dynamic memory allocation, interrupt handling, and memory-resident programs. Chapter 15 provides information about instruction timings, machine cycles, and bit-encoding of machine instructions.

Chapter Ending Materials. Each chapter contains valuable teaching materials to reinforce student learning (such materials are typically missing from trade books on this subject). The review questions ask both general and specific questions relating to chapter material. The programming exercises are based on information and skills presented during the chapter, set at varying levels of difficulty. Selected answers to review questions are available in an appendix.

Special Programming Tips. Nearly every chapter has a box containing a special topic called a programming tip. This contains more advanced or specialized information related to the current chapter material.

Two Chapters on DISK Storage and Files. Chapter 11 covers the details of disk storage and shows how to manipulate disk drives, directories, file attributes, and the file allocation table directly. This provides a valuable tool to systems programmers and application programmers alike, who must go beyond the standard file access methods available in high-level languages.

Chapter 12 concentrates on file applications, covering extended DOS functions, text files, fixed-length records, random access, and indexed record retrieval.

Creating Object Libraries. I emphasize a toolbox approach to programming, as early as Chapter 5. Linking separately compiled modules is introduced in Chapter 8, and Chapter 9 shows how to use the Microsoft LIB (or Borland TLIB) utility to build a link library. Chapters 11–12 use and extend the library.

I'm hoping that students will recognize that a toolbox of assembly routines can be a valuable resource when writing application programs. It frees one from having to write the same low-level code over and over again, and it encourages a structured approach to programming.

Complete Chapter on Macros. Macros are an important topic in any assembly language course. They give the student a chance to learn about procedure parameters and to see how high-level languages build on standard routines. Chapter 13 is devoted to macros and advanced assembler directives. This chapter might easily be covered immediately after Chapter 8. Special emphasis is given to showing how simple macros can streamline procedure calls.

Linking to High-Level Languages. A continuing topic of interest is the linking of assembly routines to high-level languages. In fact, this is the area where assembler is used most often. Chapter 15 discusses the most common ways of passing arguments to subroutines, coordinating identifiers and segment declarations, and linking to Pascal and C.

Instructional Aids. All program listings and libraries are available on disk from the publisher. A comprehensive instructor's manual is also available, containing topic outlines, solutions to programming exercises, lecture strategies, and transparency masters taken from selected figures in the text.

REFERENCE MATERIALS

One of the most important differences between a commercial trade book and a textbook lies in its special reference materials. I find that students have a difficult time obtaining the original manuals in most computer labs, so they depend on the following appendixes:

Binary and Hexadecimal Number Tutorial. Appendix A explains binary and hexadecimal numbers from the ground up. Special emphasis is placed on binary/decimal, binary/hexadecimal, and decimal/hexadecimal conversions.

CodeView, Turbo Debugger, and DEBUG Tutorials. Appendixes B, C, and D contain quick-reference guides to DEBUG, CodeView, and Turbo Debugger. There is also a hands-on tutorial for DEBUG. The DEBUG appendix should be read before doing the programming exercises in Chapters 2 and 3.

Guide to Companion Diskette. Appendix E lists all programs and files on the companion diskette. It contains concise documentation for all procedures in the link library and macro library.

Assembler References. Appendix F lists all reserved words for the Microsoft and Borland assemblers, and Appendix G contains a quick reference guide to assembler directives and operators.

DOS and BIOS Functions. Appendix G contains a quick reference to DOS and BIOS interrupts. One may quickly look up an interrupt, note its standard calling sequence, and use it in a program. Detailed information on individual interrupts is also available in Chapters 5, 11, and 12.

Complete Instruction Set Reference. Appendix H contains a listing of the Intel 8086/8088 instruction set. For each instruction, you can see which flags are affected, how the instruction works, and the standard syntax formats.

Answers to Selected Review Questions. Appendix I contains the answers to selected review questions from each chapter.

ASCII Codes and Keyboard Scan Codes. The inside back cover contains a listing of all ASCII codes. There are also tables containing keyboard scan codes for special keyboard keys. The front inside cover contains a chart of IBM-PC graphics characters.

ACKNOWLEDGMENTS

I want to express my warm thanks to the many people at Macmillan who contributed to the book's preparation, particularly John Griffin, Acquisition Editor, and Ron Harris, Production Editor. Special thanks are due the following groups and individuals who contributed to the book:

Barry Brosch, Bruce DeSautel, and Richard White of the Computer Information Systems department at Miami-Dade Community College (South Campus) reviewed individual chapters of the first edition and offered valuable suggestions.

Richard A. Beebe of Simpson College (Indianola Iowa) field tested the first edition in his class during summer 1988.

Bob Galivan wrote a marvelous instructor's manual, which really helps when planning lectures. Bob Galivan and George Kamenz proofread the manuscript, catching many mistakes.

Members of the Borland and Microsoft Compuserve forums donated information on MASM, TASM, and DOS.

Microsoft Corporation and Borland International generously donated software.

I would also like to thank the select group of reviewers of the First Edition who offered their suggestions: Richard A. Beebe, Simpson College; John V. Erhart, Northeast Missouri State University; and Michael Walton, Miami-Dade Community College-North.

Reviewers of the Second Edition include Gonshin Liu, University of Bridgeport, S. K. Sachdev, Eastern Michigan University; Douglas W. Knight, University of Southern Colorado; and Don Retzlaff, University of North Texas.

K.R.I.

Assembly Language for the IBM-PC

Contents