

GLOBAL
EDITION

Assembly Language

for x86 Processors

SEVENTH EDITION



Kip R. Irvine

ALWAYS LEARNING

PEARSON

Vice President and Editorial Director, ECS:
Marcia Horton
Executive Editor: *Tracy Johnson*
Executive Marketing Manager: *Tim Galligan*
Marketing Assistant: *Jon Bryant*
Program Management Team Lead: *Scott Disanno*
Program Manager: *Clare Romeo*
Project Manager: *Greg Dulles*
Head, Learning Asset Acquisition, Global Edition:
Laura Dent
Assistant Acquisitions Editor, Global Edition:
Murchana Borthakur

Assistant Project Editor, Global Edition:
Mrithyunjayan Nilayamgode
Associate Print & Media Editor, Global Edition:
Anuprova Dey Chowdhuri
Senior Manufacturing Controller, Production,
Global Edition: *Trudy Kimber*
Cover Art: © *Michelangelus/Shutterstock*
Cover Designer: *PreMediaGlobal*
Senior Operations Specialist: *Nick Sklitsis*
Operations Specialist: *Linda Sager*
Permissions Project Manager: *Karen Sanatar*

IA-32, Pentium, i486, Intel64, Celeron, and Intel 386 are trademarks of Intel Corporation. Athlon, Phenom, and Opteron are trademarks of Advanced Micro Devices. TASM and Turbo Debugger are trademarks of Borland International. Microsoft Assembler (MASM), Windows Vista, Windows 7, 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. Autocad is a trademark of Autodesk. Java is a trademark of Sun Microsystems. PartitionMagic is a trademark of Symantec. All other trademarks or product names are the property of their respective owners.

Pearson Education Limited

Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:
www.pearsonglobaleditions.com

© Pearson Education Limited 2015

The rights of Kip R. Irvine to be identified as the author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

Authorized adaptation from the United States edition, entitled Assembly Language for x86 Processors, 7th edition, ISBN 978-0-13-376940-1, by Kip R. Irvine, published by Pearson Education © 2015.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

Previously published as Assembly Language for Intel-Based Computers.

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.

ISBN 10: 1-292-06121-9

ISBN 13: 978-1-292-06121-4 (Print)

ISBN 13: 978-1-292-06655-4 (PDF)

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

14 13 12 11 10

Typeset in Times by Pavithra Jayapaul, Jouve

Printed and bound by Courier/Westford in The United States of America

Assembly Language for x86 Processors, Global Edition

Table of Contents

Cover

Contents

Preface

About the Author

Chapter 1: Basic Concepts

1.1 Welcome to Assembly Language

1.1.1 Questions You Might Ask

1.1.2 Assembly Language Applications

1.1.3 Section Review

1.2 Virtual Machine Concept

1.2.1 Section Review

1.3 Data Representation

1.3.1 Binary Integers

1.3.2 Binary Addition

1.3.3 Integer Storage Sizes

1.3.4 Hexadecimal Integers

1.3.5 Hexadecimal Addition

1.3.6 Signed Binary Integers

1.3.7 Binary Subtraction

1.3.8 Character Storage

1.3.9 Section Review

1.4 Boolean Expressions

1.4.1 Truth Tables for Boolean Functions

1.4.2 Section Review

1.5 Chapter Summary

1.6 Key Terms

1.7 Review Questions and Exercises

1.7.1 Short Answer

1.7.2 Algorithm Workbench

Chapter 2: x86 Processor Architecture

2.1 General Concepts

2.1.1 Basic Microcomputer Design

2.1.2 Instruction Execution Cycle

2.1.3 Reading from Memory

2.1.4 Loading and Executing a Program

2.1.5 Section Review

Table of Contents

2.2 32-Bit x86 Processors

- 2.2.1 Modes of Operation
- 2.2.2 Basic Execution Environment
- 2.2.3 x86 Memory Management
- 2.2.4 Section Review

2.3 64-Bit x86-64 Processors

- 2.3.1 64-Bit Operation Modes
- 2.3.2 Basic 64-Bit Execution Environment

2.4 Components of a Typical x86 Computer

- 2.4.1 Motherboard
- 2.4.2 Memory
- 2.4.3 Section Review

2.5 InputOutput System

- 2.5.1 Levels of I/O Access
- 2.5.2 Section Review

2.6 Chapter Summary

2.7 Key Terms

2.8 Review Questions

Chapter 3: Assembly Language Fundamentals

3.1 Basic Language Elements

- 3.1.1 First Assembly Language Program
- 3.1.2 Integer Literals
- 3.1.3 Constant Integer Expressions
- 3.1.4 Real Number Literals
- 3.1.5 Character Literals
- 3.1.6 String Literals
- 3.1.7 Reserved Words
- 3.1.8 Identifiers
- 3.1.9 Directives
- 3.1.10 Instructions
- 3.1.11 Section Review

3.2 Example: Adding and Subtracting Integers

- 3.2.1 The AddTwo Program
- 3.2.2 Running and Debugging the AddTwo Program
- 3.2.3 Program Template
- 3.2.4 Section Review

3.3 Assembling, Linking, and Running Programs

- 3.3.1 The Assemble-Link-Execute Cycle
- 3.3.2 Listing File
- 3.3.3 Section Review

Table of Contents

3.4 Defining Data

- 3.4.1 Intrinsic Data Types
- 3.4.2 Data Definition Statement
- 3.4.3 Adding a Variable to the AddTwo Program
- 3.4.4 Defining BYTE and SBYTE Data
- 3.4.5 Defining WORD and SWORD Data
- 3.4.6 Defining DWORD and SDWORD Data
- 3.4.7 Defining QWORD Data
- 3.4.8 Defining Packed BCD (TBYTE) Data
- 3.4.9 Defining Floating-Point Types
- 3.4.10 A Program That Adds Variables
- 3.4.11 Little-Endian Order
- 3.4.12 Declaring Uninitialized Data
- 3.4.13 Section Review

3.5 Symbolic Constants

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

3.6 64-Bit Programming

3.7 Chapter Summary

3.8 Key Terms

- 3.8.1 Terms
- 3.8.2 Instructions, Operators, and Directives

3.9 Review Questions and Exercises

- 3.9.1 Short Answer
- 3.9.2 Algorithm Workbench

3.10 Programming Exercises

Chapter 4: Data Transfers, Addressing, and Arithmetic

4.1 Data Transfer Instructions

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

Table of Contents

4.2 Addition and Subtraction

- 4.2.1 INC and DEC Instructions
- 4.2.2 ADD Instruction
- 4.2.3 SUB Instruction
- 4.2.4 NEG Instruction
- 4.2.5 Implementing Arithmetic Expressions
- 4.2.6 Flags Affected by Addition and Subtraction
- 4.2.7 Example Program
- 4.2.8 Section Review

4.3 Data-Related Operators and Directives

- 4.3.1 OFFSET Operator
- 4.3.2 ALIGN Directive
- 4.3.3 PTR Operator
- 4.3.4 TYPE Operator
- 4.3.5 LENGTHOF Operator
- 4.3.6 SIZEOF Operator
- 4.3.7 LABEL Directive
- 4.3.8 Section Review

4.4 Indirect Addressing

- 4.4.1 Indirect Operands
- 4.4.2 Arrays
- 4.4.3 Indexed Operands
- 4.4.4 Pointers
- 4.4.5 Section Review

4.5 JMP and LOOP Instructions

- 4.5.1 JMP Instruction
- 4.5.2 LOOP Instruction
- 4.5.3 Displaying an Array in the Visual Studio Debugger
- 4.5.4 Summing an Integer Array
- 4.5.5 Copying a String
- 4.5.6 Section Review

4.6 64-Bit Programming

- 4.6.1 MOV Instruction
- 4.6.2 64-Bit Version of SumArray
- 4.6.3 Addition and Subtraction
- 4.6.4 Section Review

4.7 Chapter Summary

4.8 Key Terms

- 4.8.1 Terms
- 4.8.2 Instructions, Operators, and Directives

4.9 Review Questions and Exercises

Table of Contents

4.9.1 Short Answer

4.9.2 Algorithm Workbench

4.10 Programming Exercises

Chapter 5: Procedures

5.1 Stack Operations

5.1.1 Runtime Stack (32-Bit Mode)

5.1.2 PUSH and POP Instructions

5.1.3 Section Review

5.2 Defining and Using Procedures

5.2.1 PROC Directive

5.2.2 CALL and RET Instructions

5.2.3 Nested Procedure Calls

5.2.4 Passing Register Arguments to Procedures

5.2.5 Example: Summing an Integer Array

5.2.6 Saving and Restoring Registers

5.2.7 Section Review

5.3 Linking to an External Library

5.3.1 Background Information

5.3.2 Section Review

5.4 The Irvine32 Library

5.4.1 Motivation for Creating the Library

5.4.2 Overview

5.4.3 Individual Procedure Descriptions

5.4.4 Library Test Programs

5.4.5 Section Review

5.5 64-Bit Assembly Programming

5.5.1 The Irvine64 Library

5.5.2 Calling 64-Bit Subroutines

5.5.3 The x64 Calling Convention

5.5.4 Sample Program that Calls a Procedure

5.6 Chapter Summary

5.7 Key Terms

5.7.1 Terms

5.7.2 Instructions, Operators, and Directives

5.8 Review Questions and Exercises

5.8.1 Short Answer

5.8.2 Algorithm Workbench

5.9 Programming Exercises

Chapter 6: Conditional Processing

6.1 Conditional Branching

Table of Contents

6.2 Boolean and Comparison Instructions

- 6.2.1 The CPU Status Flags
- 6.2.2 AND Instruction
- 6.2.3 OR Instruction
- 6.2.4 Bit-Mapped Sets
- 6.2.5 XOR Instruction
- 6.2.6 NOT Instruction
- 6.2.7 TEST Instruction
- 6.2.8 CMP Instruction
- 6.2.9 Setting and Clearing Individual CPU Flags
- 6.2.10 Boolean Instructions in 64-Bit Mode
- 6.2.11 Section Review

6.3 Conditional Jumps

- 6.3.1 Conditional Structures
- 6.3.2 Jcond Instruction
- 6.3.3 Types of Conditional Jump Instructions
- 6.3.4 Conditional Jump Applications
- 6.3.5 Section Review

6.4 Conditional Loop Instructions

- 6.4.1 LOOPZ and LOOPE Instructions
- 6.4.2 LOOPNZ and LOOPNE Instructions
- 6.4.3 Section Review

6.5 Conditional Structures

- 6.5.1 Block-Structured IF Statements
- 6.5.2 Compound Expressions
- 6.5.3 WHILE Loops
- 6.5.4 Table-Driven Selection
- 6.5.5 Section Review

6.6 Application: Finite-State Machines

- 6.6.1 Validating an Input String
- 6.6.2 Validating a Signed Integer
- 6.6.3 Section Review

6.7 Conditional Control Flow Directives

- 6.7.1 Creating IF Statements
- 6.7.2 Signed and Unsigned Comparisons
- 6.7.3 Compound Expressions
- 6.7.4 Creating Loops with .REPEAT and .WHILE

6.8 Chapter Summary

6.9 Key Terms

- 6.9.1 Terms
- 6.9.2 Instructions, Operators, and Directives

Table of Contents

6.10 Review Questions and Exercises

6.10.1 Short Answer

6.10.2 Algorithm Workbench

6.11 Programming Exercises

6.11.1 Suggestions for Testing Your Code

6.11.2 Exercise Descriptions

Chapter 7: Integer Arithmetic

7.1 Shift and Rotate Instructions

7.1.1 Logical Shifts and Arithmetic Shifts

7.1.2 SHL Instruction

7.1.3 SHR Instruction

7.1.4 SAL and SAR Instructions

7.1.5 ROL Instruction

7.1.6 ROR Instruction

7.1.7 RCL and RCR Instructions

7.1.8 Signed Overflow

7.1.9 SHLD/SHRD Instructions

7.1.10 Section Review

7.2 Shift and Rotate Applications

7.2.1 Shifting Multiple Doublewords

7.2.2 Binary Multiplication

7.2.3 Displaying Binary Bits

7.2.4 Extracting File Date Fields

7.2.5 Section Review

7.3 Multiplication and Division Instructions

7.3.1 MUL Instruction

7.3.2 IMUL Instruction

7.3.3 Measuring Program Execution Times

7.3.4 DIV Instruction

7.3.5 Signed Integer Division

7.3.6 Implementing Arithmetic Expressions

7.3.7 Section Review

7.4 Extended Addition and Subtraction

7.4.1 ADC Instruction

7.4.2 Extended Addition Example

7.4.3 SBB Instruction

7.4.4 Section Review

7.5 ASCII and Unpacked Decimal Arithmetic

7.5.1 AAA Instruction

7.5.2 AAS Instruction

7.5.3 AAM Instruction

Table of Contents

7.5.4 AAD Instruction

7.5.5 Section Review

7.6 Packed Decimal Arithmetic

7.6.1 DAA Instruction

7.6.2 DAS Instruction

7.6.3 Section Review

7.7 Chapter Summary

7.8 Key Terms

7.8.1 Terms

7.8.2 Instructions, Operators, and Directives

7.9 Review Questions and Exercises

7.9.1 Short Answer

7.9.2 Algorithm Workbench

7.10 Programming Exercises

Chapter 8: Advanced Procedures

8.1 Introduction

8.2 Stack Frames

8.2.1 Stack Parameters

8.2.2 Disadvantages of Register Parameters

8.2.3 Accessing Stack Parameters

8.2.4 32-Bit Calling Conventions

8.2.5 Local Variables

8.2.6 Reference Parameters

8.2.7 LEA Instruction

8.2.8 ENTER and LEAVE Instructions

8.2.9 LOCAL Directive

8.2.10 The Microsoft x64 Calling Convention

8.2.11 Section Review

8.3 Recursion

8.3.1 Recursively Calculating a Sum

8.3.2 Calculating a Factorial

8.3.3 Section Review

8.4 INVOKE, ADDR, PROC, and PROTO

8.4.1 INVOKE Directive

8.4.2 ADDR Operator

8.4.3 PROC Directive

8.4.4 PROTO Directive

8.4.5 Parameter Classifications

8.4.6 Example: Exchanging Two Integers

8.4.7 Debugging Tips

8.4.8 WriteStackFrame Procedure

Table of Contents

8.4.9 Section Review

8.5 Creating Multimodule Programs

8.5.1 Hiding and Exporting Procedure Names

8.5.2 Calling External Procedures

8.5.3 Using Variables and Symbols across Module Boundaries

8.5.4 Example: ArraySum Program

8.5.5 Creating the Modules Using Extern

8.5.6 Creating the Modules Using INVOKE and PROTO

8.5.7 Section Review

8.6 Advanced Use of Parameters (Optional Topic)

8.6.1 Stack Affected by the USES Operator

8.6.2 Passing 8-Bit and 16-Bit Arguments on the Stack

8.6.3 Passing 64-Bit Arguments

8.6.4 Non-Doubleword Local Variables

8.7 Java Bytecodes (Optional Topic)

8.7.1 Java Virtual Machine

8.7.2 Instruction Set

8.7.3 Java Disassembly Examples

8.7.4 Example: Conditional Branch

8.8 Chapter Summary

8.9 Key Terms

8.9.1 Terms

8.9.2 Instructions, Operators, and Directives

8.10 Review Questions and Exercises

8.10.1 Short Answer

8.10.2 Algorithm Workbench

8.11 Programming Exercises

Chapter 9: Strings and Arrays

9.1 Introduction

9.2 String Primitive Instructions

9.2.1 MOVSB, MOVSW, and MOVSD

9.2.2 CMPSB, CMPSW, and CMPSD

9.2.3 SCASB, SCASW, and SCASD

9.2.4 STOSB, STOSW, and STOSD

9.2.5 LODSB, LODSW, and LODSD

9.2.6 Section Review

9.3 Selected String Procedures

9.3.1 Str_compare Procedure

9.3.2 Str_length Procedure

9.3.3 Str_copy Procedure

Table of Contents

- 9.3.4 Str_trim Procedure
- 9.3.5 Str_ucase Procedure
- 9.3.6 String Library Demo Program
- 9.3.7 String Procedures in the Irvine64 Library
- 9.3.8 Section Review

9.4 Two-Dimensional Arrays

- 9.4.1 Ordering of Rows and Columns
- 9.4.2 Base-Index Operands
- 9.4.3 Base-Index-Displacement Operands
- 9.4.4 Base-Index Operands in 64-Bit Mode
- 9.4.5 Section Review

9.5 Searching and Sorting Integer Arrays

- 9.5.1 Bubble Sort
- 9.5.2 Binary Search
- 9.5.3 Section Review

9.6 Java Bytecodes: String Processing (Optional Topic)

9.7 Chapter Summary

9.8 Key Terms and Instructions

9.9 Review Questions and Exercises

- 9.9.1 Short Answer
- 9.9.2 Algorithm Workbench

9.10 Programming Exercises

Chapter 10: Structures and Macros

10.1 Structures

- 10.1.1 Defining Structures
- 10.1.2 Declaring Structure Variables
- 10.1.3 Referencing Structure Variables
- 10.1.4 Example: Displaying the System Time
- 10.1.5 Structures Containing Structures
- 10.1.6 Example: Drunkards Walk
- 10.1.7 Declaring and Using Unions
- 10.1.8 Section Review

10.2 Macros

- 10.2.1 Overview
- 10.2.2 Defining Macros
- 10.2.3 Invoking Macros
- 10.2.4 Additional Macro Features
- 10.2.5 Using the Book's Macro Library (32-Bit Mode Only)
- 10.2.6 Example Program: Wrappers
- 10.2.7 Section Review

10.3 Conditional-Assembly Directives

Table of Contents

- 10.3.1 Checking for Missing Arguments
- 10.3.2 Default Argument Initializers
- 10.3.3 Boolean Expressions
- 10.3.4 IF, ELSE, and ENDIF Directives
- 10.3.5 The IFIDN and IFIDNI Directives
- 10.3.6 Example: Summing a Matrix Row
- 10.3.7 Special Operators
- 10.3.8 Macro Functions
- 10.3.9 Section Review

10.4 Defining Repeat Blocks

- 10.4.1 WHILE Directive
- 10.4.2 REPEAT Directive
- 10.4.3 FOR Directive
- 10.4.4 FORC Directive
- 10.4.5 Example: Linked List
- 10.4.6 Section Review

10.5 Chapter Summary

10.6 Key Terms

- 10.6.1 Terms
- 10.6.2 Operators and Directives

10.7 Review Questions and Exercises

- 10.7.1 Short Answer
- 10.7.2 Algorithm Workbench

10.8 Programming Exercises

Chapter 11: MS-Windows Programming

11.1 Win32 Console Programming

- 11.1.1 Background Information
- 11.1.2 Win32 Console Functions
- 11.1.3 Displaying a Message Box
- 11.1.4 Console Input
- 11.1.5 Console Output
- 11.1.6 Reading and Writing Files
- 11.1.7 File I/O in the Irvine32 Library
- 11.1.8 Testing the File I/O Procedures
- 11.1.9 Console Window Manipulation
- 11.1.10 Controlling the Cursor
- 11.1.11 Controlling the Text Color
- 11.1.12 Time and Date Functions
- 11.1.13 Using the 64-Bit Windows API
- 11.1.14 Section Review

11.2 Writing a Graphical Windows Application

Table of Contents

- 11.2.1 Necessary Structures
- 11.2.2 The MessageBox Function
- 11.2.3 The WinMain Procedure
- 11.2.4 The WinProc Procedure
- 11.2.5 The ErrorHandler Procedure
- 11.2.6 Program Listing
- 11.2.7 Section Review

11.3 Dynamic Memory Allocation

- 11.3.1 HeapTest Programs
- 11.3.2 Section Review

11.4 x86 Memory Management

- 11.4.1 Linear Addresses
- 11.4.2 Page Translation
- 11.4.3 Section Review

11.5 Chapter Summary

11.6 Key Terms

11.7 Review Questions and Exercises

- 11.7.1 Short Answer
- 11.7.2 Algorithm Workbench

11.8 Programming Exercises

Chapter 12: Floating-Point Processing and Instruction Encoding

12.1 Floating-Point Binary Representation

- 12.1.1 IEEE Binary Floating-Point Representation
- 12.1.2 The Exponent
- 12.1.3 Normalized Binary Floating-Point Numbers
- 12.1.4 Creating the IEEE Representation
- 12.1.5 Converting Decimal Fractions to Binary Reals
- 12.1.6 Section Review

12.2 Floating-Point Unit

- 12.2.1 FPU Register Stack
- 12.2.2 Rounding
- 12.2.3 Floating-Point Exceptions
- 12.2.4 Floating-Point Instruction Set
- 12.2.5 Arithmetic Instructions
- 12.2.6 Comparing Floating-Point Values
- 12.2.7 Reading and Writing Floating-Point Values
- 12.2.8 Exception Synchronization
- 12.2.9 Code Examples
- 12.2.10 Mixed-Mode Arithmetic
- 12.2.11 Masking and Unmasking Exceptions
- 12.2.12 Section Review

Table of Contents

12.3 x86 Instruction Encoding

- 12.3.1 Instruction Format
- 12.3.2 Single-Byte Instructions
- 12.3.3 Move Immediate to Register
- 12.3.4 Register-Mode Instructions
- 12.3.5 Processor Operand-Size Prefix
- 12.3.6 Memory-Mode Instructions
- 12.3.7 Section Review

12.4 Chapter Summary

12.5 Key Terms

12.6 Review Questions and Exercises

- 12.6.1 Short Answer
- 12.6.2 Algorithm Workbench

12.7 Programming Exercises

Chapter 13: High-Level Language Interface

13.1 Introduction

- 13.1.1 General Conventions
- 13.1.2 .MODEL Directive
- 13.1.3 Examining Compiler-Generated Code
- 13.1.4 Section Review

13.2 Inline Assembly Code

- 13.2.1 __asm Directive in Visual C++
- 13.2.2 File Encryption Example
- 13.2.3 Section Review

13.3 Linking 32-Bit Assembly Language Code to C/C++

- 13.3.1 IndexOf Example
- 13.3.2 Calling C and C++ Functions
- 13.3.3 Multiplication Table Example
- 13.3.4 Calling C Library Functions
- 13.3.5 Directory Listing Program
- 13.3.6 Section Review

13.4 Chapter Summary

13.5 Key Terms

13.6 Review Questions

13.7 Programming Exercises

Appendix A: MASM Reference

A.1 Introduction

A.2 MASM Reserved Words

A.3 Register Names

Table of Contents

A.4 Microsoft Assembler (ML)

A.5 Microsoft Assembler Directives

Appendix B: The x86 Instruction Set

B.1 Introduction

B.1.1 Flags (EFlags)

B.1.2 Instruction Descriptions and Formats

B.2 Instruction Set Details(Non Floating-Point)

B.3 Floating-Point Instructions

Appendix C: Answers to Section Review Questions

1 Basic Concepts

1.1 Welcome to Assembly Language

1.2 Virtual Machine Concept

1.3 Data Representation

1.4 Boolean Expressions

2 x86 Processor Architecture Details

2.1 General Concepts

2.2 x86 Architecture Details

2.3 64-Bit x86-64 Processors

2.4 Components of a Typical x86 Computer

2.5 InputOutput System

3 Assembly Language Fundamentals

3.1 Basic Language Elements

3.2 Example: Adding and Subtracting Integers

3.3 Assembling, Linking, and Running Programs

3.4 Defining Data

3.5 Symbolic Constants

3.6 64-Bit Programming

4 Data Transfers, Addressing, and Arithmetic

4.1 Data Transfer Instructions

4.2 Addition and Subtraction

4.3 Data-Related Operators and Directives

4.4 Indirect Addressing

4.5 JMP and LOOP Instructions

4.6 64-Bit Programming

5 Procedures

5.1 Stack Operations

5.2 Defining and Using Procedures

5.3 Linking to an External Library

5.4 The Irvine32 Library

5.5 64-Bit Assembly Programming

Table of Contents

6 Conditional Processing

- 6.1 Conditional Branching
- 6.2 Boolean and Comparison Instructions
- 6.3 Conditional Loops
- 6.4 Conditional Loop Instructions
- 6.5 Conditional Structures
- 6.6 Application: Finite-State Machines
- 6.7 Conditional Control Flow Directives

7 Integer Arithmetic

- 7.1 Shift and Rotate Instructions
- 7.2 Shift and Rotate Applications
- 7.3 Multiplication and Division Instructions
- 7.4 Extended Addition and Subtraction
- 7.5 ASCII and Unpacked Decimal Arithmetic
- 7.6 Packed Decimal Arithmetic

8 Advanced Procedures

- 8.1 Introduction
- 8.2 Stack Frames
- 8.3 Recursion
- 8.4 INVOKE, ADDR, PROC, and PROTO
- 8.5 Creating Multimodule Programs
- 8.6 Advanced Use of Parameters
- 8.7 Java Bytecodes

9 Strings and Arrays

- 9.1 Introduction
- 9.2 String Primitive Instructions
- 9.3 Selected String Procedures
- 9.4 Two-Dimensional Arrays
- 9.5 Searching and Sorting Integer Arrays
- 9.6 Java Bytecodes: String Processing (Optional Topic)

10 Structures and Macros

- 10.1 Structures
- 10.2 Macros
- 10.3 Conditional-Assembly Directives
- 10.4 Defining Repeat Blocks

11 MS-Windows Programming

- 11.1 Win32 Console Programming
- 11.2 Writing a Graphical Windows Application
- 11.3 Dynamic Memory Allocation
- 11.4 x86 Memory Management

12 Floating-Point Processing and Instruction Encoding

Table of Contents

12.1 Floating-Point Binary Representation

12.2 Floating-Point Unit

12.3 x86 Instruction Encoding

13 High-Level Language Interface

13.1 Introduction

13.2 Inline Assembly Code

13.3 Linking 32-Bit Assembly Language Code to C/C++

Index