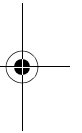
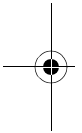


Assembly Language for Intel[®]-Based Computers

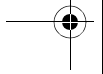
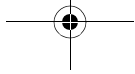
Fourth Edition

Kip R. Irvine
Florida International University



**Prentice
Hall**

Pearson Education Inc.
Upper Saddle River, NJ 07458



Library of Congress Cataloging-in-Publication Data

Irvine, Kip R.

Assembly language for Intel-based computers--4th edition / Kip R. Irvine.
CIP DATA AVAILABLE.

Vice President and Editorial Director, ECS: *Marcia Horton*
Executive Editor: *Petra Recter*
Editorial Assistant: *Renee Makras*
Vice President and Director of Production and Manufacturing, ESM: *David W. Riccardi*
Executive Managing Editor: *Vince O'Brien*
Assistant Managing Editor: *Camille Trencost*
Production Editor: *Irwin Zucker*
Manufacturing Manager: *Trudy Piscioti*
Manufacturing Buyer: *Lisa McDowell*
Director of Creative Services: *Paul Belfanti*
Creative Director: *Carole Anson*
Art Director: *Jayne Conte*
Cover Designer: *KIWI Design*
Cover Art: *Photograph of Shell, Dorling Kindersley Media Library*
Executive Marketing Manager: *Pamela Shaffer*
Marketing Assistant: *Barrie Reinhold*



© 2003, 1999 Pearson Education, Inc.
Pearson Education, Inc.
Upper Saddle River, New Jersey 07458

All rights reserved. No part of this book may be reproduced, in any format or by any means, without permission in writing from the publisher

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

TRADEMARK INFORMATION

TextPad is a trademark of Helios Software Solutions.
TASM and Turbo Debugger are trademarks of Borland International.
Microsoft Assembler (MASM), Windows NT, Windows Me, Windows 95, Windows 98, Windows 2000, Windows XP, MS-Windows, PowerPoint, Win32, DEBUG, WinDbg, MS-DOS, Visual Studio, Visual C++, and CodeView are registered trademarks of Microsoft Corporation.

Printed in the United States of America

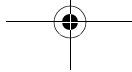
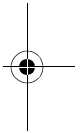
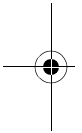
10 9 8 7 6 5 4 3 2 1

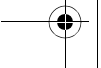
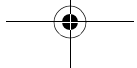
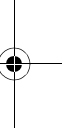
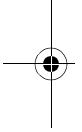
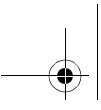
ISBN 0-13-091013-9

Pearson Education Ltd., *London*
Pearson Education Australia Pty. Limited, *Sydney*
Pearson Education Singapore Pte. Ltd.
Pearson Education North Asia Ltd. *Hong Kong*
Pearson Education Canada Inc., *Toronto*
Pearson Educación de México, S.A. de C.V.
Pearson Education—Japan, Inc., *Tokyo*
Pearson Education—Malaysia Pte. Ltd.
Pearson Education Inc., *Upper Saddle River, New Jersey*



To Jack and Candy Irvine







Preface

Assembly Language for Intel-Based Computers, Fourth Edition is based on the Intel IA-32 Processor architecture, seen from a programmer's point of view. It is appropriate as a text in the following types of college courses for computer science majors:

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

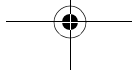
Although this book was originally designed as a programming textbook for community college students, it has gradually developed into much more. Currently, many universities use the book for their introductory computer architecture courses. At Florida International University, for example, this book is used in a course named *Fundamentals of Computer Systems*, which leads to a more comprehensive course in Computer Architecture.

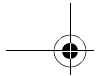
The present edition includes topics that lead naturally into subsequent courses in computer architecture, operating systems, and compiler writing:

- Virtual machine concept
- Elementary boolean operations
- Instruction execution cycle
- Memory access using clock cycles
- Interrupts and polling
- Multi-stage pipeline
- Superscalar architecture
- Multitasking
- Loading and executing programs
- Floating-point binary representation

Other topics relate specifically to Intel IA-32 architecture, using information gained from its manuals:

- IA-32 Protected Memory addressing and paging
- Memory segmentation in Real-address mode
- Interrupt handling
- Direct hardware I/O
- 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

There are a number of new features in this edition that relate to programming:

- A more comprehensive and logical explanation of data definition.
- A more careful explanation of addressing modes.
- A simplified link library that requires fewer input parameters for nearly all procedures. There are new procedures to dump the CPU registers and sections of memory, as well as a delay timer.
- An explanation and demonstration of top-down program design.
- Use of flowcharts as code-generation tools.
- Even more thorough coverage of assembly language directives, macros, and operators. For example, the PROC, PROTO, and INVOKE directives are thoroughly explained and demonstrated.
- More complete coverage of structures, including nested structures and arrays of structures.
- Block-structured IF, WHILE, and REPEAT statements (an advanced feature of MASM).
- Introduction to video graphics, using both BIOS and direct-memory mapping techniques.
- Mouse programming.
- Win32 Console programming, using calls to the Kernel32 Windows library.
- More array manipulation examples.

Still a Programming Book It is important to note that 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 the computer really works. The value of this cannot be underestimated, because they will retain a great deal more theoretical knowledge by having immediate contact with the machine. In an engineering course, students construct prototypes; in a software course, students write 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 Many professors have indicated a desire to move to 32-bit programming, using Intel's protected memory model. This edition primarily emphasizes 32-bit Protected mode, but it still has three chapters devoted exclusively to Real-mode programming. For example, there is an entire chapter on BIOS programming for the keyboard, video display (including graphics), and mouse. There is another chapter exclusively on MS-DOS

programming using interrupt (function) calls. It is very beneficial for students to have some experience programming directly for firmware and hardware.

The examples in the first part of the book are nearly all presented as 32-bit text-oriented applications running in Protected mode using the flat memory model. This is extremely straightforward. No longer do students have to deal with segment-offset addressing. There are specially marked paragraphs and popup boxes that note the small differences between Protected mode and Real-mode programming. Most of the differences are hidden away in the book's two link libraries.

Link Libraries There are two versions of the link library that students use for basic input-output in this book. The 32-bit version (*Irvine32.lib*) works in Win32 Console mode, under any version of MS-Windows. The 16-bit version (*Irvine16.lib*) works under MS-DOS, MS-Windows, and a Linux DOS emulator. In later chapters, all the functions in these two libraries are exposed, and readers can modify the libraries as they wish. It is important to realize that the link libraries are there only for convenience, not to prevent students from learning how to program input-output themselves.

Included Software and Examples All the example programs have been tested with the Microsoft Macro Assembler Version 6.15. For the most part, the programs will assemble with Borland TASM 4.0 and 5.0, but there are some features that Borland does not fully support.

Web Site Information Updates and corrections to this book may be found at the book's Web site, including additional programming projects for professors to assign at the ends of chapters:

<http://www.nuvisionmiami.com/books/asm>

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 full author name "Kip Irvine." The author's e-mail address is kip@nuvisionmiami.com

Overall Goals

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

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

One of my 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 each

of its actions 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. I 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, *Assembly Language for Intel-Based Computers* contains more than 115 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. There is a comprehensive link library that makes the user interface much more accessible for students writing their first programs. The macro library included with the book may also provide inspiration for further development by professors and students.

Required Background The reader should already be able to program confidently in at least one other programming language, preferably Pascal, Java, C, or C++. One chapter goes into C++ interfacing in some depth, 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. I used Microsoft Visual C++ 6.0 and Borland C++ 5.0 for the examples that deal with high-level language interfacing.

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 author'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. Students are given the complete source code for the 16-bit and 32-bit link libraries.

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 helps students to create more efficient implementations and to better understand how language 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 module decomposition are strongly emphasized. Students are given more complex programming problems that require the ability to carefully structure their code and to deal with complexity.

Disk Storage Concepts Students learn the fundamental principles behind the disk storage system on the PC, from both hardware and software points of view.

Creating Link Libraries Students are free to add their own procedures to the book's link library and can create libraries of their own. 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 important in both assembly language and high-level languages. 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 actual 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 all review questions, solutions to programming exercises, and a Microsoft PowerPoint slide presentation for each chapter.

Presentation Sequence

Chapters 1–8 represent the basic foundation of assembly language and should be covered in sequence. A great deal of effort went into making these chapters flow smoothly.

- 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 frames, local variables, parameter declarations, recursion, and advanced parameter passing.

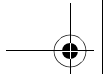
Chapters 9–16 may be covered in any order, giving instructors the opportunity to choose topics that are most relevant to their courses.

- 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. 32-Bit Windows Programming:** Protected mode memory management, and using the Microsoft Windows API to display text and colors on the console.
- 12. High-Level Language Interface:** Parameter passing conventions, inline assembly code, and linking assembly language modules to C/C++ programs.
- 13. 16-Bit MS-DOS Programming:** Calling MS-DOS interrupts for both console and file input-output.
- 14. Disk Fundamentals:** Disk storage systems, sectors, clusters, directories, file allocation table, handling MS-DOS error codes, and drive and directory manipulation.
- 15. BIOS-Level Programming:** Keyboard input, video text and graphics programming, and mouse programming.
- 16. Expert MS-DOS Programming:** Custom-designed segments, runtime program structure, and Interrupt handling.
- 17. Advanced Topics (on the enclosed CD-ROM):** Hardware control using I/O ports, instruction encoding, floating-point binary representation, and floating-point arithmetic.
 - **Appendix A:** Installing and Using the Assembler
 - **Appendix B:** The Intel Instruction Set
 - **Appendix C:** BIOS and MS-DOS Interrupts
 - **Appendix D:** MASM Reference

Reference Materials

In my own assembly course, I rely heavily on instructional materials such as tutorials, review questions, electronic slide shows, and workbooks. In that spirit, I have tried to provide ongoing support for instructors. If you find that something important is missing, please contact me and I may be able to provide it. The following reference information is included either in the book, on the accompanying CD-ROM, or on my Web site.

Assembly Language Workbook An interactive workbook is included on the attached CD-ROM, 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 professors to add their own customized content. This workbook is also available on my 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 taken from my own classroom lectures is available on the instructor Web site.

Answers to Review Questions Answers to all the odd-numbered review questions are available on the book's Web site. Answers to the even-numbered questions are available via the instructor Web site.

Acknowledgments

Special thanks are due to Petra Recter, Senior Computer Science Editor at Prentice Hall, who provided friendly, helpful guidance during the writing of the fourth edition. Irwin Zucker did a terrific job as production editor, constantly keeping track of numerous minute details. Bob Englehardt was a great help when preparing the book's CD-ROM. Camille Trentacoste was the book's managing editor.

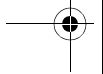
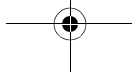
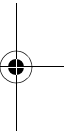
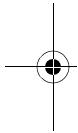
I offer my special thanks and gratitude to the following three professors who boosted my morale, gave me great pedagogical tips, and tirelessly examined the entire book:

- **Gerald Cahill** from Antelope Valley College, who offered numerous excellent suggestions and corrections. A great many of his ideas became reality in this book.
- **James Brink** of Pacific Lutheran University gave me many great suggestions. His own 32-bit link library inspired me to create one for this book.
- **Maria Kolatis** of the County College of Morris provided incisive, in-depth reviews of my chapters that forced me to rethink the presentation of many topics.

In addition, three people contributed a great deal of their time either by proofreading my book or developing examples that inspired me:

- **Tom Joyce**, Chief Engineer at Premier Heart, LLC.
- **Jeff Wothke** of Purdue Calumet University.
- **Tim Downey** of Florida International University.

Several of my top students at Florida International University read the manuscript and made valuable suggestions: Sylvia Miner, Eric Kobrin, Jose Gonzalez, Ian Merkel, Pablo Maurin, and Hien Nguyen. Andres Altamirano wrote excellent solutions for many programming exercises.

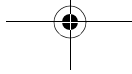
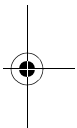
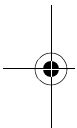


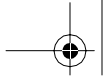


Proofreaders Many thanks to the following individuals for proofreading individual chapters. Unless otherwise noted, all are teaching faculty:

- Courtney Amor, a mathematics student at UCLA
- Ronald Davis, Kennedy-King College
- Ata Elahi, Southern Connecticut State University
- Leroy Highsmith, Southern Connecticut State University
- Sajid Iqbal, Faran Institute of Technology
- Charles Jones, Maryville College
- Vincent Kayes, Mount St. Mary College, Newburgh, New York
- Barry Meaker, Design Engineer, Boeing Corporation
- M. Nawaz, OPSTEC College of Computer Science
- Kam Ng, Chinese University of Hong Kong
- Ernie Philipp, Northern Virginia Community College
- Boyd Stephens, UGMO Research, LLC
- 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

Microsoft generously provided its Macro Assembler software for inclusion with this book. Helios Software Solutions Inc. permitted me to include an evaluation copy of the TextPad editor.





Contents

1 Basic Concepts 1

1.1 Welcome to Assembly Language 1

- 1.1.1 Some Good Questions to Ask 2
- 1.1.2 Assembly Language Applications 7
- 1.1.3 Section Review 8

1.2 Virtual Machine Concept 8

- 1.2.1 The History of PC Assemblers 11
- 1.2.2 Section Review 12

1.3 Data Representation 12

- 1.3.1 Binary Numbers 13
 - 1.3.1.1 Unsigned Binary Integers 13
 - 1.3.1.2 Translating Unsigned Binary Integers to Decimal 14
 - 1.3.1.3 Translating Unsigned Decimal Integers to Binary 14
- 1.3.2 Binary Addition 15
- 1.3.3 Integer Storage Sizes 16
- 1.3.4 Hexadecimal Integers 16
 - 1.3.4.1 Converting Unsigned Hexadecimal to Decimal 17
 - 1.3.4.2 Converting Unsigned Decimal to Hexadecimal 18
- 1.3.5 Signed Integers 18
 - 1.3.5.1 Two's Complement Notation 19
 - 1.3.5.2 Maximum and Minimum Values 21
- 1.3.6 Character Storage 21
- 1.3.7 Section Review 23

1.4 Boolean Operations 25

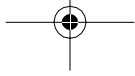
- 1.4.1 Truth Tables for Boolean Functions 27
- 1.4.2 Section Review 29

1.5 Chapter Summary 29

2 IA-32 Processor Architecture 31

2.1 General Concepts 31

- 2.1.1 Basic Microcomputer Design 32
- 2.1.2 Instruction Execution Cycle 33



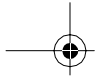
2.1.2.1	Multi-Stage Pipeline	34
2.1.2.2	Superscalar Architecture	36
2.1.3	Reading from Memory	37
2.1.4	How Programs Run	38
2.1.4.1	Load and Execute Process	38
2.1.4.2	Multitasking	39
2.1.5	Section Review	40
2.2	IA-32 Processor Architecture	41
2.2.1	Modes of Operation	41
2.2.2	Basic Execution Environment	41
2.2.2.1	Address Space	41
2.2.2.1	Basic Program Execution Registers	42
2.2.3	Floating-Point Unit	44
2.2.3.1	Other Registers	44
2.2.4	Intel Microprocessor History	45
2.2.4.1	IA-32 Processor Family	46
2.2.4.2	P6 Processor Family	46
2.2.4.3	CISC and RISC	46
2.2.5	Section Review	47
2.3	IA-32 Memory Management	48
2.3.1	Real-address Mode	48
2.3.1.1	20-bit Linear Address Calculation	49
2.3.2	Protected Mode	50
2.3.2.1	Flat Segmentation Model	50
2.3.2.2	Multi-Segment Model	51
2.3.2.3	Paging	52
2.3.3	Section Review	53
2.4	Components of an IA-32 Microcomputer	53
2.4.1	Motherboard	53
2.4.1.1	PCI Bus Architecture	54
2.4.1.2	Motherboard Chipset	54
2.4.2	Video Output	55
2.4.3	Memory	55
2.4.4	Input-Output Ports	56
2.4.5	Section Review	57
2.5	Input-Output System	57
2.5.1	How It All Works	57
2.5.2	Section Review	60
2.6	Chapter Summary	60
3	Assembly Language Fundamentals	63
3.1	Basic Elements of Assembly Language	64
3.1.1	Integer Constants	64

- 3.1.2 Integer Expressions 65
- 3.1.3 Real Number Constants 66
- 3.1.4 Character Constants 67
- 3.1.5 String Constants 67
- 3.1.6 Reserved Words 67
- 3.1.7 Identifiers 67
- 3.1.8 Directives 68
- 3.1.9 Instructions 68
 - 3.1.9.1 Label 69
 - 3.1.9.2 Instruction Mnemonic 70
 - 3.1.9.3 Operands 70
 - 3.1.9.4 Comments 71
- 3.1.10 Section Review 71
- 3.2 Example: Adding Three Integers 72**
 - 3.2.1 Program Listing 72
 - 3.2.2 Program Output 72
 - 3.2.3 Program Description 73
 - 3.2.3.1 Alternative Version of AddSub 74
 - 3.2.4 Program Template 76
 - 3.2.5 Section Review 76
- 3.3 Assembling, Linking, and Running Programs 77**
 - 3.3.1 The Assemble-Link-Execute Cycle 77
 - 3.3.1.1 Listing File 78
 - 3.3.1.2 Files Created or Updated by the Linker 79
 - 3.3.2 Section Review 80
- 3.4 Defining Data 80**
 - 3.4.1 Intrinsic Data Types 80
 - 3.4.2 Data Definition Statement 81
 - 3.4.3 Defining BYTE and SBYTE Data 81
 - 3.4.3.1 Multiple Initializers 82
 - 3.4.3.2 Defining Strings 83
 - 3.4.3.3 Using the DUP Operator 83
 - 3.4.4 Defining WORD and SWORD Data 84
 - 3.4.5 Defining DWORD and SDWORD Data 84
 - 3.4.6 Defining QWORD Data 85
 - 3.4.7 Defining TBYTE Data 85
 - 3.4.8 Defining Real Number Data 85
 - 3.4.9 Little Endian Order 86
 - 3.4.10 Adding Variables to the AddSub Program 87
 - 3.4.11 Declaring Uninitialized Data 87
 - 3.4.12 Section Review 88
- 3.5 Symbolic Constants 89**
 - 3.5.1 Equal-Sign Directive 89

3.5.2	Calculating the Sizes of Arrays and Strings	90
3.5.3	EQU Directive	91
3.5.4	TEXTEQU Directive	92
3.5.5	Section Review	93
3.6	Real-Address Mode Programming (Optional)	93
3.6.1	Basic Changes	94
3.6.1.1	The AddSub2 Program	94
3.7	Chapter Summary	95
3.8	Programming Exercises	96
4	Data Transfers, Addressing, and Arithmetic	97
4.1	Data Transfer Instructions	98
4.1.1	Introduction	98
4.1.2	Operand Types	98
4.1.3	Direct Memory Operands	99
4.1.4	MOV Instruction	100
4.1.5	Zero/Sign Extension of Integers	101
4.1.5.1	Copying Smaller Values to Larger Ones	101
4.1.5.2	MOVZX Instruction	102
4.1.5.3	MOVSX Instruction	103
4.1.6	LAHF and SAHF Instructions	103
4.1.7	XCHG Instruction	104
4.1.8	Direct-Offset Operands	104
4.1.9	Example Program (Moves)	105
4.1.10	Section Review	106
4.2	Addition and Subtraction	107
4.2.1	INC and DEC Instructions	107
4.2.2	ADD Instruction	108
4.2.3	SUB Instruction	108
4.2.4	NEG Instruction	109
4.2.5	Implementing Arithmetic Expressions	109
4.2.6	Flags Affected by Arithmetic	110
4.2.6.1	Zero and Sign Flags	110
4.2.6.2	Carry Flag (unsigned arithmetic)	110
4.2.6.3	Overflow Flag (signed arithmetic)	111
4.2.7	Example Program (AddSub3)	113
4.2.8	Section Review	114
4.3	Data-Related Operators and Directives	115
4.3.1	OFFSET Operator	115
4.3.1.1	OFFSET Example	115
4.3.2	ALIGN Directive	116

4.3.3	PTR Operator	117
4.3.4	TYPE Operator	118
4.3.5	LENGTHOF Operator	118
4.3.6	SIZEOF Operator	119
4.3.7	LABEL Directive	119
4.3.8	Section Review	120
4.4	Indirect Addressing	120
4.4.1	Indirect Operands	121
4.4.2	Arrays	122
4.4.3	Indexed Operands	123
4.4.4	Pointers	124
4.4.4.1	Using the TYPDEF Operator	125
4.4.5	Section Review	126
4.5	JMP and LOOP Instructions	127
4.5.1	JMP Instruction	127
4.5.2	LOOP Instruction	128
4.5.3	Summing an Integer Array	129
4.5.4	Copying a String	130
4.5.5	Section Review	131
4.6	Chapter Summary	132
4.7	Programming Exercises	133
5	Procedures	137
5.1	Introduction	137
5.2	Linking to an External Library	138
5.2.1	Background Information	138
5.2.2	Section Review	139
5.3	The Book's Link Library	140
5.3.1	Overview	140
5.3.2	Individual Procedure Descriptions	141
5.3.2.1	The Irvine32.inc Include File	147
5.3.3	Library Test Program	148
5.3.4	Section Review	152
5.4	Stack Operations	153
5.4.1	Runtime Stack	153
5.4.1.1	Push Operation	154
5.4.1.2	Pop Operation	155
5.4.1.3	Stack Applications	155
5.4.2	PUSH and POP Instructions	156
5.4.2.1	PUSH Instruction	156
5.4.2.2	POP Instruction	156

5.4.2.3	PUSHFD and POPFD Instructions	156
5.4.2.4	PUSHAD, PUSHA, POPAD, and POPA	157
5.4.2.5	Example: Reversing a String	157
5.4.3	Section Review	158
5.5	Defining and Using Procedures	159
5.5.1	PROC Directive	159
5.5.1.1	Defining a Procedure	159
5.5.1.2	Example: Sum of Three Integers	160
	Documenting Procedures	160
5.5.2	CALL and RET Instructions	161
5.5.2.1	Call and Return Example	161
5.5.2.2	Nested Procedure Calls	162
5.5.2.3	Local Labels and Global Labels	163
5.5.2.4	Passing Register Arguments to Procedures	164
5.5.3	Example: Summing an Integer Array	164
5.5.4	Flowcharts	165
5.5.5	Saving and Restoring Registers	166
5.5.5.1	USES Operator	166
5.5.6	Section Review	168
5.6	Program Design Using Procedures	169
5.6.1	Integer Summation Program (Design)	170
5.6.1.1	Integer Summation Implementation	172
5.6.2	Section Review	175
5.7	Chapter Summary	175
5.8	Programming Exercises	176
6	Conditional Processing	179
6.1	Introduction	180
6.2	Boolean and Comparison Instructions	180
6.2.1	The CPU Flags	181
6.2.2	AND Instruction	181
6.2.2.1	Converting Characters to Upper Case	182
6.2.3	OR Instruction	183
6.2.4	XOR Instruction	184
6.2.5	NOT Instruction	186
6.2.6	TEST Instruction	186
6.2.7	CMP Instruction	186
6.2.8	Setting and Clearing Individual CPU Flags	188
6.2.9	Section Review	188
6.3	Conditional Jumps	189
6.3.1	Conditional Structures	189
6.3.2	<i>Jcond</i> Instruction	190



- 6.3.3 Types of Conditional Jump Instructions 191
 - 6.3.3.1 Equality Comparisons 192
 - 6.3.3.2 Unsigned Comparisons 192
 - 6.3.3.3 Signed Comparisons 192
- 6.3.4 Conditional Jump Applications 193
 - 6.3.4.1 Testing Status Bits 193
 - 6.3.4.2 Application: Scanning an Array 194
 - 6.3.4.3 Application: String Encryption 195
- 6.3.5 Bit Testing Instructions (Optional) 198
 - 6.3.5.1 BT Instruction 198
 - 6.3.5.2 BTC Instruction 199
 - 6.3.5.3 BTR Instruction 199
 - 6.3.5.4 BTS Instruction 199
- 6.3.6 Section Review 200
- 6.4 Conditional Loop Instructions 200**
 - 6.4.1 LOOPZ and LOOPE Instructions 200
 - 6.4.2 LOOPNZ and LOOPNE Instructions 201
 - 6.4.3 Section Review 202
- 6.5 Conditional Structures 202**
 - 6.5.1 Block-Structured IF Statements 202
 - 6.5.2 Compound Expressions 204
 - 6.5.2.1 Logical AND Operator 204
 - 6.5.2.2 Logical OR Operator 205
 - 6.5.3 WHILE Loops 205
 - 6.5.3.1 Example: IF statement Nested in a Loop 206
 - 6.5.4 Table-Driven Selection 208
 - 6.5.5 Section Review 210
- 6.6 Application: Finite-State Machines 211**
 - 6.6.1 Validating an Input String 212
 - 6.6.2 Validating a Signed Integer 213
 - 6.6.3 Section Review 216
- 6.7 Using the .IF Directive (Optional) 217**
 - 6.7.1 Signed and Unsigned Comparisons 219
 - 6.7.2 Compound Expressions 220
 - 6.7.2.1 SetCursorPosition Example 220
 - 6.7.2.2 College Registration Example 221
 - 6.7.3 .REPEAT and .WHILE Directives 222
 - 6.7.3.1 Example: Loop Containing an IF Statement 222
- 6.8 Chapter Summary 223**
- 6.9 Programming Exercises 224**



7 Integer Arithmetic 227

7.1 Introduction 228

7.2 Shift and Rotate Instructions 228

- 7.2.1 Logical Shifts versus Arithmetic Shifts 229
- 7.2.2 SHL Instruction 229
- 7.2.3 SHR Instruction 230
- 7.2.4 SAL and SAR Instructions 231
- 7.2.5 ROL Instruction 231
- 7.2.6 ROR Instruction 232
- 7.2.7 RCL and RCR Instructions 232
- 7.2.8 SHLD/SHRD Instructions 233
- 7.2.9 Section Review 235

7.3 Shift and Rotate Applications 236

- 7.3.1 Shifting Multiple Doublewords 236
- 7.3.2 Binary Multiplication 237
- 7.3.3 Displaying Binary Bits 237
- 7.3.4 Isolating a Bit String 238
- 7.3.5 Section Review 239

7.4 Multiplication and Division Instructions 239

- 7.4.1 MUL Instruction 240
- 7.4.2 IMUL Instruction 241
- 7.4.3 DIV Instruction 242
- 7.4.4 Signed Integer Division 243
 - 7.4.4.1 CBW, CWD, CDQ Instructions 243
 - 7.4.4.2 The IDIV Instruction 243
 - 7.4.4.3 Divide Overflow 244
- 7.4.5 Implementing Arithmetic Expressions 245
- 7.4.6 Section Review 247

7.5 Extended Addition and Subtraction 248

- 7.5.1 ADC Instruction 248
- 7.5.2 Extended Addition Example 249
- 7.5.3 SBB Instruction 250
- 7.5.4 Section Review 250

7.6 ASCII and Packed Decimal Arithmetic (Optional) 251

- 7.6.1 AAA Instruction 252
- 7.6.2 AAS Instruction 253
- 7.6.3 AAM Instruction 253
- 7.6.4 AAD Instruction 253
- 7.6.5 Packed Decimal Integers 254
 - 7.6.5.1 DAA Instruction 254
 - 7.6.5.2 DAS Instruction 254

7.7	Chapter Summary	255
7.8	Programming Exercises	256
8	Advanced Procedures	259
8.1	Introduction	259
8.2	Local Variables	260
8.2.1	LOCAL Directive	261
8.2.2	Section Review	263
8.3	Stack Parameters	263
8.3.1	INVOKE Directive	264
8.3.1.1	DDR Operator	265
8.3.2	PROC Directive	266
8.3.2.1	Examples	267
8.3.3	PROTO Directive	268
8.3.3.1	ArraySum Example	269
8.3.4	Passing by Value or by Reference	269
8.3.5	Parameter Classifications	271
8.3.6	Example: Exchanging Two Integers	271
8.3.7	Trouble-Shooting Tips	272
8.3.7.1	Saving and Restoring Registers	272
8.3.7.2	Wrong Operand Sizes	273
8.3.7.3	Passing the Wrong Type of Pointer	274
8.3.7.4	Passing Immediate Values	274
8.3.8	Section Review	274
8.4	Stack Frames	275
8.4.1	Memory Models	276
8.4.2	Language Specifiers	277
8.4.2.1	STDCALL Specifier	277
8.4.2.2	C Specifier	278
8.4.2.3	PASCAL Specifier	278
8.4.3	Explicit Access to Stack Parameters	278
8.4.3.1	Saving and Restoring Registers	280
8.4.4	Passing Arguments by Reference	280
8.4.4.1	ArrayFill Example	281
8.4.4.2	LEA Instruction	282
8.4.5	Creating Local Variables	282
8.4.6	ENTER and LEAVE Instructions (Optional)	283
8.4.7	Section Review	285
8.5	Recursion	285
8.5.1	Recursively Calculating a Sum	286
8.5.2	Calculating a Factorial	288
8.5.3	Section Review	290

- 8.6 Creating Multimodule Programs 290**
 - 8.6.1 Example: ArraySum Program 291
 - 8.6.1.1 Include File: Function Prototypes 292
 - 8.6.1.2 Main Module 292
 - 8.6.1.3 PromptForIntegers Module 293
 - 8.6.1.4 ArraySum Module 294
 - 8.6.1.5 DisplaySum Module 295
 - 8.6.1.6 Batch File for Assembling and Linking 295
 - 8.6.2 Section Review 296
- 8.7 Chapter Summary 296**
- 8.8 Programming Exercises 298**
- 9 Strings and Arrays 301**
 - 9.1 Introduction 301**
 - 9.2 String Primitive Instructions 302**
 - 9.2.1 MOVS_B, MOVS_W, and MOVS_D 304
 - 9.2.2 CMPS_B, CMPS_W, and CMPS_D 304
 - 9.2.2.1 Example: Comparing Two Strings 306
 - 9.2.3 SCAS_B, SCAS_W, and SCAS_D 307
 - 9.2.4 STOS_B, STOS_W, and STOS_D 308
 - 9.2.5 LODS_B, LODS_W, and LODS_D 308
 - 9.2.6 Section Review 309
 - 9.3 Selected String Procedures 309**
 - 9.3.1 Str_compare Procedure 310
 - 9.3.2 Str_length Procedure 311
 - 9.3.3 Str_copy Procedure 311
 - 9.3.4 Str_trim Procedure 312
 - 9.3.5 Str_ucase Procedure 314
 - 9.3.6 Section Review 315
 - 9.4 Two-Dimensional Arrays 315**
 - 9.4.1 Base-Index Operands 315
 - 9.4.2 Base-Index Displacement 317
 - 9.4.3 Section Review 318
 - 9.5 Searching and Sorting Integer Arrays 318**
 - 9.5.1 Bubble Sort 319
 - 9.5.2 Binary Search 321
 - 9.5.2.1 Test Program 324
 - 9.5.3 Section Review 328
 - 9.6 Chapter Summary 328**
 - 9.7 Programming Exercises 330**

10 Structures and Macros 333

10.1 Structures 334

- 10.1.1 Defining Structures 334
- 10.1.2 Declaring Structure Variables 335
- 10.1.3 Referencing Structure Variables 336
- 10.1.4 Example: Displaying the System Time 338
- 10.1.5 Nested Structures 340
- 10.1.6 Example: Drunkard's Walk 341
- 10.1.7 Declaring and Using Unions 344
- 10.1.8 Section Review 346

10.2 Macros 347

- 10.2.1 Overview 347
- 10.2.2 Defining Macros 348
- 10.2.3 Invoking Macros 349
- 10.2.4 Macro Examples 350
 - 10.2.4.1 mWriteStr Macro 351
 - 10.2.4.2 mReadStr Macro 352
 - 10.2.4.3 mGotoxy Macro 352
 - 10.2.4.4 mDumpMem Macro 353
 - 10.2.4.5 Macros Containing Code and Data 354
- 10.2.5 Nested Macros 355
- 10.2.6 Example Program: Wrappers 356
- 10.2.7 Section Review 357

10.3 Conditional-Assembly Directives 358

- 10.3.1 Checking for Missing Arguments 359
- 10.3.2 Default Argument Initializers 360
- 10.3.3 Boolean Expressions 360
- 10.3.4 IF, ELSE, and ENDIF Directives 361
- 10.3.5 The IFIDN and IFIDNI Directives 362
- 10.3.6 Special Operators 363
 - 10.3.6.1 Substitution Operator (&) 363
 - 10.3.6.2 Expansion Operator (%) 364
 - 10.3.6.3 Literal-Text Operator (<>) 366
 - 10.3.6.4 Literal-Character Operator (!) 366
- 10.3.7 Macro Functions 367
- 10.3.8 Section Review 369

10.4 Defining Repeat Blocks 370

- 10.4.1 WHILE Directive 370
- 10.4.2 REPEAT Directive 371
- 10.4.3 FOR Directive 371
- 10.4.4 FORC Directive 372
- 10.4.5 Example: Linked List 373
- 10.4.6 Section Review 375

10.5	Chapter Summary	375
10.6	Programming Exercises	376
11	32-Bit Windows Programming	379
11.1	Win32 Console Programming	379
11.1.1	Background Information	381
11.1.1.1	Windows Data Types	382
11.1.1.2	Console Handles	383
11.1.2	Win32 Console Functions	384
11.1.3	Console Input	386
11.1.3.1	ReadConsole Function	387
11.1.3.2	Single-Character Input	388
11.1.4	Console Output	389
11.1.4.1	Data Structures	390
11.1.4.2	WriteConsole Function	390
11.1.4.3	Example Program: Console1	390
11.1.4.4	WriteConsoleOutputCharacter Function	391
11.1.5	Reading and Writing Files	392
11.1.5.1	CreateFile Function	392
11.1.5.2	CloseHandle Function	395
11.1.5.3	ReadFile Function	395
11.1.5.4	WriteFile Function	395
11.1.5.5	Example WriteFile Program	396
11.1.5.6	Moving the File Pointer	397
11.1.5.7	Example ReadFile Program	397
11.1.6	Console Window Manipulation	398
11.1.6.1	SetConsoleTitle	399
11.1.6.2	GetConsoleScreenBufferInfo	399
11.1.6.3	SetConsoleWindowInfo Function	400
11.1.6.4	SetConsoleScreenBufferSize Function	402
11.1.7	Controlling the Cursor	402
11.1.7.1	GetConsoleCursorInfo Function	402
11.1.7.2	SetConsoleCursorInfo Function	402
11.1.7.3	SetConsoleCursorPosition	403
11.1.8	Controlling the Text Color	403
11.1.8.1	SetConsoleTextAttribute Function	403
11.1.8.2	WriteConsoleOutputAttribute Function	403
11.1.8.3	Example WriteColors Program	403
11.1.9	Time and Date Functions	405
11.1.9.1	GetLocalTime and SetLocalTime	406
11.1.9.2	GetTickCount Function	407
11.1.9.3	Sleep Function	407
11.1.9.4	GetDateTime Procedure	408
11.1.9.5	Creating a Stopwatch Timer	409
11.1.10	Section Review	410

11.2 Writing a Graphical Windows Application 411

- 11.2.1 Necessary Structures 412
- 11.2.2 The MessageBox Function 413
- 11.2.3 The WinMain Procedure 414
- 11.2.4 The WinProc Procedure 414
- 11.2.5 The ErrorHandler Procedure 415
- 11.2.6 Program Listing 416
 - 11.2.6.1 Running the Program 419
- 11.2.7 Section Review 420

11.3 IA-32 Memory Management 421

- 11.3.1 Linear Addresses 421
 - 11.3.1.1 Translating Logical Addresses to Linear Addresses 421
 - 11.3.1.2 Paging 423
 - 11.3.1.3 Descriptor Tables 424
 - 11.3.1.4 Segment Descriptor Details 425
- 11.3.2 Page Translation 425
 - 11.3.2.1 MS-Windows Virtual Machine Manager 426
- 11.3.3 Section Review 427

11.4 Chapter Summary 428**11.5 Programming Exercises 429****12 High-Level Language Interface 431****12.1 Introduction 431**

- 12.1.1 General Conventions 431
- 12.1.2 Section Review 433

12.2 Inline Assembly Code 433

- 12.2.1 `__asm` Directive in Microsoft Visual C++ 433
 - 12.2.1.1 Using the LENGTH, TYPE, and SIZE Operators 435
- 12.2.2 File Encryption Example 436
 - 12.2.2.1 Procedure Call Overhead 437
- 12.2.3 Section Review 439

12.3 Linking to C++ Programs 439

- 12.3.1 Linking to Borland C++ 440
- 12.3.2 ReadSector Example 441
 - 12.3.2.1 Main C++ Program That Calls ReadSector 442
 - 12.3.2.2 Assembly Language Module 444
- 12.3.3 Example: Large Random Integers 446
- 12.3.4 Using Assembly Language to Optimize C++ Code 448
 - 12.3.4.1 FindArray Code Generated by Visual C++ 449
 - 12.3.4.2 Linking MASM to Visual C++ 450
- 12.3.5 Section Review 454

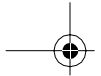
12.4 Chapter Summary 455

12.5	Programming Exercises	456
13	16-Bit MS-DOS Programming	457
13.1	MS-DOS and the IBM-PC	457
13.1.1	Memory Organization	458
13.1.2	Redirecting Input-Output	460
13.1.3	Software Interrupts	461
13.1.4	INT Instruction	461
13.1.4.1	Interrupt Vectoring	461
13.1.4.2	Common Interrupts	462
13.1.5	Section Review	462
13.2	MS-DOS Function Calls (INT 21h)	463
13.2.1	Selected Output Functions	464
13.2.2	Hello World Program Example	467
13.2.3	Selected Input Functions	467
13.2.3.1	Example: String Encryption Program	469
13.2.3.2	Int 21h Function 3Fh	470
13.2.4	Date/Time Functions	472
13.2.4.1	Example: Displaying the Time and Date	474
13.2.5	Section Review	476
13.3	Standard MS-DOS File I/O Services	476
13.3.0.1	Create or Open File (716Ch)	478
13.3.1	Close File Handle (3Eh)	479
13.3.2	Move File Pointer (42h)	480
13.3.2.1	Get File Creation Date and Time	481
13.3.3	Selected Library Procedures	481
13.3.3.1	ReadString	482
13.3.3.2	WriteString	482
13.3.4	Example: Read and Copy a Text File	483
13.3.5	Reading the MS-DOS Command Tail	485
13.3.6	Example: Creating a Binary File	487
13.3.7	Section Review	490
13.4	Chapter Summary	491
13.5	Chapter Exercises	492
14	Disk Fundamentals	495
14.1	Disk Storage Systems	495
14.1.1	Tracks, Cylinders, and Sectors	496
14.1.2	Disk Partitions (Volumes)	498
14.1.3	Section Review	499
14.2	File Systems	500
14.2.1	FAT12	501

14.2.2	FAT16	501
14.2.3	FAT32	501
14.2.4	NTFS	502
14.2.5	Primary Disk Areas	503
14.2.6	Section Review	504
14.3	Disk Directory	505
14.3.1	MS-DOS Directory Structure	505
14.3.2	Long Filenames in MS-Windows	508
14.3.3	File Allocation Table (FAT)	510
14.3.4	Section Review	511
14.4	Reading and Writing Disk Sectors (7305h)	511
14.4.1	Sector Display Program	513
14.4.2	Section Review	517
14.5	System-Level File Functions	517
14.5.1	Get Disk Free Space (7303h)	518
14.5.1.1	Disk Free Space Program	519
14.5.2	Create Subdirectory (39h)	521
14.5.3	Remove Subdirectory (3Ah)	521
14.5.4	Set Current Directory (3Bh)	522
14.5.5	Get Current Directory (47h)	522
14.5.6	Section Review	522
14.6	Chapter Summary	523
14.7	Programming Exercises	524
15	BIOS-Level Programming	527
15.1	Introduction	527
15.1.1	BIOS Data Area	528
15.2	Keyboard Input with INT 16h	529
15.2.1	How the Keyboard Works	530
15.2.2	INT 16h Functions	531
15.2.2.1	Set Typematic Rate (03h)	531
15.2.2.2	Push Key into Keyboard Buffer (05h)	531
15.2.2.3	Wait for Key (10h)	532
15.2.2.4	Check Keyboard Buffer (11h)	533
15.2.2.5	Get Keyboard Flags	534
15.2.2.6	Clearing the Keyboard Buffer	535
15.2.3	Section Review	537
15.3	VIDEO Programming with INT 10h	537
15.3.1	Basic Background	537
15.3.1.1	Three Levels of Access	537
15.3.1.2	Running Programs in Full-Screen Mode	538

- 15.3.1.3 Understanding Video Text 538
- 15.3.2 Controlling the Color 539
 - 15.3.2.1 Mixing Primary Colors 539
 - 15.3.2.2 Attribute Byte 540
- 15.3.3 INT 10h Video Functions 541
 - 15.3.3.1 Set Video Mode (00h) 542
 - 15.3.3.2 15.3.3.Set Cursor Lines (01h) 543
 - 15.3.3.3 Set Cursor Position (02h) 544
 - 15.3.3.4 Get Cursor Position and Size (03h) 544
 - 15.3.3.5 Scroll Window Up (06h) 546
 - 15.3.3.6 Example: Writing Text to a Window 547
 - 15.3.3.7 Scroll Window Down (07h) 548
 - 15.3.3.8 Read Character and Attribute (08h) 548
 - 15.3.3.9 Write Character and Attribute (09h) 548
 - 15.3.3.10 Write Character (0Ah) 549
 - 15.3.3.11 Toggle Blinking and Intensity Modes 550
 - 15.3.3.12 Get Video Mode Information (0Fh) 550
 - 15.3.3.13 Write String in Teletype Mode (13h) 551
 - 15.3.3.14 Example: Displaying a Color String 552
- 15.3.4 Library Procedure Examples 554
 - 15.3.4.1 Gotoxy Procedure 554
 - 15.3.4.2 Clrscr Procedure 554
- 15.3.5 Section Review 555
- 15.4 Drawing Graphics Using INT 10h 555**
 - 15.4.1 INT 10h Pixel-Related Functions 556
 - 15.4.1.1 Write Graphics Pixel (0Ch) 556
 - 15.4.1.2 Read Graphics Pixel (0Dh) 557
 - 15.4.2 DrawLine Program 557
 - 15.4.3 Cartesian Coordinates Program 559
 - 15.4.4 Converting Cartesian Coordinates to Screen Coordinates 562
 - 15.4.5 Section Review 563
- 15.5 Memory-Mapped Graphics 563**
 - 15.5.1 Mode 13h: 320 X 200, 256 Colors 563
 - 15.5.2 Memory-Mapped Graphics Program 565
 - 15.5.3 Section Review 568
- 15.6 Mouse Programming 568**
 - 15.6.1 Mouse INT 33h Functions 568
 - 15.6.1.1 Reset Mouse and Get Status 568
 - 15.6.1.2 Showing and Hiding the Mouse Pointer 569
 - 15.6.1.3 Get Mouse Position and Status 570
 - 15.6.1.4 Set Mouse Position 571
 - 15.6.1.5 Get Button Presses and Releases 571
 - 15.6.1.6 Setting Horizontal and Vertical Limits 573
 - 15.6.1.7 Miscellaneous Mouse Functions 573

15.6.2	Mouse Tracking Program	574
15.6.3	Section Review	579
15.7	Chapter Summary	580
15.8	Chapter Exercises	580
16	Expert MS-DOS Programming	583
16.1	Introduction	583
16.2	Defining Segments	584
16.2.1	Simplified Segment Directives	584
16.2.2	Explicit Segment Definitions	586
16.2.2.1	Align Type	587
16.2.2.2	Combine Type	587
16.2.2.3	Class Type	588
16.2.2.4	ASSUME Directive	588
16.2.2.5	Example: Multiple Data Segments	588
16.2.3	Segment Overrides	589
16.2.4	Combining Segments	590
16.2.5	Section Review	592
16.3	Runtime Program Structure	592
16.3.1	COM Programs	593
16.3.2	EXE Programs	595
16.3.2.1	Memory Usage	595
16.3.2.2	EXE Header	596
16.3.3	Section Review	597
16.4	Interrupt Handling	597
16.4.1	Hardware Interrupts	599
16.4.2	Interrupt Control Instructions	600
16.4.3	Writing a Custom Interrupt Handler	601
16.4.3.1	Ctrl-Break Handler Example	602
16.4.4	Terminate and Stay Resident Programs	604
16.4.4.1	Keyboard Example	604
16.4.5	Application: The No_Reset Program	605
16.4.6	Section Review	609
16.5	Chapter Summary	609
17	Advanced Topics	
17.1	Hardware Control Using I/O Ports	17-1
17.1.1	Input-Output Ports	17-1
17.1.1.1	PC Sound Program	17-2
17.2	Intel Instruction Encoding	17-4
17.2.1	Single-Byte Instructions	17-5



- 17.2.2 Immediate Operands 17-6
- 17.2.3 Register-Mode Instructions 17-6
- 17.2.4 Memory-Mode Instructions 17-7
 - 17.2.4.1 MOV Instruction Examples 17-9
- 17.2.5 Section Review 17-11

17.3 Floating-Point Arithmetic 17-12

- 17.3.1 IEEE Binary Floating-Point Representation 17-12
 - 17.3.1.1 The Sign 17-12
 - 17.3.1.2 The Mantissa 17-12
- 17.3.2 The Exponent 17-13
- 17.3.3 Normalizing the Mantissa 17-14
- 17.3.4 Creating the IEEE Bit Representation 17-15
- 17.3.5 Converting Decimal Fractions to Binary Reals 17-15
- 17.3.6 IA-32 Floating Point Architecture 17-17
- 17.3.7 Instruction Formats 17-18
- 17.3.8 Floating-Point Code Examples 17-21
 - 17.3.8.1 Example 1: Evaluating an Expression 17-21

Appendix A: Installing and Using the Assembler 611

Appendix B: The Intel Instruction Set 617

Appendix C: BIOS and MS-DOS Interrupts 649

Appendix D: MASM Reference 661

Index 689

