1. Record Nr. UNINA9910154777003321

Autore	Irvine Kip R. <1951->
Titolo	Assembly language for x86 processors / / Kip R. Irvine
Pubbl/distr/stampa	Boston : , : Pearson, , [2015] ©2015
ISBN	1-292-06655-5
Edizione	[Seventh, Global edition.]
Descrizione fisica	1 online resource (177 pages) : illustrations, charts, tables
Collana	Always Learning
Disciplina	005.265
Soggetti	IBM microcomputers - Programming
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Includes index.
Nota di contenuto	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 -- 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 Input-Output 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 -- 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 -- 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 -- 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 -- 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 -- 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 -- 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.

Sommario/riassunto

Assembly Language for x86 Processors, 7e is suitable for undergraduate courses in assembly language programming and introductory courses in computer systems and computer architecture. Proficiency in one other programming language, preferably Java, C, or C++, is recommended. Written specifically for 32- and 64-bit Intel/Windows platform, this complete and fully updated study of assembly language teaches students to write and debug programs at the machine level. This text simplifies and demystifies concepts that students need to grasp before they can go on to more advanced computer architecture and operating systems courses. Students put theory into practice through writing software at the machine level, creating a memorable experience that gives them the confidence to work in any OS/machine-oriented environment. Teaching and Learning Experience This program presents a better teaching and learning experience-for you and your students. It will help: Teach Effective Design Techniques: Top-down program design demonstration and explanation allows students to apply techniques to multiple programming courses. Put Theory into Practice: Students will write software at the machine level, preparing them to work in any OS/machine-oriented environment. Tailor the Text to Fit your Course: Instructors can cover optional chapter topics in varying order and depth. Support Instructors and Students: Visit the author's web site http://asmirvine.com/ for chapter objectives, debugging tools, supplemental files, a Getting Started with MASM and Visual Studio 2012 tutorial, and more.