

## Low Level Programming C Assembly And Program Execution On

Unlike high-level languages such as Java and C++, assembly language is much closer to the machine code that actually runs computers; it's used to create programs or modules that are very fast and efficient, as well as in hacking exploits and reverse engineering. Covering assembly language in the Pentium microprocessor environment, this code-intensive guide shows programmers how to create stand-alone assembly language programs as well as how to incorporate assembly language libraries or routines into existing high-level applications. Demonstrates how to manipulate data, incorporate advanced functions and libraries, and maximize application performance. Examples use C as a high-level language, Linux as the development environment, and GNU tools for assembling, compiling, linking, and debugging. Functional C teaches how to program in C, assuming that the student has already learnt how to formulate algorithms in a functional style. By using this as a starting point, the student will become a better C programmer, capable of writing programs that are easier to comprehend, maintain and that avoid common errors and pitfalls. All program code that appears in Functional C is available on our ftp server - see below. How to find a code fragment? To access a particular code fragment, use the book to locate the section or subsection in which the code fragment appears, then click on that section in the code index. This will open the appropriate page at the beginning of the section. The code fragment may then be selected using the copy/paste facilities of your browser. Each chapter is represented by a separate page, so as an alternative to the procedure above you can use the save-as menu of your browser to up-load all code fragments in a particular chapter at once. Also available on our ftp server is errata for Functional C.

-Access Real mode from Protected mode; Protected mode from Real mode Apply OOP concepts to assembly language programs Interface assembly language programs with high-level languages Achieve direct hardware manipulation and memory access Explore the architecture The authors begin by explaining why C++ is worth learning and then move on to the most important elements of C++. This book emphasizes understanding and practical use of the language. It explores the basics, covers inheritance and object-oriented programming, discusses templates and the powerful kind of abstraction they provide, and shows how to design and use libraries.

Effective C

For Software Engineers

Automated Data Acquisition and Control Systems

C++ Crash Course

History of Programming Languages

Where C and Assembly Meet

X86 Assembly Language and C Fundamentals

**Assembly is a low-level programming language that's one step above a computer's native machine language. Although assembly language is commonly used for writing device drivers, emulators, and video games, many programmers find its somewhat unfriendly syntax intimidating to learn and use. Since 1996, Randall Hyde's *The Art of Assembly Language* has provided a comprehensive, plain-English, and patient introduction to 32-bit x86 assembly for non-assembly programmers. Hyde's primary teaching tool, *High Level Assembler (or HLA)*, incorporates many of the features found in high-level languages (like C, C++, and Java) to help you quickly grasp basic assembly concepts. HLA lets you write true low-level code while enjoying the benefits of high-level language programming. As you read *The Art of Assembly Language*, you'll learn the low-level theory fundamental to computer science and turn that understanding into real, functional code. You'll learn how to: –Edit, compile, and run HLA programs –Declare and use constants, scalar variables, pointers, arrays, structures, unions, and namespaces –Translate arithmetic expressions (integer and floating point) –Convert high-level control structures This much anticipated second edition of *The Art of Assembly Language* has been updated to reflect recent changes to HLA and to support Linux, Mac OS X, and FreeBSD. Whether you're new to programming or you have experience with high-level languages, *The Art of Assembly Language, 2nd Edition* is your essential guide to learning this complex, low-level language.**

**Incorporate the assembly language routines in your high level language applications About This Book Understand the Assembly programming concepts and the benefits of examining the AL codes generated from high level languages Learn to incorporate the assembly language routines in your high level language applications Understand how a CPU works when programming in high level languages Who This Book Is For This book is for developers who would like to learn about Assembly language. Prior programming knowledge of C and C++ is assumed. What You Will Learn Obtain deeper understanding of the underlying platform Understand binary arithmetic and logic operations Create elegant and efficient code in Assembly language Understand how to link Assembly code to outer world Obtain in-depth understanding of relevant internal mechanisms of Intel CPU Write stable, efficient and elegant patches for running processes In Detail The Assembly language is the lowest level human readable programming language on any platform. Knowing the way things are on the Assembly level will help developers design their code in a much more elegant and efficient way. It may be produced by compiling source code from a high-level programming language (such as C/C++) but can also be written from scratch. Assembly code can be converted to machine code using an assembler. The first section of the book starts with setting up the development environment on Windows and Linux, mentioning most common toolchains. The reader is led through the basic structure of CPU and memory, and is presented the most important Assembly instructions through examples for both Windows and Linux, 32 and 64 bits. Then the reader would understand how high level languages are translated into Assembly and then compiled into object code. Finally we will cover patching existing code, either legacy code without sources or a running code in same or remote process. Style and approach This book takes a step-by-step, detailed approach to *Comprehensively learning Assembly Programming.***

**Today's programmers are often narrowly trained because the industry moves too fast. That's where *Write Great Code, Volume 1: Understanding the Machine* comes in. This, the first of four volumes by author Randall Hyde, teaches important concepts of machine organization in a language-independent fashion, giving programmers what they need to know to write great code in any language, without the usual overhead of learning assembly language to master this topic. A solid foundation in software engineering, *The Write Great Code* series will help programmers make wiser choices with respect to programming statements and data types when writing software.**

**This book teaches computer programming to the complete beginner using the native C language. As such,**

**it assumes you have no knowledge whatsoever about programming. The main goal of this book is to teach fundamental programming principles using C, one of the most widely used programming languages in the world today. We discuss only those features and statements in C that are necessary to achieve our goal. Once you learn the principles well, they can be applied to any language. If you are worried that you are not good at high-school mathematics, don't be. It is a myth that you must be good at mathematics to learn programming. C is considered a 'modern' language even though its roots date back to the 1970s. Originally, C was designed for writing 'systems' programs—things like operating systems, editors, compilers, assemblers and input/output utility programs. But, today, C is used for writing all kinds of applications programs as well—word processing programs, spreadsheet programs, database management programs, accounting programs, games, robots, embedded systems/electronics (i.e., Arduino), educational software—the list is endless. Note: Appendices A-D are available as part of the free source code download at the Apress website. What You Will Learn: How to get started with programming using the C language How to use the basics of C How to program with sequence, selection and repetition logic How to work with characters How to work with functions How to use arrays Who This Book Is For: This book is intended for anyone who is learning programming for the first time.**

**Professional Assembly Language**

**SPARC Architecture, Assembly Language Programming, and C**

**Introduction to 80x86 Assembly Language and Computer Architecture**

**The D Programming Language**

**Introduction to Compilers and Language Design**

**The Art of Assembly Language, 2nd Edition**

*A fast-paced, thorough introduction to modern C++ written for experienced programmers. After reading C++ Crash Course, you'll be proficient in the core language concepts, the C++ Standard Library, and the Boost Libraries. C++ is one of the most widely used languages for real-world software. In the hands of a knowledgeable programmer, C++ can produce small, efficient, and readable code that any programmer would be proud of. Designed for intermediate to advanced programmers, C++ Crash Course cuts through the weeds to get you straight to the core of C++17, the most modern revision of the ISO standard. Part 1 covers the core of the C++ language, where you'll learn about everything from types and functions, to the object life cycle and expressions. Part 2 introduces you to the C++ Standard Library and Boost Libraries, where you'll learn about all of the high-quality, fully-featured facilities available to you. You'll cover special utility classes, data structures, and algorithms, and learn how to manipulate file systems and build high-performance programs that communicate over networks. You'll learn all the major features of modern C++, including:*

- Fundamental types, reference types, and user-defined types
- The object lifecycle including storage duration, memory management, exceptions, call stacks, and the RAII paradigm
- Compile-time polymorphism with templates and run-time polymorphism with virtual classes
- Advanced expressions, statements, and functions
- Smart pointers, data structures, dates and times, numerics, and probability/statistics facilities
- Containers, iterators, strings, and algorithms
- Streams and files, concurrency, networking, and application development

*With well over 500 code samples and nearly 100 exercises, C++ Crash Course is sure to help you build a strong C++ foundation.*

*The official book on the Rust programming language, written by the Rust development team at the Mozilla Foundation, fully updated for Rust 2018. The Rust Programming Language is the official book on Rust: an open source systems programming language that helps you write faster, more reliable software. Rust offers control over low-level details (such as memory usage) in combination with high-level ergonomics, eliminating the hassle traditionally associated with low-level languages. The authors of The Rust Programming Language, members of the Rust Core Team, share their knowledge and experience to show you how to take full advantage of Rust's features--from installation to creating robust and scalable programs. You'll begin with basics like creating functions, choosing data types, and binding variables and then move on to more advanced concepts, such as:*

- Ownership and borrowing, lifetimes, and traits
- Using Rust's memory safety guarantees to build fast, safe programs
- Testing, error handling, and effective refactoring
- Generics, smart pointers, multithreading, trait objects, and advanced pattern matching
- Using Cargo, Rust's built-in package manager, to build, test, and document your code and manage dependencies
- How best to use Rust's advanced compiler with compiler-led programming techniques

*You'll find plenty of code examples throughout the book, as well as three chapters dedicated to building complete projects to test your learning: a number guessing game, a Rust implementation of a command line tool, and a multithreaded server. New to this edition: An extended section on Rust macros, an expanded chapter on modules, and appendixes on Rust development tools and editions.*

*Learn Intel 64 assembly language and architecture, become proficient in C, and understand how the programs are compiled and executed down to machine instructions, enabling you to write robust, high-performance code. Low-Level Programming explains Intel 64 architecture as the result of von Neumann architecture evolution. The book teaches the latest version of the C language (C11) and assembly language from scratch. It covers the entire path from source code to program execution, including generation of ELF object files, and static and dynamic linking. Code examples and exercises are included along with the best code practices. Optimization capabilities and limits of modern compilers are examined, enabling you to balance between program readability and performance. The use of various performance-gain techniques is demonstrated, such as SSE instructions and pre-fetching. Relevant Computer Science topics such as models of computation and formal grammars are addressed, and their practical value explained. What You'll Learn*

*Low-Level Programming teaches programmers to:*

- Freely write in assembly language
- Understand the programming model of Intel 64
- Write maintainable and robust code in C11
- Follow the compilation process and decipher assembly listings
- Debug errors in compiled assembly code
- Use appropriate models of computation to greatly reduce program complexity
- Write performance-critical code
- Comprehend the impact of a weak memory model in multi-threaded applications

*Who This Book Is For* Intermediate to advanced programmers and programming students

*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*

*The Art of 64-Bit Assembly, Volume 1*

*Mastering Assembly Programming*

*A Gentle Introduction to Computer Systems*

*Covers x86 64-bit, AVX, AVX2, and AVX-512*

*Fundamentals of Embedded Software*

*Guide to Assembly Language*

*16- and 32-Bit Low-Level Programming for the PC and Windows*

This is the third edition of this assembly language programming textbook introducing programmers to 64 bit Intel assembly language. The primary addition to the third edition is the discussion of the new version of the free integrated development environment, ebe, designed by the author specifically to meet the needs of assembly language programmers. The new ebe is a C++ program using the Qt library to implement a GUI environment consisting of a source window, a data window, a register, a floating point register window, a backtrace window, a console window, a terminal window and a project window along with 2 educational tools called the "toy box" and the "bit bucket." The source window includes a full-featured text editor with convenient controls for assembling, linking and debugging a program. The project facility allows a program to be built from C source code files and assembly source files. Assembly is performed automatically using the yasm assembler and linking is performed with ld or gcc. Debugging operates by transparently sending commands into the gdb debugger while automatically displaying registers and variables after each debugging step. Additional information about ebe can be found at <http://www.rayseyfarth.com>. The second important addition is support for the OS X operating system. Assembly language is similar enough between the two systems to cover in a single book. The book discusses the differences between the systems. The book is intended as a first assembly language book for programmers experienced in high level programming in a language like C or C++. The assembly programming is performed using the yasm assembler automatically from the ebe IDE under the Linux operating system. The book primarily teaches how to write assembly code compatible with C programs. The reader will learn to call C functions from assembly language and to call assembly functions from C in addition to writing complete programs in assembly language. The gcc compiler is used internally to compile C programs. The book starts early emphasizing using ebe to debug programs, along with teaching equivalent commands using gdb. Being able to single-step assembly programs is critical in learning assembly programming. Ebe makes this far easier than using gdb directly. Highlights of the book include doing input/output programming using the Linux system calls and the C library, implementing data structures in assembly language and high performance assembly language programming. Early chapters of the book rely on using the debugger to observe program behavior. After a chapter on functions, the user is prepared to use printf and scanf from the C library to perform I/O. The chapter on data structures covers singly linked lists, doubly linked circular lists, hash tables and binary trees. Test programs are presented for all these data structures. There is a chapter on optimization techniques and 3 chapters on specific optimizations. One chapter covers how to efficiently count the 1 bits in an array with the most efficient version using the recently-introduced popcnt instruction. Another chapter covers using SSE instructions to create an efficient implementation of the Sobel filtering algorithm. The final high performance programming chapter discusses computing correlation between data in 2 arrays. There is an AVX implementation which achieves 20.5 GFLOPs on a single core of a Core i7 CPU. A companion web site, <http://www.rayseyfarth.com>, has a collection of PDF slides which instructors can use for in-class presentations and source code for sample programs.

No one has done more to conquer the performance limitations of the PC than Michael Abrash, a software engineer for Microsoft. His complete works are contained in this massive volume, including everything he has written about performance coding and real-time graphics. The CD-ROM contains the entire text in Adobe Acrobat 3.0 format, allowing fast searches for specific facts.

Introduces the features of the C programming language, discusses data types, variables, operators, control flow, functions, pointers, arrays, and structures, and looks at the UNIX system interface. Master the booting procedure of various operating systems with in-depth analysis of bootloaders and firmware. The primary focus is on the Linux booting procedure along with other popular operating systems such as Windows and Unix. Hands-on Booting begins by explaining what a bootloader is, starting with the Linux bootloader followed by bootloaders for Windows and Unix systems. Next, you'll address the BIOS and UEFI firmware by installing multiple operating systems on one machine and booting them through the Linux bootloader. Further, you'll see the kernel's role in the booting procedure of the operating system and the dependency between kernel, initramfs, and dracut. You'll also cover systemd, examining its structure and how it mounts the user root filesystem. In the final section, the book explains troubleshooting methodologies such as debugging shells followed by live images and rescue mode. On completing this book, you will understand the booting process of major operating systems such as Linux, Windows, and Unix. You will also know how to fix the Linux booting issues through various boot modes. What You Will Learn Examine the BIOS and UEFI firmware Understanding the Linux boot loader (GRUB) Work with initramfs, dracut, and systemd Fix can't-boot issues on Linux Who This Book Is For Linux users, administrators, and developers.

Third Edition - for Linux and OS X

From instruction set to kernel module with Intel processor  
Assembly Programming and Computer Architecture

## Programming with 64-Bit ARM Assembly Language

### Low-Level Programming

### The Elements of Computing Systems

### A Decade of Programming Insight and Experience

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.

This book introduces basic programming of ARM Cortex chips in assembly language and the fundamentals of embedded system design. It presents data representations, assembly instruction syntax, implementing basic controls of C language at the assembly level, and instruction encoding and decoding. The book also covers many advanced components of embedded systems, such as software and hardware interrupts, general purpose I/O, LCD driver, keypad interaction, real-time clock, stepper motor control, PWM input and output, digital input capture, direct memory access (DMA), digital and analog conversion, and serial communication (USART, I2C, SPI, and USB).

A compiler translates a program written in a high level language into a program written in a lower level language. For students of computer science, building a compiler from scratch is a rite of passage: a challenging and fun project that offers insight into many different aspects of computer science, some deeply theoretical, and others highly practical. This book offers a one semester introduction into compiler construction, enabling the reader to build a simple compiler that accepts a C-like language and translates it into working X86 or ARM assembly language. It is most suitable for undergraduate students who have some experience programming in C, and have taken courses in data structures and computer architecture.

Mastering ARM hardware architecture opens a world of programming for nearly all phones and tablets including the iPhone/iPad and most Android phones. It's also the heart of many single board computers like the Raspberry Pi. Gain the skills required to dive into the fundamentals of the ARM hardware architecture with this book and start your own projects while you develop a working knowledge of assembly language for the ARM 64-bit processor. You'll review assembly language programming for the ARM Processor in 64-bit mode and write programs for a number of single board computers, including the Nvidia Jetson Nano and the Raspberry Pi (running 64-bit Linux). The book also discusses how to target assembly language programs for Apple iPhones and iPads along with 64-Bit ARM based Android phones and tablets. It covers all the tools you require, the basics of the ARM hardware architecture, all the groups of ARM 64-Bit Assembly instructions, and how data is stored in the computer's memory. In addition, interface apps to hardware such as the Raspberry Pi's GPIO ports. The book covers code optimization, as well as how to inter-operate with C and Python code. Readers will develop enough background to use the official ARM reference documentation for their own projects. With Programming with 64-Bit ARM Assembly Language as your guide you'll study how to read, reverse engineer and hack machine code, then be able to apply these new skills to study code examples and take control of both your ARM devices' hardware and software. What You'll Learn Make operating system calls from assembly language and include other software libraries in your projects Interface apps to hardware devices such as the Raspberry Pi GPIO ports Reverse engineer and hack code Use the official ARM reference documentation for your own projects Who This Book Is For Software developers who have already learned to program in a higher-level language like Python, Java, C#, or even C and now wish to learn Assembly programming.

Windows Assembly Language and Systems Programming

Programming Embedded Systems

X86-64 Assembly Language Programming with Ubuntu

32-bit, 64-bit, SSE, and AVX

x86-64 Machine Organization and Programming

Learn the Boot Process of Linux, Windows, and Unix

C, Assembly, and Program Execution on Intel® 64 Architecture

**For Assembly Language and Architecture courses emphasizing SPARC architecture found in computer science, engineering and business departments. Written from a programmer's perspective, this long-awaited revision introduces the SPARC assembly language to readers early on. Other introductory material encompasses making**

*use of UNIX tools (the m4 macro processor; the assembler; the gnu emacs editor; and the gdb debugger). Further coverage includes a formal definition of the von Neumann machine, its relationship to programmable calculators, and to the JAVA bytecode and JAVA virtual machine. Not only is this book suitable for introductory computer architecture courses, but for programmers who will be programming SPARC architecture machine in languages such as C and C++.*

*Dive into Systems is a vivid introduction to computer organization, architecture, and operating systems that is already being used as a classroom textbook at more than 25 universities. This textbook is a crash course in the major hardware and software components of a modern computer system. Designed for use in a wide range of introductory-level computer science classes, it guides readers through the vertical slice of a computer so they can develop an understanding of the machine at various layers of abstraction. Early chapters begin with the basics of the C programming language often used in systems programming. Other topics explore the architecture of modern computers, the inner workings of operating systems, and the assembly languages that translate human-readable instructions into a binary representation that the computer understands. Later chapters explain how to optimize code for various architectures, how to implement parallel computing with shared memory, and how memory management works in multi-core CPUs. Accessible and easy to follow, the book uses images and hands-on exercise to break down complicated topics, including code examples that can be modified and executed.*

**Low-Level Programming C, Assembly, and Program Execution on Intel x86-64 Architecture** Apress

*D is a programming language built to help programmers address the challenges of modern software development. It does so by fostering modules interconnected through precise interfaces, a federation of tightly integrated programming paradigms, language-enforced thread isolation, modular type safety, an efficient memory model, and more. The D Programming Language is an authoritative and comprehensive introduction to D. Reflecting the author's signature style, the writing is casual and conversational, but never at the expense of focus and precision. It covers all aspects of the language (such as expressions, statements, types, functions, contracts, and modules), but it is much more than an enumeration of features. Inside the book you will find In-depth explanations, with idiomatic examples, for all language features How feature groups support major programming paradigms Rationale and best-use advice for each major feature Discussion of cross-cutting issues, such as error handling, contract programming, and concurrency Tables, figures, and "cheat sheets" that serve as a handy quick reference for day-to-day problem solving with D Written for the working programmer, The D Programming Language not only introduces the D language—it presents a compendium of good practices and idioms to help both your coding with D and your coding in general.*

**Hackers**

**Michael Abrash's Graphics Programming Black Book**

**The Rust Programming Language (Covers Rust 2018)**

**C, Assembly, and Program Execution on Intel x86-64 Architecture**

**Building a Modern Computer from First Principles**

**Real World Instrumentation with Python**

**Modern X86 Assembly Language Programming**

Computer Science

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.

Reflecting current industrial applications and programming practice, this book lays a foundation that supports the multi-threaded style of programming and high-reliability requirements of embedded software. Using a non-product specific approach and a programming (versus hardware) perspective, it focuses on the 32-bit protected mode processors and on C as the dominant programming language—with coverage of Assembly and how it can be used in conjunction with, and support of, C. Features an abundance of examples in C and an accompanying CD-ROM with software tools. Data Representation. Getting the Most Out of C. A Programmer's View of Computer Organization. Mixing C and Assembly. Input/Output Programming. Concurrent Software. Scheduling. Memory Management. Shared Memory. System Initialization. For Computer Scientists, Computer Engineers, and Electrical Engineers involved with embedded software applications.

History of Programming Languages presents information pertinent to the technical aspects of the language design and



creation. This book provides an understanding of the processes of language design as related to the environment in which languages are developed and the knowledge base available to the originators. Organized into 14 sections encompassing 77 chapters, this book begins with an overview of the programming techniques to use to help the system produce efficient programs. This text then discusses how to use parentheses to help the system identify identical subexpressions within an expression and thereby eliminate their duplicate calculation. Other chapters consider FORTRAN programming techniques needed to produce optimum object programs. This book discusses as well the developments leading to ALGOL 60. The final chapter presents the biography of Adin D. Falkoff. This book is a valuable resource for graduate students, practitioners, historians, statisticians, mathematicians, programmers, as well as computer scientists and specialists.

Heroes of the Computer Revolution - 25th Anniversary Edition

Embedded Systems with Arm Cortex-M Microcontrollers in Assembly Language and C: Third Edition

Single Board Computer Development for Raspberry Pi and Mobile Devices

Understanding the Machine

Industrial Motor Control

Learn to Program with C

Programming from the Ground Up

This concise guide is designed to enable the reader to learn how to program in assembly language as quickly as possible. Through a hands-on programming approach, readers will also learn about the architecture of the Intel processor, and the relationship between high-level and low-level languages. This updated second edition has been expanded with additional exercises, and enhanced with new material on floating-point numbers and 64-bit processing. Topics and features: provides guidance on simplified register usage, simplified input/output using C-like statements, and the use of high-level control structures; describes the implementation of control structures, without the use of high-level structures, and often with related C program code; illustrates concepts with one or more complete program; presents review summaries in each chapter, together with a variety of exercises, from short-answer questions to programming assignments; covers selection and iteration structures, logic, shift, arithmetic shift, rotate, and stack instructions, procedures and macros, arrays, and strings; includes an introduction to floating-point instructions and 64-bit processing; examines machine language from a discovery perspective, introducing the principles of computer organization. A must-have resource for undergraduate students seeking to learn the fundamentals necessary to begin writing logically correct programs in a minimal amount of time, this work will serve as an ideal textbook for an assembly language course, or as a supplementary text for courses on computer organization and architecture. The presentation assumes prior knowledge of the basics of programming in a high-level language such as C, C++, or Java.

Authored by two of the leading authorities in the field, this guide offers readers the knowledge and skills needed to achieve proficiency with embedded software.

Push the limits of what C - and you - can do, with this high-intensity guide to the most advanced capabilities of C Key FeaturesMake the most of C's low-level control, flexibility, and high performanceA comprehensive guide to C's most powerful and challenging featuresA thought-provoking guide packed with hands-on exercises and examplesBook Description There's a lot more to C than knowing the language syntax. The industry looks for developers with a rigorous, scientific understanding of the principles and practices. Extreme C will teach you to use C's advanced low-level power to write effective, efficient systems. This intensive, practical guide will help you become an expert C programmer. Building on your existing C knowledge, you will master preprocessor directives, macros, conditional compilation, pointers, and much more. You will gain new insight into algorithm design, functions, and structures. You will discover how C helps you squeeze maximum performance out of critical, resource-constrained applications. C still plays a critical role in 21st-century programming, remaining the core language for precision engineering, aviations, space research, and more. This book shows how C works with Unix, how to implement OO principles in C, and fully covers multi-processing. In Extreme C, Amini encourages you to think, question, apply, and experiment for yourself. The book is essential for anybody who wants to take their C to the next level. What you will learnBuild advanced C knowledge on strong foundations, rooted in first principlesUnderstand memory structures and compilation pipeline and how they work, and how to make most out of themApply object-oriented design principles to your procedural C codeWrite low-level code that's close to the hardware and squeezes maximum performance out of a computer systemMaster concurrency, multithreading, multi-processing, and integration with other languagesUnit Testing and debugging, build systems, and inter-process communication for C programmingWho this book is for Extreme C is for C programmers who want to dig deep into the language and its capabilities. It will help you make the most of the low-level control C gives you. Learn how to develop your own applications to monitor or control instrumentation hardware. Whether you need to acquire data from a device or automate its functions, this practical book shows you how to use Python's rapid development capabilities to build interfaces that include everything from software to wiring. You get step-by-step instructions, clear examples, and hands-on tips for interfacing a PC to a variety of devices. Use the book's hardware survey to identify the interface type for your particular device, and then follow detailed examples to develop an interface with Python and C. Organized by interface type, data processing activities, and user interface implementations, this book is for anyone who works with instrumentation, robotics, data acquisition, or process control. Understand how to define the scope of an application and determine the algorithms necessary, and why it's important Learn how to use industry-standard interfaces such as RS-232, RS-485, and GPIB Create low-level extension modules in C to interface Python with a variety of hardware and test instruments Explore the console, curses, TkInter, and wxPython for graphical and text-based user interfaces Use open source software tools and libraries to reduce costs and avoid implementing functionality from scratch

Dive Into Systems

Extreme C

With C and GNU Development Tools

Taking you to the limit in Concurrency, OOP, and the most advanced capabilities of C

The C Programming Language

Write Great Code, Volume 1

Hands-on Booting

**Programming from the Ground Up** uses Linux assembly language to teach new programmers the most important concepts in programming. It takes you a step at a time through these concepts: \* How the processor views memory \* How the processor operates \* How programs interact with the operating system \* How computers represent data internally \* How to do low-level and high-level optimization Most beginning-level programming books attempt to shield the reader from how their computer really works.

**Programming from the Ground Up** starts by teaching how the computer works under the hood, so that the programmer will have a sufficient background to be successful in all areas of programming. This book is being used by Princeton University in their COS 217 "Introduction to Programming Systems" course.

A detailed introduction to the C programming language for experienced programmers. The world runs on code written in the C programming language, yet most schools begin the curriculum with Python or Java. **Effective C** bridges this gap and brings C into the modern era--covering the modern C17 Standard as well as potential C2x features. With the aid of this instant classic, you'll soon be writing professional, portable, and secure C programs to power robust systems and solve real-world problems. Robert C. Seacord introduces C and the C Standard Library while addressing best practices, common errors, and open debates in the C community. Developed together with other C Standards committee experts, **Effective C** will teach you how to debug, test, and analyze C programs. You'll benefit from Seacord's concise explanations of C language constructs and behaviors, and from his 40 years of coding experience. You'll learn: • How to identify and handle undefined behavior in a C program • The range and representations of integers and floating-point values • How dynamic memory allocation works and how to use nonstandard functions • How to use character encodings and types • How to perform I/O with terminals and filesystems using C Standard streams and POSIX file descriptors • How to understand the C compiler's translation phases and the role of the preprocessor • How to test, debug, and analyze C programs **Effective C** will teach you how to write professional, secure, and portable C code that will stand the test of time and help strengthen the foundation of the computing world.

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.

**INDUSTRIAL MOTOR CONTROL 7E** is an integral part of any electrician training. Comprehensive and up to date, this book provides crucial information on basic relay control systems, programmable logic controllers, and solid state devices commonly found in an industrial setting. Written by a highly qualified and respected author, you will find easy-to-follow instructions and essential information on controlling industrial motors and commonly used devices in contemporary industry. **INDUSTRIAL MOTOR CONTROL 7E** successfully bridges the gap between industrial maintenance and instrumentation, giving you a fundamental understanding of the operation of variable frequency drives, solid state relays, and other applications that employ electronic devices. **Important Notice:** Media content referenced within the product description or the product text may not be available in the ebook version.

Ruminations on C++

A Fast-Paced Introduction

A Concise Introduction

Functional C

Introduction to 64 Bit Assembly Programming for Linux and OS X

An Introduction to Professional C Programming

Second Edition

***This book teaches programmers and programming students how to use x64 assembly to write low-level code in C for performance-critical programs and how to compile and execute it inside the Intel 64 hardware and OS framework. Low-Level Programming presents Intel 64 architecture as a development of von Neumann architecture featuring protection mechanisms and performance amplifiers such as caches and branch predicting. It proceeds to investigate the compilation cycle and ELF object files. Elucidating a structured approach to C with code examples, exercises, and a companion annex of source code, the book models best coding practices for implementing language abstractions on top of assembly. The author examines the optimization capabilities and limits of modern compilers, and he demonstrates the use of various performance-gain techniques, such as specialized instructions and prefetching. What Readers Will Learn***  
***Low-Level Programming teaches programmers how to use assembly language and C to write code for Intel 64 platforms and to look under the hood for various purposes, including the following:***• Making code more performant on

***the assembly level• Debugging compiler and optimizer errors in native code• Fixing executables by disassembly in the absence of source code• Diagnosing malware Who This Book Is ForIntermediate-to-advanced programmers and programming students.***

***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).***

***This title gives students an integrated and rigorous picture of applied computer science, as it comes to play in the construction of a simple yet powerful computer system.***

***This 25th anniversary edition of Steven Levy's classic book traces the exploits of the computer revolution's original hackers -- those brilliant and eccentric nerds from the late 1950s through the early '80s who took risks, bent the rules, and pushed the world in a radical new direction. With updated material from noteworthy hackers such as Bill Gates, Mark Zuckerberg, Richard Stallman, and Steve Wozniak, Hackers is a fascinating story that begins in early computer research labs and leads to the first home computers. Levy profiles the imaginative brainiacs who found clever and unorthodox solutions to computer engineering problems. They had a shared sense of values, known as "the hacker ethic," that still thrives today. Hackers captures a seminal period in recent history when underground activities blazed a trail for today's digital world, from MIT students finagling access to clunky computer-card machines to the DIY culture that spawned the Altair and the Apple II.***