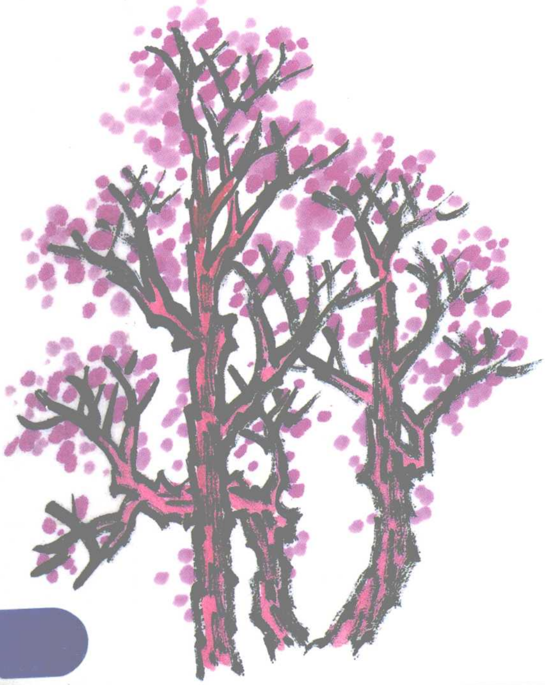
PEARSON Prentice Hall

大学计算机教育国外著名教材系列 影印版

Assembly Language for Intel-Based Computers Fiveth Edition

Intel汇编语言程序设计（第5版）



Kip R. Irvine 著

PEARSON Education

清华大学出版社

图书在版编目(CIP)数据

Intel 汇编语言程序设计 = Assembly Language for Intel-Based Computers, Fiveth Edition：第 5 版：英文 / （美）埃尔温（Irvine, K. R.）著. —影印本. —北京：清华大学出版社，2009.5

（大学计算机教育国外著名教材系列（影印版））

ISBN 978-7-302-19793-5

Ⅰ. I…　Ⅱ. 埃…　Ⅲ. 汇编语言－程序设计－高等学校－教材－英文　Ⅳ. TP313

中国版本图书馆 CIP 数据核字（2009）第 045374 号

责任印制：孟凡玉

出版发行：清华大学出版社　　地　址：北京清华大学学研大厦 A 座

http://www.tup.com.cn　　邮　编：100084

社 总 机：010-62770175　　邮　购：010-62786544

投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者：北京密云胶印厂

装 订 者：三河市新茂装订有限公司

发 行 者：全国新华书店

开　本：185×230　　印张：46

版　次：2009 年 5 月第 1 版　　印　次：2009 年 5 月第 1 次印刷

印　数：1~3000

定　价：69.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：010-62770177 转 3103　　产品编号：023984-01

出 版 说 明

进入 21 世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从 1996 年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书（影印版）”等一系列引进图书，受到国内读者的欢迎和支持。跨入 21 世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高校本科及研究生计算机教育的国外经典教材或著名教材，组成本套“大学计算机教育国外著名教材系列（影印版）”，以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材系列（影印版）”做得更好，更适合高校师生的需要。

清华大学出版社

Preface

Assembly Language for Intel-Based Computers, Fifth Edition, teaches assembly language programming and architecture for Intel IA-32 processors. It is an appropriate text for the following types of college courses:

- Assembly Language Programming
- Fundamentals of Computer Systems
- Fundamentals of Computer Architecture

Students use Intel or AMD processors and program with **Microsoft Macro Assembler (MASM) 8.0,** running on any of the following MS-Windows platforms: Windows 95, 98, Millenium, NT, 2000, and XP.

Although this book was originally designed as a programming textbook for college students, it has evolved over the last 15 years into much more. Many universities use the book for their introductory computer architecture courses. As a testament to its popularity, the fourth edition was translated into Korean, Chinese, French, Russian, and Polish.

Emphasis of Topics This edition includes topics that lead naturally into subsequent courses in computer architecture, operating systems, and compiler writing:

- Virtual machine concept
- Elementary boolean operations
- Instruction execution cycle
- Memory access and handshaking
- Interrupts and polling
- Pipelining and superscalar concepts
- Hardware-based I/O
- Floating-point binary representation

Other topics relate specifically to Intel IA-32 architecture:

- IA-32 protected memory and paging
- Memory segmentation in real-address mode
- 16-bit interrupt handling
- MS-DOS and BIOS system calls (interrupts)
- IA-32 Floating-Point Unit architecture and programming
- IA-32 Instruction encoding

Certain examples presented in the book lend themselves to courses that occur later in a computer science curriculum:

- Searching and sorting algorithms
- High-level language structures
- Finite-state machines
- Code optimization examples

Improvements in the Fifth Edition A number of improvements and new information have been added in this edition, listed in the following table by chapter number:

Chapter	Improvements
2	Improved explanation of the instruction execution cycle.
5	An expanded link library with additional subroutines to write rich user interfaces, calculate program timings, generate pseudorandom integers, and parse integer strings. The documentation of the library has greatly improved.
6	Improved explanation of conditional jump encoding and relative jump ranges.
7	Two-operand and three-operand IMUL instructions are added. Performance comparisons are shown for differing approaches to integer multiplication.
8	Completely redesigned so that low-level details of stack frames (activation records) are explained first before introducing MASM's high-level INVOKE and PROC directives.
10	Improved documentation of the book's macro library.
11	New topic: Dynamic memory allocation in MS-Windows applications. Improved coverage of file handling and error reporting in MS-Windows applications.
12	Improved coverage of calling C and C++ functions from assembly language.
17	Introduction to the IA-32 floating-point instruction set. Floating-point data types. IA-32 Instruction encoding and decoding.

Still a Programming Book This book is still focused on its original mission: to teach students how to write and debug programs at the machine level. It will never replace a complete book on computer architecture, but it does give students the first-hand experience of writing software in an environment that teaches them how a computer works. Our premise is that students retain knowledge better when theory is combined with experience. In an engineering course, students construct prototypes; in a computer architecture course, students should write machine-level programs. In both cases, they have a memorable experience that gives them the confidence to work in any OS/machine-oriented environment.

Real Mode and Protected Mode This edition emphasizes 32-bit protected mode, but it still has three chapters devoted to real-mode programming. For example, there is an entire chapter on BIOS programming for the keyboard, video display (including graphics), and mouse. Another chapter covers MS-DOS programming using interrupts (system calls). Students can benefit from programming directly to hardware and the BIOS.

The examples in the first half of the book are nearly all presented as 32-bit text-oriented applications running in protected mode using the flat memory model. This approach is wonderfully simple because it avoids the complications of segment-offset addressing. Specially marked paragraphs and popup boxes point out occasional differences between protected mode and real mode programming. Most differences are abstracted by the book's parallel link libraries for real mode and protected mode programming.

Link Libraries We supply two versions of the link library that students use for basic input-output, simulations, timing, and other useful stuff. The 32-bit version (*Irvine32.lib*) runs in protected mode, sending its output to the Win32 console. The 16-bit version (*Irvine16.lib*) runs in real-address mode. Full source code for the libraries is supplied on the book's Web site. The link libraries are available only for convenience, not to prevent students from learning how to program input-output themselves. Students are encouraged to create their own libraries.

Included Software and Examples All the example programs were tested with Microsoft Macro Assembler Version 8.0. The 32-bit C++ applications in Chapter 12 were tested with Microsoft Visual C++ .NET. The real-address mode programs in Chapter 12 (linking to C++) were assembled with Borland Turbo Assembler (TASM).

Web Site Information Updates and corrections to this book may be found at the book's Web site, `http://www.asmirvine.com`, including additional programming projects for instructors to assign at the ends of chapters. If for some reason you cannot access this site, information about the book and a link to its current Web site can be found at **`www.prenhall.com`** by searching for the book title or for the author name "Kip Irvine."

Overall Goals

The following goals of this book are designed to broaden the student's interest and knowledge in topics related to assembly language:

- Intel IA-32 processor architecture and programming
- Real-address mode and protected mode programming
- Assembly language directives, macros, operators, and program structure
- Programming methodology, showing how to use assembly language to create system-level software tools and application programs
- Computer hardware manipulation
- Interaction between assembly language programs, the operating system, and other application programs

One of our goals is to help students approach programming problems with a machine-level mind set. It is important to think of the CPU as an interactive tool, and to learn to monitor its operation as directly as possible. A debugger is a programmer's best friend, not only for catching errors, but as an educational tool that teaches about the CPU and operating system. We encourage students to look beneath the surface of high-level languages and to realize that most programming languages are designed to be portable and, therefore, independent of their host machines.

In addition to the short examples, this book contains hundreds of ready-to-run programs that demonstrate instructions or ideas as they are presented in the text. Reference materials, such as guides to MS-DOS interrupts and instruction mnemonics, are available at the end of the book.

Required Background The reader should already be able to program confidently in at least one other programming language, preferably Java, C, or C++. One chapter covers C++ interfacing, so it is very helpful to have a compiler on hand. I have used this book in the classroom with majors in both computer science and management information systems, and it has been used elsewhere in engineering courses.

Features

Complete Program Listings A companion CD-ROM contains all the source code from the examples in this book. Additional listings are available on the book's Web page. An extensive link library is supplied with the book, containing more than 30 procedures that simplify user input-output, numeric processing, disk and file handling, and string handling. In the beginning stages of the course, students can use this library to enhance their programs. Later, they can create their own procedures and add them to the library.

Programming Logic Two chapters emphasize boolean logic and bit-level manipulation. A conscious attempt is made to relate high-level programming logic to the low-level details of the machine. This approach helps students to create more efficient implementations and to better understand how compilers generate object code.

Hardware and Operating System Concepts The first two chapters introduce basic hardware and data representation concepts, including binary numbers, CPU architecture, status flags, and memory mapping. A survey of the computer's hardware and a historical perspective of the Intel processor family helps students to better understand their target computer system.

Structured Programming Approach Beginning with Chapter 5, procedures and functional decomposition are emphasized. Students are given more complex programming exercises, requiring them to focus on design before starting to write code.

Disk Storage Concepts Students learn the fundamental principles behind the disk storage system on MS-Windows–based systems from hardware and software points of view.

Creating Link Libraries Students are free to add their own procedures to the book's link library and create new libraries. They learn to use a toolbox approach to programming and to write code that is useful in more than one program.

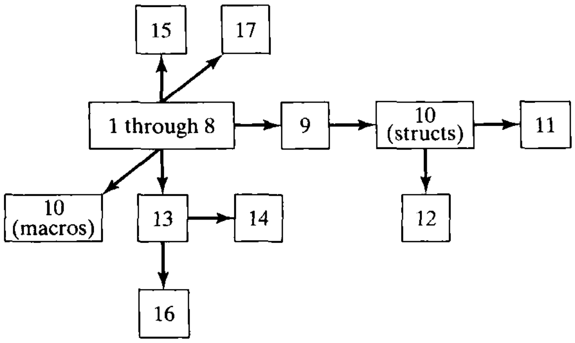
Macros and Structures A chapter is devoted to creating structures, unions, and macros, which are esential in assembly language and systems programming. Conditional macros with advanced operators serve to make the macros more professional.

Interfacing to High-Level Languages A chapter is devoted to interfacing assembly language to C and C++. This is an important job skill for students who are likely to find jobs programming in high-level languages. They can learn to optimize their code and see examples of how C++ compilers optimize code.

Instructional Aids All the program listings are available on disk and on the Web. Instructors are provided a test bank, answers to review questions, solutions to programming exercises, and a Microsoft PowerPoint slide presentation for each chapter.

Summary of Chapters

Chapters 1 to 8 contain a basic foundation of assembly language and should be covered in sequence. After that, you have a fair amount of freedom. The following chapter dependency graph shows how later chapters depend on knowledge gained from other chapters. Chapter 10 was split into two parts for this graph because no other chapter depends on one's knowing how to create macros:



1. **Basic Concepts:** Applications of assembly language, basic concepts, machine language, and data representation.
2. **IA-32 Processor Architecture:** Basic microcomputer design, instruction execution cycle, IA-32 processor architecture, IA-32 memory management, components of a microcomputer, and the input-output system.
3. **Assembly Language Fundamentals:** Introduction to assembly language, linking and debugging, and defining constants and variables.

4. **Data Transfers, Addressing, and Arithmetic:** Simple data transfer and arithmetic instructions, assemble-link-execute cycle, operators, directives, expressions, JMP and LOOP instructions, and indirect addressing.
5. **Procedures:** Linking to an external library, description of the book's link library, stack operations, defining and using procedures, flowcharts, and top-down structured design.
6. **Conditional Processing:** Boolean and comparison instructions, conditional jumps and loops, high-level logic structures, and finite state machines.
7. **Integer Arithmetic:** Shift and rotate instructions with useful applications, multiplication and division, extended addition and subtraction, and ASCII and packed decimal arithmetic.
8. **Advanced Procedures:** Stack parameters, local variables, advanced PROC and INVOKE directives, and recursion.
9. **Strings and Arrays:** String primitives, manipulating arrays of characters and integers, two-dimensional arrays, sorting, and searching.
10. **Structures and Macros:** Structures, macros, conditional assembly directives, and defining repeat blocks.
11. **MS-Windows Programming:** Protected mode memory management concepts, using the Microsoft Windows API to display text and colors, and dynamic memory allocation.
12. **High-Level Language Interface:** Parameter passing conventions, inline assembly code, and linking assembly language modules to C and C++ programs.
13. **16-Bit MS-DOS Programming:** Calling MS-DOS interrupts for console and file input-output.
14. **Disk Fundamentals:** Disk storage systems, sectors, clusters, directories, file allocation tables, handling MS-DOS error codes, and drive and directory manipulation.
15. **BIOS-Level Programming:** Keyboard input, video text, graphics, and mouse programming.
16. **Expert MS-DOS Programming:** Custom-designed segments, runtime program structure, and Interrupt handling. Hardware control using I/O ports.
17. **Floating-Point Processing and Instruction Encoding:** Floating-point binary representation and floating-point arithmetic. Learning to program the IA-32 Floating-Point Unit. Understanding the encoding of IA-32 machine instructions.

Appendix A: MASM Reference
Appendix B: The IA-32 Instruction Set
Appendix C: BIOS and MS-DOS Interrupts
Appendix D: Answers to Review Questions

Reference Materials

Web Site The author maintains an active Web site at **www.asmirvine.com**.

Help File Help file (in Windows Help Format) by Gerald Cahill of Antelope Valley College. Documents the book's link libraries, as well as Win32 data structures.

Assembly Language Workbook An interactive workbook is included on the book's Web site, covering such important topics as number conversions, addressing modes, register usage, Debug programming, and floating-point binary numbers. The content pages are HTML documents, making it easy for students and instructors to add their own customized content. This workbook is also available on the book's Web site.

Debugging Tools Tutorials on using Microsoft CodeView, Microsoft Visual Studio, and Microsoft Windows Debugger (WinDbg).

BIOS and MS-DOS Interrupts Appendix C contains a brief listing of the most-often-used INT 10h (video), INT 16h (keyboard), and INT 21h (MS-DOS) functions.

Instruction Set Appendix B lists most nonprivileged instructions for the IA-32 processor family.

For each instruction, we describe its effect, show its syntax, and show which flags are affected.

PowerPoint Presentations A complete set of Microsoft PowerPoint presentations written by the author.

Acknowledgments

Special thanks are due to Tracy Dunkleberger, Executive Editor for Computer Science at Prentice Hall, who provided friendly, helpful guidance during the writing of the fifth edition. Karen Ettinger did a terrific job as production editor, constantly keeping track of numerous minute details. Camille Trentacoste was the book's managing editor.

Fifth Edition

I offer my special thanks and gratitude to the following professors who boosted my morale, gave me great pedagogical tips, and tirelessly examined the entire book. They have been a huge influence on the development of this book, in some cases across multiple editions:

- **Gerald Cahill**, Antelope Valley College
- **James Brink**, Pacific Lutheran University
- **William Barrett**, San Jose State University

Many thanks to **Scott Blackledge** and **John Taylor**, both professional programmers, who proofread most of the manuscript and flagged numerous errors. Several people reviewed individual chapters:

- Jerry Joyce, Keene State College
- Tianzheng Wu, Mount Mercy College
- Ron Davis, Kennedy-King College
- David Topham, Ohlone College
- Harvey Nice, DePaul University

Fourth Edition

The following people were tremendously helpful in creating the fourth edition:

- Gerald Cahill, Antelope Valley College
- James Brink, Pacific Lutheran University
- Maria Kolatis, County College of Morris
- Tom Joyce, Chief Engineer at Premier Heart, LLC
- Jeff Wothke, Purdue Calumet University
- Tim Downey, Florida International University

The following individuals provided valuable proofreading help in the fourth edition:

- Andres Altamirano, Miami
- Courtney Amor, Los Angeles
- Scott Blackledge, Platform Solutions, Inc.
- Ronald Davis, Kennedy-King College
- Ata Elahi, Southern Connecticut State University
- Jose Gonzalez, Miami
- Leroy Highsmith, Southern Connecticut State University
- Sajid Iqbal, Faran Institute of Technology
- Charles Jones, Maryville College
- Vincent Kayes, Mount St. Mary College
- Eric Kobrin, Miami
- Pablo Maurin, Miami
- Barry Meaker, Design Engineer, Boeing Corporation
- Ian Merkel, Miami
- Sylvia Miner, Miami

- M. Nawaz, OPSTEC College of Computer Science
- Kam Ng, Chinese University of Hong Kong
- Hien Nguyen, Miami
- Ernie Philipp, Northern Virginia Community College
- Boyd Stephens, UGMO Research, LLC
- John Taylor, England
- Zachary Taylor, Columbia College
- Virginia Welsh, Community College of Baltimore County
- Robert Workman, Southern Connecticut State University
- Tianzheng Wu, Mount Mercy College
- Matthew Zukoski, Lehigh University

CONTENTS