

PEARSON
Prentice
Hall

大学计算机教育国外著名教材系列



Assembly Language for x86 Processors Sixth Edition

汇编语言程序设计 (第6版)



Kip R. Irvine 著

清华大学出版社

大学计算机教育国外著名教材系列（影印版）

Assembly Language for x86 Processors

Sixth Edition

汇编语言程序设计

（第6版）

Kip R.Irvine

清华大学出版社
北 京

English reprint edition copyright © 2011 by **PEARSON EDUCATION ASIA LIMITED and TSINGHUA UNIVERSITY PRESS.**

Original English language title from Proprietor's edition of the Work.

Original English language title: **Assembly Language for x86 Processors, Sixth Edition by Kip R. Irvine, Copyright © 2011**
All Rights Reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Prentice Hall, Inc.

This edition is authorized for sale and distribution only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong, Macao SAR and Taiwan).
本书影印版由 Pearson Education (培生教育出版集团) 授权给清华大学出版社出版发行。

For sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macao SAR).
仅限于中华人民共和国境内(不包括中国香港、澳门特别行政区和中国台湾地区)销售发行。

北京市版权局著作权合同登记号 图字 01-2011-1756 号

本书封面贴有 **Pearson Education**(培生教育出版集团)激光防伪标签, 无标签者不得销售。
版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目(CIP)数据

汇编语言程序设计=Assembly Language for x86 Processors·第6版: 英文/(美)埃尔温
(Irvine, K. R.) 著.--影印本 --北京: 清华大学出版社, 2011 10
(大学计算机教育国外著名教材系列)

ISBN 978-7-302-26030-1

I. ①汇… II. ①埃… III ①汇编语言—程序设计—英文 IV ①TP313

中国版本图书馆 CIP 数据核字 (2011) 第 131333 号

责任编辑: 龙啟铭

责任印制: 王秀菊

出版发行: 清华大学出版社

<http://www.tup.com.cn>

社 总 机: 010-62770175

投稿与读者服务: 010-62795954, jsjic@tup.tsinghua.edu.cn

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

地 址: 北京清华大学学研大厦 A 座

邮 编: 100084

邮 购: 010-62786544

印 装 者: 三河市金元印装有限公司

发 行 者: 全国新华书店

开 本: 148×210 印张: 23.375

版 次: 2011 年 10 月第 1 版

印 次: 2011 年 10 月第 1 次印刷

印 数: 1~3000

定 价: 39.00 元

产品编号: 037755-01

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等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

To Jack and Candy Irvine

PREFACE

Assembly Language for x86 Processors, Sixth Edition, teaches assembly language programming and architecture for Intel and AMD 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)**, running on Windows 98, XP, Vista, and Windows 7. Although this book was originally designed as a programming textbook for college students, it serves as an effective supplement to computer architecture courses. As a testament to its popularity, previous editions have been translated into Spanish, 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
- Instruction set architecture
- Elementary Boolean operations
- Instruction execution cycle
- Memory access and handshaking
- Interrupts and polling
- Hardware-based I/O
- Floating-point binary representation

Other topics relate specially to Intel and AMD architecture:

- Protected memory and paging
- Memory segmentation in real-address mode
- 16-bit interrupt handling
- MS-DOS and BIOS system calls (interrupts)
- Floating-point unit architecture and programming
- 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

What's New in the Sixth Edition

In this revision, we have placed a strong emphasis on improving the descriptions of important programming concepts and relevant program examples.

- We have added numerous step-by-step descriptions of sample programs, particularly in Chapters 1–8.
- Many new illustrations have been inserted into the chapters to improve student comprehension of concepts and details.
- **Java Bytecodes:** The Java Virtual Machine (JVM) provides an excellent real-life example of a stack-oriented architecture. It provides an excellent contrast to x86 architecture. Therefore, in Chapters 8 and 9, the author explains the basic operation of Java bytecodes with short illustrative examples. Numerous short examples are shown in disassembled bytecode format, followed by detailed step-by-step explanations.
- Selected programming exercises have been replaced in the first 8 chapters. Programming exercises are now assigned stars to indicate their difficulty. One star is the easiest, four stars indicate the most difficult level.
- Tutorial videos by the author are available on the Companion Web site (www.pearsonhighered.com/irvine) to explain worked-out programming exercises.
- The order of chapters in the second half of the book has been revised to form a more logical sequence of topics, and selected chapters are supplied in electronic form for easy searching.

This book is still focused on its primary goal, 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 electronic 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 Companion 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 10.0, running in Microsoft Visual Studio 2010. In addition, batch files are supplied that permit students to assemble and run applications from the Windows command prompt. The 32-bit C++ applications in Chapter 14 were tested with Microsoft Visual C++ .NET.

Web Site Information Updates and corrections to this book may be found at the Companion Web site, including additional programming projects for instructors to assign at the ends of chapters.

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 and AMD 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 high-level programming language such as Python, 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 The Companion Web site contains supplemental learning materials, study guides, and all the source code from the book's examples. 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.

Java Bytecodes and the Java Virtual Machine In Chapters 8 and 9, the author explains the basic operation of Java bytecodes with short illustrative examples. Numerous short examples are shown in disassembled bytecode format, followed by detailed step-by-step explanations.

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 essential 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.

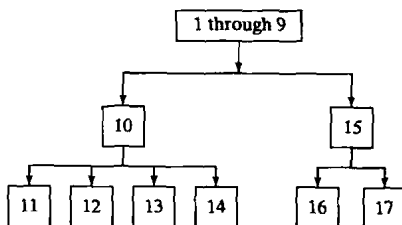
Instructional Aids All the program listings are available on the Web. Instructors are provided a test bank, answers to review questions, solutions to programming exercises, and a Microsoft PowerPoint slide presentation for each chapter.

VideoNotes VideoNotes are Pearson's new visual tool designed to teach students key programming concepts and techniques. These short step-by-step videos demonstrate how to solve problems from design through coding. VideoNotes allow for self-paced instruction with easy navigation including the ability to select, play, rewind, fast-forward, and stop within each VideoNote exercise. A note appears within the text to designate that a VideoNote is available.

VideoNotes are free with the purchase of a new textbook. To *purchase* access to VideoNotes, go to www.pearsonhighered.com/irvine and click on the VideoNotes under *Student Resources*.

Chapter Descriptions

Chapters 1 to 8 contain core concepts 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.



1. **Basic Concepts:** Applications of assembly language, basic concepts, machine language, and data representation.
2. **x86 Processor Architecture:** Basic microcomputer design, instruction execution cycle, x86 processor architecture, x86 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. **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.

13. High-Level Language Interface: Parameter passing conventions, inline assembly code, and linking assembly language modules to C and C++ programs.

14. 16-Bit MS-DOS Programming: Calling MS-DOS interrupts for console and file input-output.

- **Appendix A:** MASM Reference
- **Appendix B:** The x86 Instruction Set
- **Appendix C:** Answers to Review Questions

The following chapters and appendices are supplied online at the Companion Web site:

15. Disk Fundamentals: Disk storage systems, sectors, clusters, directories, file allocation tables, handling MS-DOS error codes, and drive and directory manipulation.

16. BIOS-Level Programming: Keyboard input, video text, graphics, and mouse programming.

17. Expert MS-DOS Programming: Custom-designed segments, runtime program structure, and Interrupt handling. Hardware control using I/O ports.

- **Appendix D:** BIOS and MS-DOS Interrupts
- **Appendix E:** Answers to Review Questions (Chapters 15–17)

Instructor and Student Resources

Instructor Resource Materials

The following protected instructor material is available on the Companion Web site:

www.pearsonhighered.com/irvine

For username and password information, please contact your Pearson Representative.

- Lecture PowerPoint Slides
- Instructor Solutions Manual

Student Resource Materials

The student resource materials can be accessed through the publisher's Web site located at www.pearsonhighered.com/irvine. These resources include:

- VideoNotes
- Online Chapters and Appendices
 - Chapter 15: *Disk Fundamentals*
 - Chapter 16: *BIOS-Level Programming*
 - Chapter 17: *Expert MS-DOS Programming*
 - Appendix D: *BIOS and MS-DOS Interrupts*
 - Appendix E: *Answers to Review Questions (Chapters 15–17)*

Students must use the access card located in the front of the book to register and access the online chapters and VideoNotes. If there is no access card in the front of this textbook, students can purchase access by going to www.pearsonhighered.com/irvine and selecting “purchase access to premium content.” Instructors must also register on the site to access this material. Students will also find a link to the author's Web site. An access card is not required for the following materials:

- *Getting Started*, a comprehensive step-by-step tutorial that helps students customize Visual Studio for assembly language programming.
- Supplementary articles on assembly language programming topics.

- Complete source code for all example programs in the book, as well as the source code for the author's supplementary library.
- *Assembly Language Workbook*, an interactive workbook covering number conversions, addressing modes, register usage, debug programming, and floating-point binary numbers. Content pages are HTML documents to allow for customization. Help File in Windows Help Format.
- Debugging Tools: Tutorials on using Microsoft CodeView, MS-DOS Debug, and Microsoft Visual Studio.

Acknowledgments

Many thanks are due to Tracy Dunkelberger, Executive Editor for Computer Science at Pearson Education, who has provided friendly, helpful guidance over the past few years. Maheswari Pon-Saravanan of TexTech International did an excellent job on the book production, along with Jane Bonnell as the production editor at Pearson. Many thanks to Scott Disanno, the book's managing editor, and Melinda Haggerty, the assistant editor.

Sixth Edition

Many thanks are due to Professor James Brink of Pacific Lutheran University, Professor David Topham of Ohlone College, and Professor W. A. Barrett of San Jose State University. All have contributed excellent code examples and debugging suggestions to this book. In addition, I give grateful acknowledgment to the reviewers of the Sixth edition:

- Hisham Al-Mubaid, University of Houston, Clearlake
- John-Thones Ameny, York College of CUNY
- John F. Doyle, Indiana University, Southeast
- Nicole Jiao, South Texas Community College
- Remzi Seker, University of Arkansas, Little Rock

Previous Editions

I offer my special thanks to the following individuals who were most helpful during the development of earlier editions of this book:

- William Barrett, San Jose State University
- Scott Blackledge
- James Brink, Pacific Lutheran University
- Gerald Cahill, Antelope Valley College
- John Taylor

About THE AUTHOR

Kip Irvine has written five computer programming textbooks, for Intel Assembly Language, C++, Visual Basic (beginning and advanced), and COBOL. His book *Assembly Language for Intel-Based Computers* has been translated into six languages. His first college degrees (B.M., M.M., and doctorate) were in Music Composition, at University of Hawaii and University of Miami. He began programming computers for music synthesis around 1982 and taught programming at Miami-Dade Community College for 17 years. Kip earned an M.S. degree in Computer Science from the University of Miami, and he has been a full-time member of the faculty in the School of Computing and Information Sciences at Florida International University since 2000.

CONTENTS

Preface xix

1 Basic Concepts 1

1.1 Welcome to Assembly Language 1

- 1.1.1 Good Questions to Ask 2
- 1.1.2 Assembly Language Applications 5
- 1.1.3 Section Review 6

1.2 Virtual Machine Concept 7

- 1.2.1 Section Review 9

1.3 Data Representation 9

- 1.3.1 Binary Integers 9
- 1.3.2 Binary Addition 11
- 1.3.3 Integer Storage Sizes 12
- 1.3.4 Hexadecimal Integers 13
- 1.3.5 Signed Integers 15
- 1.3.6 Character Storage 17
- 1.3.7 Section Review 19

1.4 Boolean Operations 22

- 1.4.1 Truth Tables for Boolean Functions 24
- 1.4.2 Section Review 26

1.5 Chapter Summary 26

1.6 Exercises 27

- 1.6.1 Programming Tasks 27
- 1.6.2 Nonprogramming Tasks 27

2 x86 Processor Architecture 29

2.1 General Concepts 29

- 2.1.1 Basic Microcomputer Design 30
- 2.1.2 Instruction Execution Cycle 31
- 2.1.3 Reading from Memory 33
- 2.1.4 How Programs Run 34
- 2.1.5 Section Review 35

2.2	x86 Architecture Details	36
2.2.1	Modes of Operation	36
2.2.2	Basic Execution Environment	36
2.2.3	Floating-Point Unit	39
2.2.4	Overview of Intel Microprocessors	39
2.2.5	Section Review	42
2.3	x86 Memory Management	43
2.3.1	Real-Address Mode	43
2.3.2	Protected Mode	45
2.3.3	Section Review	47
2.4	Components of a Typical x86 Computer	48
2.4.1	Motherboard	48
2.4.2	Video Output	50
2.4.3	Memory	50
2.4.4	Input-Output Ports and Device Interfaces	50
2.4.5	Section Review	52
2.5	Input-Output System	52
2.5.1	Levels of I/O Access	52
2.5.2	Section Review	55
2.6	Chapter Summary	55
2.7	Chapter Exercises	57
3	Assembly Language Fundamentals	58
3.1	Basic Elements of Assembly Language	58
3.1.1	Integer Constants	59
3.1.2	Integer Expressions	60
3.1.3	Real Number Constants	61
3.1.4	Character Constants	61
3.1.5	String Constants	61
3.1.6	Reserved Words	62
3.1.7	Identifiers	62
3.1.8	Directives	62
3.1.9	Instructions	63
3.1.10	The NOP (No Operation) Instruction	65
3.1.11	Section Review	66
3.2	Example: Adding and Subtracting Integers	66
3.2.1	Alternative Version of AddSub	69
3.2.2	Program Template	70
3.2.3	Section Review	70
3.3	Assembling, Linking, and Running Programs	71
3.3.1	The Assemble-Link-Execute Cycle	71
3.3.2	Section Review	77

3.4 Defining Data 77

- 3.4.1 Intrinsic Data Types 77
- 3.4.2 Data Definition Statement 77
- 3.4.3 Defining BYTE and SBYTE Data 78
- 3.4.4 Defining WORD and SWORD Data 80
- 3.4.5 Defining DWORD and SDWORD Data 81
- 3.4.6 Defining QWORD Data 81
- 3.4.7 Defining Packed Binary Coded Decimal (TBYTE) Data 82
- 3.4.8 Defining Real Number Data 83
- 3.4.9 Little Endian Order 83
- 3.4.10 Adding Variables to the AddSub Program 84
- 3.4.11 Declaring Uninitialized Data 85
- 3.4.12 Section Review 85

3.5 Symbolic Constants 86

- 3.5.1 Equal-Sign Directive 86
- 3.5.2 Calculating the Sizes of Arrays and Strings 87
- 3.5.3 EQU Directive 88
- 3.5.4 TEXTEQU Directive 89
- 3.5.5 Section Review 90

3.6 Real-Address Mode Programming (Optional) 90

- 3.6.1 Basic Changes 90

3.7 Chapter Summary 91

3.8 Programming Exercises 92

4 Data Transfers, Addressing, and Arithmetic 94

4.1 Data Transfer Instructions 94

- 4.1.1 Introduction 94
- 4.1.2 Operand Types 95
- 4.1.3 Direct Memory Operands 96
- 4.1.4 MOV Instruction 96
- 4.1.5 Zero/Sign Extension of Integers 98
- 4.1.6 LAHF and SAHF Instructions 100
- 4.1.7 XCHG Instruction 100
- 4.1.8 Direct-Offset Operands 101
- 4.1.9 Example Program (Moves) 102
- 4.1.10 Section Review 103

4.2 Addition and Subtraction 104

- 4.2.1 INC and DEC Instructions 104
- 4.2.2 ADD Instruction 104
- 4.2.3 SUB Instruction 105
- 4.2.4 NEG Instruction 105

4.2.5	Implementing Arithmetic Expressions	106
4.2.6	Flags Affected by Addition and Subtraction	106
4.2.7	Example Program (AddSub3)	110
4.2.8	Section Review	111
4.3	Data-Related Operators and Directives	112
4.3.1	OFFSET Operator	112
4.3.2	ALIGN Directive	113
4.3.3	PTR Operator	114
4.3.4	TYPE Operator	115
4.3.5	LENGTHOF Operator	115
4.3.6	SIZEOF Operator	116
4.3.7	LABEL Directive	116
4.3.8	Section Review	117
4.4	Indirect Addressing	117
4.4.1	Indirect Operands	118
4.4.2	Arrays	119
4.4.3	Indexed Operands	120
4.4.4	Pointers	121
4.4.5	Section Review	123
4.5	JMP and LOOP Instructions	124
4.5.1	JMP Instruction	124
4.5.2	LOOP Instruction	124
4.5.3	Summing an Integer Array	126
4.5.4	Copying a String	126
4.5.5	Section Review	127
4.6	Chapter Summary	128
4.7	Programming Exercises	129
5	Procedures	132
5.1	Introduction	132
5.2	Linking to an External Library	132
5.2.1	Background Information	133
5.2.2	Section Review	134
5.3	The Book's Link Library	134
5.3.1	Overview	136
5.3.2	Individual Procedure Descriptions	137
5.3.3	Library Test Programs	149
5.3.4	Section Review	157
5.4	Stack Operations	157
5.4.1	Runtime Stack	158
5.4.2	PUSH and POP Instructions	160
5.4.3	Section Review	162