Assembly Language For X86 Processors 6th Edition

What is this book about? Tomcat is an open source web server that processes JavaServer Pages and Java Servlets. It can run as a stand-alone server or be integrated with Apache. Like Apache, the core Tomcat program is relatively simple, but there are many enhancements that can be added to make it much more complex. What does this book cover? You will learn how to integrate Tomcat with the Apache HTTP server (and the situations when you should and you shouldn't), how to set up database connectivity through JDBC, and how to ensure your web applications are secure. This book will provide you, the server administrator, with the necessary knowledge to install and configure Tomcat, as well as many of the most popular enhancements to the Tomcat package. It will help you plan the installation and possible growth options of your site. Here are just a few of the things you'll find covered in this book: Tomcat 3.x, Tomcat 4.0.x, and Tomcat 4.1.x The Tomcat architecture Tomcat installation and configuration Apache and Tomcat integration using the AJP and WARP connectors Tomcat security with SSL, realms, and the Java Security Manager Shared Tomcat hosting, server load testing, and load balancing Managing and administering web applications JDBC with Tomcat Ant and Log4j It will also cover some of the tools that can be integrated with Tomcat, such as Ant (for automatically building web applications) and Log4J (for advanced logging). Who is this book for? This book is for professionals working with Java web applications. It assumes a certain knowledge of the JSP and Servlet technologies, but only from an administrator's point of view. Knowledge of databases, XML, HTML, networking, and general administrative techniques is also assumed.
Information Technology: An Introduction for Today’s Digital World introduces undergraduate students to a wide variety of concepts they will encounter throughout their IT studies and careers. The book covers computer organization and hardware, Windows and Linux operating systems, system administration duties, scripting, computer networks, regular expressions, binary numbers, the Bash shell in Linux, DOS, managing processes and services, and computer security. It also gives students insight on IT-related careers, such as network and web administration, computer forensics, web development, and software engineering. Suitable for any introductory IT course, this classroom-tested text presents many of the topics recommended by the ACM Special Interest Group on IT Education (SIGITE). It offers a far more detailed examination of the computer than current computer literacy texts, focusing on concepts essential to all IT professionals—from operating systems and hardware to information security and computer ethics. The book highlights Windows/DOS and Linux with numerous examples of issuing commands and controlling the operating systems. It also provides details on hardware, programming, and computer networks. Ancillary Resources The book includes laboratory exercises and some of the figures from the text online. PowerPoint lecture slides, answers to exercises, and a test bank are also available for instructors.
This first introductory book designed to train novice programmers is based on a student course taught by the author, and has been optimized for biology students without previous experience in programming. By interspersing theory chapters with numerous small and large programming exercises, the author quickly shows readers how to do their own programming, and throughout uses anecdotes and real-life examples from the biosciences to 'spice up' the text. This practical book thus teaches essential programming skills for life scientists who want -- or need -- to write their own bioinformatics software tools.
For undergraduate courses in assembly language programming, introductory courses in computer systems, and computer architecture. Teach effective design techniques to help students put theory into practice Written specifically for 32- and 64-bit Intel/Windows platform, Assembly Language for x86 Processors , establishes a complete and fully updated study of assembly language. The text teaches students to write and debug programs at the machine level, using effective design techniques that apply to multiple programming courses through top-down program design demonstration and explanation. This approach 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 to create a memorable experience that gives them the confidence to work in any OS/machine-oriented environment. With the 8th Edition, and for the first time, Assembly Language for x86 Processors moves into the world of interactive electronic textbooks, enabling students to experiment and interact with review questions, code animations, tutorial videos, and multiple-input exercises. The convenient, simple-to-use mobile reading experience extends learning beyond class time. Pearson eText allows educators to easily share their own notes with students so they see the connection between their reading and what they learn in class -- motivating them to keep reading, and keep learning. Portable access lets students study on the go, even offline. And, student usage analytics offer insight into how students use the eText, helping educators tailor their instruction.
The eagerly anticipated new edition of the bestselling introduction to x86 assembly language The long-awaited third edition of this bestselling introduction to assembly language has been completely rewritten to focus on 32-bit protected-mode Linux and the free NASM assembler. Assembly is the fundamental language bridging human ideas and the pure silicon hearts of computers, and popular author Jeff Dunteman retains his distinctive lighthearted style as he presents a step-by-step approach to this difficult technical discipline. He starts at the very beginning, explaining the basic ideas of programmable computing, the binary and hexadecimal number systems, the Intel x86 computer architecture, and the process of software development under Linux. From that foundation he systematically treats the x86 instruction set, memory addressing, procedures, macros, and interface to the C-language code libraries upon which Linux itself is built. Serves as an ideal introduction to x86 computing concepts, as demonstrated by the only language directly understood by the CPU itself Uses an approachable, conversational style that assumes no prior experience in programming of any kind Presents x86 architecture and assembly concepts through a cumulative tutorial approach that is ideal for self-paced instruction Focuses entirely on free, open-source software, including Ubuntu Linux, the NASM assembler, the Kate

editor, and the Gdb/Insight debugger Includes an x86 instruction set reference for the most common machine instructions, specifically tailored for use by programming beginners Woven into the presentation are plenty of assembly code examples, plus practical tips on software design, coding, testing, and debugging, all using free, open-source software that may be downloaded without charge from the Internet.

This book constitutes the refereed proceedings of the Second International Symposium on Engineering Secure Software and Systems, ESSoS 2010, held in Pisa, Italy, in February 2010. The 9 revised full papers presented together with 8 ideas papers were carefully reviewed and selected from 58 submissions. The papers are organized in topical sections on attack analysis and prevention, policy verification and enforcement, and secure system and software development.

Never HIGHLIGHT a Book Again! Virtually all of the testable terms, concepts, persons, places, and events from the textbook are included. Cram101 Just the FACTS101 studyguides give all of the outlines, highlights, notes, and quizzes for your textbook with optional online comprehensive practice tests. Only Cram101 is Textbook Specific. Accompanys: 9780136022121 .

What is Assembly Language?Each personal computer has a microprocessor that manages the computer's arithmetical, logical, and control activities.Each family of processors has its own set of instructions for handling various operations such as getting input from keyboard, displaying information on screen and performing various other jobs. These set of instructions are called 'machine language instructions'.A processor understands only machine language instructions, which are strings of 1's and 0's. However, machine language is too obscure and complex for using in software development. So, the low-level assembly language is designed for a specific family of processors that represents various instructions in symbolic code and a more understandable form.Advantages of Assembly LanguageHaving an understanding of assembly language makes one aware of ?How programs interface with OS, processor, and BIOS;How data is represented in memory and other external devices;How the processor accesses and executes instruction;How instructions access and process data;How a program accesses external devices.Other advantages of using assembly language are ?It requires less memory and execution time;It allows hardware-specific complex jobs in an easier way;It is suitable for time-critical jobs;It is most suitable for writing interrupt service routines and other memory resident programs.

Master x86 language from the Linux point of view with this one-concept-at-a-time guide. Neveln gives an "under the hood" perspective of how Linux works and shows how to create device drivers. The CD-ROM includes all source code from the book plus edlinas, an x86 simulator that's perfect for hands-on, interactive assembler development.

This text introduces the spirit and theory of hacking as well as the science behind it all; it also provides some core techniques and tricks of hacking so you can think like a hacker, write your own hacks or thwart potential system attacks.

This book is intended for beginners who would like to learn the basics of Assembly Programming. This book uses Simple words, Short sentences, and Straightforward paragraphs. The triple S way to learn Assembly Programming. The topics covered in this book includes a brief introduction to assembly, common arithmetic instructions, character and string input and display routines, flow controls including conditional and looping statements, stack, and procedures. This assembly language book is intended for complete beginners in assembly programming. However, it is assumed that the reader has prior or basic knowledge with other programming languages. This book includes screenshots of step by step of how to code, compile, link, and run assembly programs. This book is packed with working sample assembly programs and after reading this book, the reader would be able to develop assembly programs based particularly on problems given in computer science courses.

Along with the increasingly important runtime engines pervasive in our daily-life computing, there is a strong demand from the software community for a solid presentation on the design and implementation of modern virtual machines, including the Java virtual machine, JavaScript engine and Android execution engine. The community expects to see not only formal algorithm description, but also pragmatic code snippets; to understand not only research topics, but also engineering solutions. This book meets these demands by providing a unique description that combines high level design with low level implementations and academic advanced topics with commercial solutions. This book takes a holistic approach to the design of VM architecture, with contents organized into a consistent framework, introducing topics and algorithms in an easily understood step by step process. It focuses on the critical aspects of VM design, which are often overlooked in other works, such as runtime helpers, stack unwinding and native interface. The algorithms are fully illustrated in figures and implemented in easy to digest code snippets, making the abstract concepts tangible and programmable for system software developers.

The increasing complexity of programming environments provides a number of opportunities for assembly language programmers. 32/64-Bit 80x86 Assembly Language Architecture attempts to break through that complexity by providing a step-by-step understanding of programming Intel and AMD 80x86 processors in assembly language. This book explains 32-bit and 64-bit 80x86 assembly language programming inclusive of the SIMD (single instruction multiple data) instruction supersets that bring the 80x86 processor into the realm of the supercomputer, gives insight into the FPU (floating-point unit) chip in every Pentium processor, and offers strategies for optimizing code.

Annotation The predominant language used in embedded microprocessors, assembly language lets you write programs that are typically faster and more compact than programs written in a high-level language and provide greater control over the program applications. Focusing on the languages used in X86 microprocessors, X86 Assembly Language and C Fundamentals explains how to write programs in the X86 assembly language, the C programming language, and X86 assembly language modules embedded in a C program. A wealth of program design examples, including the complete code and outputs, help you grasp the concepts more easily. Where needed, the book also details the theory behind the design. Learn the X86 Microprocessor Architecture and Commonly Used Instructions Assembly language programming requires knowledge of number representations, as well as the architecture of the computer on which the language is

being used. After covering the binary, octal, decimal, and hexadecimal number systems, the book presents the general architecture of the X86 microprocessor, individual addressing modes, stack operations, procedures, arrays, macros, and input/output operations. It highlights the most commonly used X86 assembly language instructions, including data transfer, branching and looping, logic, shift and rotate, and string instructions, as well as fixed-point, binary-coded decimal (BCD), and floating-point arithmetic instructions. Get a Solid Foundation in a Language Commonly Used in Digital Hardware Written for students in computer science and electrical, computer, and software engineering, the book assumes a basic background in C programming, digital logic design, and computer architecture. Designed as a tutorial, this comprehensive and self-contained text offers a solid foundation in assembly language for anyone working with the design of digital hardware.

Assembly Language for x86 Processors, 7e is intended for use in undergraduate courses in assembly language programming and introductory courses in computer systems and computer architecture. This title is also suitable for embedded systems programmers and engineers, communication specialists, game programmers, and graphics programmers. 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 fullyupdated 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. Additional learning and teaching tools are available on the author's web site at http://asmirvine.com/ where both instructors and students can access chapter objectives, debugging tools, supplemental files, a Getting Started with MASM and Visual Studio 2012 tutorial, and more. 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 studentsto 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.

Many personal computers are based on one of the 86 series of Intel microprocessors, namely the 8086, 80286, 80386, and the 80486, in order of increasing power. Programming the computer in the relevant assembly language allows the user to take full advantage of the speed and power of the microprocessor. This book is written for PC users who already have some familiarity with a high-level language such as Basic, C, or Pascal, and who want the extra facilities and efficiency of assembly language. The book starts at an elementary level with the basics of assembly programming and the properties of the microprocessor in its simplest mode of operation, the real address mode. Instructions for this mode of the 8086 and 80286 are progressively introduced through illustrative programs and subroutines. Further topics discussed are operating system calls and the 80286 protected mode. A separate chapter deals with the additional instructions of the 80386 and its modes of operation. Expanded and extended memory are also covered. The 80x87 coprocessors are treated for the benefit of readers who have one either as a part of the 80486 or as a complement to their 80x86. Numerous exercises are provided throughout the text. These enable readers to test their understanding and to gain experience in assembly programming.

Thought-provoking and accessible in approach, this updated and expanded second edition of the Assembly Language for x86 Processors, 7/e provides a user-friendly introduction to the subject, Taking a clear structural framework, it guides the reader through the subject's core elements. A flowing writing style combines with the use of illustrations and diagrams throughout the text to ensure the reader understands even the most complex of concepts. This succinct and enlightening overview is a required reading for advanced graduate-level students. We hope you find this book useful in shaping your future career. Feel free to send us your enquiries related to our publications to info@risepress.pw Rise Press

Tips for the practical use of debuggers, such as NuMega SoftIce, Microsoft Visual Studio Debugger, and Microsoft Kernel Debugger, with minimum binding to a specific environment are disclosed in this debugger guide. How debuggers operate and how to overcome obstacles and repair debuggers is demonstrated. Programmers will learn how to look at what is inside a computer system, how to reconstruct the operating algorithm of a program distributed without source code, how to modify the program, and how to debug drivers. The use of debugging applications and drivers in Windows and Unix operating systems on Intel Pentium/DEC Alpha-based processors is also detailed.

The predominant language used in embedded microprocessors, assembly language lets you write programs that are typically faster and more compact than programs written in a high-level language and provide greater control over the program applications. Focusing on the languages used in X86 microprocessors, X86 Assembly Language and C Fundamentals explains how to write programs in the X86 assembly language, the C programming language, and X86 assembly language modules embedded in a C program. A wealth of program design examples, including the complete code and outputs, help you grasp the concepts more easily. Where needed, the book also details the theory behind the design. Learn the X86 Microprocessor Architecture and Commonly Used Instructions Assembly language programming requires knowledge of number representations, as well as the architecture of the computer on which the language is being used. After covering the binary, octal, decimal, and hexadecimal number systems, the book presents the general architecture of the X86 microprocessor, individual addressing modes, stack operations, procedures, arrays, macros, and input/output operations. It highlights the most commonly used X86 assembly language instructions, including data transfer, branching and looping, logic, shift and rotate, and string instructions, as well as fixed-point, binary-coded

decimal (BCD), and floating-point arithmetic instructions. Get a Solid Foundation in a Language Commonly Used in Digital Hardware Written for students in computer science and electrical, computer, and software engineering, the book assumes a basic background in C programming, digital logic design, and computer architecture. Designed as a tutorial, this comprehensive and self-contained text offers a solid foundation in assembly language for anyone working with the design of digital hardware.

Learn assembly language programming from an expert's decade-long experience About This Video Use Emu8086 to create assembly programs for the 8086 processor Communicate with a C program using assembly code Understand disassembly In Detail This course will teach you x86 assembly programming by taking you through how processors work and how machine-level coding is possible. The course is divided into two modules. The first module shows you how to use an emulator for the legacy Intel 8086 processor and explains what goes on under the hood. Once you've learned everything you need to know about the legacy 8086 processor and how to program it in the assembly language, you'll progress to the second module, which focuses on writing assembly code for the modern x86 processors. You'll understand how to write 32-bit programs for Windows machines and establish communication between your assembly code and a C program. Prior experience in the C programming language or, at the very least, some programming experience in another language is necessary to follow the concepts covered in the second module of this course.

This book is about programming the Intel(R) X86-X64 in assembly language using the "free" version of Microsoft(R) Visual Studio 17 software. The X86 implies the 16-bit legacy Intel(R) 8086 processor up through the 64-bit Intel(R) core i7 and even beyond.

Describes the techniques of computer hacking, covering such topics as stack-based overflows, format string exploits, and shellcode.

This book covers assembly language programming for the x86 family of microprocessors. The objective is to teach how to program in x86 assembly, as well as the history and basic architecture of x86 processor family. When referring to x86 we address the complete range of x86-based processors but keep in mind that x86-32 Assembly is commonly referred to as IA-32 (Intel Architecture, 32-bit) Assembly, a 32-bit extension of the original Intel x86 processor architecture. IA-32 has full backwards compatibility (16-bit). AMD64 or AMD 64-bit extension is called x86-64 and is backwards compatible with 32-bit code without performance loss. Intel 64 previously named IA-32e or EM64T is almost identical to x86-64. Throughout the book these terms may be used interchangeably when appropriate. A special notice will be given if covering 16-bit, 32-bit or 64-bits architectures and on any limitations so to limit confusion.

A recipepacked reference guide filled with practical tasks that are concisely explained to develop and broaden the user's abilities with the D programming language. If you are an experienced programmer who is looking to explore a language that offers plenty of advantages over more established programming languages, this is the book for you. We assume that you are already familiar with general programming language basics, but you do not need to be a proficient user of D.

This book is an introduction to computer architecture, hardware and software, presented in the context of the Intel x86 family. The x86 describes not only a line of microprocessor chips dating back to 1978, but also an instruction set architecture (ISA) that the chips implement. The chip families were built by Intel and other manufacturers, and execute the same instructions, but in different manners. The results are the same, arithmetically and logically, but may differ in their timing. Why the focus on the Intel x86? It was the basis of the IBM personal computer (PC) family and its spin-offs. It has transitioned from a 16 to a 32 to a 64-bit architecture, keeping compatibility for more than 30 years. It's an de-facto industry standard that has withstood the test of time. This book covers the Intel ISA-16 and ISA-32 architectures from the 8086/8088 to the Pentium, including the math coprocessors. A chart of ISA processors is included. The purpose of this book is to provide the basic background information for an understanding of the 80x86 family, the IBM Personal Computer (pc), and programming in assembly language as an introduction to the broader field of Computer Architecture. It will stress the pervasiveness of this pc-based technology in everyday things and events. It will provide an introduction to Software System Engineering and the Design for Debugging methodology. This book is a spin-off of a course in Computer Architecture/System Integration, taught in the graduate Engineering Science Program at Loyola College (now, Loyola University in Maryland). If we learn to program in the language c, for example, we can take our skills to any computer with a set of c-based tools. If we learn IA-32 assembly language, we have to relearn a language if we switch to a different architecture. So, why do we learn assembly language? Because it gives us insight into the underlying hardware, how it is organized, and how it operates. This book is dedicated to the graduate students in Engineering Science at Loyola College, Columbia Campus, who took the course EG-611, "System Integration I, the x86 Architecture and Assembly Language." The course was given to hundreds of students over a span of 15 years by myself and others. An Extensive bibliography is provided. Table of Contents Introduction Definitions Technological & Economic Impact Limitations of the technology Number Systems Computer Instruction Set Architecture Prefixes Position notation Infinities, overflows, and underflows Hexadecimal numbers Elementary Math operations Base conversion Logical operations on data Math in terms of logic functions Negative numbers Data structures Integers BCD Format ASCII Format Parity Lists Hardware Elements of a Computer The Central Processing Unit The fetch/execute cycle X86 Processor family Input/Output I/O Methods Polled I/O Interrupt DMA Serial versus parallel Memory Memory organization and addressing Caches Memory Management Software Elements of a Computer Instruction Set Architecture (ISA) of the 80x86 Family Programmers model of the x86 Assembly Language The compilation process Operating system: what it is; what it does The Intel x86 instruction set Stack Protocols Basic Math Operations Logical operations BCD Operations 64 Operations on STRINGS of data Shifts/rotates Multiply Divide Faster Math Interrupt architecture Pseudo operations Labels Addressing modes on the 8086 Effective Address Calculation Memory Segments Code addressing modes Data Addressing Modes Program Flow Subroutines Macro Modular design X86 Boot sequence The 8086 reset The BIOS ROM CPUid instruction Load

The purpose of this text is to provide a reference for University level assembly language and systems programming courses. Specifically, this text addresses the x86-64 instruction set for the popular x86-64 class of processors using the Ubuntu 64-bit Operating System (OS). While the provided code and various examples should work under any Linux-based 64-bit OS, they have only been tested under Ubuntu 14.04 LTS (64-bit). The x86-64 is a Complex Instruction Set Computing (CISC) CPU design. This refers to the internal processor design philosophy. CISC processors typically include a wide variety of instructions (sometimes overlapping), varying instructions sizes, and a wide range of addressing modes. The term was retroactively coined in contrast to Reduced Instruction Set Computer (RISC3).

A new assembly language programming book from a well-loved master. Art of 64-bit Assembly Language capitalizes on the long-lived success of Hyde's seminal The Art of Assembly Language. Randall Hyde's The Art of Assembly Language has been the go-to book for learning assembly language for decades. Hyde's latest work, Art of 64-bit Assembly Language is the 64-bit version of this popular text. This book guides you through the maze of assembly language programming by showing how to write assembly code that mimics operations in High-Level Languages. This leverages your HLL knowledge to rapidly understand x86-64 assembly language. This new work uses the Microsoft Macro Assembler (MASM), the most popular x86-64 assembler today. Hyde covers the standard integer set, as well as the x87 FPU, SIMD parallel instructions, SIMD scalar instructions (including high-performance floating-point instructions), and MASM's very powerful

macro facilities. You'll learn in detail: how to implement high-level language data and control structures in assembly language; how to write parallel algorithms using the SIMD (single-instruction, multiple-data) instructions on the x86-64; and how to write stand alone assembly programs and assembly code to link with HLL code. You'll also learn how to optimize certain algorithms in assembly to produce faster code.
Never HIGHLIGHT a Book Again Virtually all testable terms, concepts, persons, places, and events are included. Cram101 Textbook Outlines gives all of the outlines, highlights, notes for your textbook with optional online practice tests. Only Cram101 Outlines are Textbook Specific. Cram101 is NOT the Textbook. Accompanys: 9780521673761
Modern X86 Assembly Language Programming shows the fundamentals of x86 assembly language programming. It focuses on the aspects of the x86 instruction set that are most relevant to application software development. The book's structure and sample code are designed to help the reader quickly understand x86 assembly language programming and the computational capabilities of the x86 platform. Please note: Book appendixes can be downloaded here: http://www.apress.com/9781484200650 Major topics of the book include the following: 32-bit core architecture, data types, internal registers, memory addressing modes, and the basic instruction set X87 core architecture, register stack, special purpose registers, floating-point encodings, and instruction set MMX technology and instruction set Streaming SIMD extensions (SSE) and Advanced Vector Extensions (AVX) including internal registers, packed integer arithmetic, packed and scalar floating-point arithmetic, and associated instruction sets 64-bit core architecture, data types, internal registers, memory addressing modes, and the basic instruction set 64-bit extensions to SSE and AVX technologies X86 assembly language optimization strategies and techniques
Gain the fundamentals of x86 64-bit assembly language programming and focus on the updated aspects of the x86 instruction set that are most relevant to application software development. This book covers topics including x86 64-bit programming and Advanced Vector Extensions (AVX) programming. The focus in this second edition is exclusively on 64-bit base programming architecture and AVX programming. Modern X86 Assembly Language Programming’s structure and sample code are designed to help you quickly understand x86 assembly language programming and the computational capabilities of the x86 platform. After reading and using this book, you’ll be able to code performance-enhancing functions and algorithms using x86 64-bit assembly language and the AVX, AVX2 and AVX-512 instruction set extensions. What You Will Learn Discover details of the x86 64-bit platform including its core architecture, data types, registers, memory addressing modes, and the basic instruction set Use the x86 64-bit instruction set to create performance-enhancing functions that are callable from a high-level language (C++) Employ x86 64-bit assembly language to efficiently manipulate common data types and programming constructs including integers, text strings, arrays, and structures Use the AVX instruction set to perform scalar floating-point arithmetic Exploit the AVX, AVX2, and AVX-512 instruction sets to significantly accelerate the performance of computationally-intense algorithms in problem domains such as image processing, computer graphics, mathematics, and statistics Apply various coding strategies and techniques to optimally exploit the x86 64-bit, AVX, AVX2, and AVX-512 instruction sets for maximum possible performance Who This Book Is For Software developers who want to learn how to write code using x86 64-bit assembly language. It’s also ideal for software developers who already have a basic understanding of x86 32-bit or 64-bit assembly language programming and are interested in learning how to exploit the SIMD capabilities of AVX, AVX2 and AVX-512.
Current multimedia and telecom applications require complex, heterogeneous multiprocessor system on chip (MPSoC) architectures with specific communication infrastructure in order to achieve the required performance. Heterogeneous MPSoC includes different types of processing units (DSP, microcontroller, ASIP) and different communication schemes (fast links, non standard memory organization and access). Programming an MPSoC requires the generation of efficient software running on MPSoC from a high level environment, by using the characteristics of the architecture. This task is known to be tedious and error prone, because it requires a combination of high level programming environments with low level software design. This book gives an overview of concepts related to embedded software design for MPSoC. It details a full software design approach, allowing systematic, high-level mapping of software applications on heterogeneous MPSoC. This approach is based on gradual refinement of hardware/software interfaces and simulation models allowing to validate the software at different abstraction levels. This book combines Simulink for high level programming and SystemC for the low level software development. This approach is illustrated with multiple examples of application software and MPSoC architectures that can be used for deep understanding of software design for MPSoC.
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
Learn the fundamentals of x86 Single instruction multiple data (SIMD) programming using C++ intrinsic functions and x86-64 assembly language. This book emphasizes x86 SIMD programming topics and technologies that are relevant to

modern software development in applications which can exploit data level parallelism, important for the processing of big data, large batches of data and related important in data science and much more. Modern Parallel Programming with C++ and Assembly Language is an instructional text that explains x86 SIMD programming using both C++ and assembly language. The book’s content and organization are designed to help you quickly understand and exploit the SIMD capabilities of x86 processors. It also contains an abundance of source code that is structured to accelerate learning and comprehension of essential SIMD programming concepts and algorithms. After reading this book, you will be able to code performance-optimized AVX, AVX2, and AVX-512 algorithms using either C++ intrinsic functions or x86-64 assembly language. What You Will Learn Understand the essential details about x86 SIMD architectures and instruction sets including AVX, AVX2, and AVX-512. Master x86 SIMD data types, arithmetic instructions, and data management operations using both integer and floating-point operands. Code performance-enhancing functions and algorithms that fully exploit the SIMD capabilities of a modern x86 processor. Employ C++ intrinsic functions and x86-64 assembly language code to carry out arithmetic calculations using common programming constructs including arrays, matrices, and user-defined data structures. Harness the x86 SIMD instruction sets to significantly accelerate the performance of computationally intense algorithms in applications such as machine learning, image processing, computer graphics, statistics, and matrix arithmetic. Apply leading-edge coding strategies and techniques to optimally exploit the x86 SIMD instruction sets for maximum possible performance. Who This Book Is For Intermediate to advanced programmers/developers in general. Readers of this book should have previous programming experience with modern C++ (i.e., ANSI C++11 or later) and Assembly. Some familiarity with Microsoft’s Visual Studio or the GNU toolchain will be helpful. The target audience for Modern X86 SIMD Programming are experienced software developers, programmers and maybe some hobbyists.

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.