Modern X86 Assembly Language Programming

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

Daniel Kusswurm

Modern X86 Assembly Language Programming: 32-bit, 64-bit, SSE, and AVX

Copyright © 2014 by Daniel Kusswurm

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN-13 (pbk): 978-1-4842-0065-0

ISBN-13 (electronic): 978-1-4842-0064-3

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director: Welmoed Spahr

Lead Editors: Steve Weiss (Apress): Patrick Hauke (Intel)

Technical Reviewer: Paul Cohen

Editorial Board: Steve Anglin, Gary Cornell, Louise Corrigan, James T. DeWolf, Jonathan Gennick, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Gwenan Spearing,

Matt Wade, Steve Weiss

Coordinating Editor: Melissa Maldonado

Copy Editor: Kezia Endsley Compositor: SPi Global Indexer: SPi Global Artist: SPi Global

Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

This book is dedicated to those individuals who suffer the ravages of Alzheimer's disease and their unsung compassionate caregivers.

Contents at a Glance

About the Author	xix
About the Technical Reviewer	хх
Acknowledgments	xxii
Introduction	xxı
■Chapter 1: X86-32 Core Architecture	1
■Chapter 2: X86-32 Core Programming	27
■Chapter 3: X87 Floating-Point Unit	87
■Chapter 4: X87 FPU Programming	103
■Chapter 5: MMX Technology	133
■Chapter 6: MMX Technology Programming	147
■ Chapter 7: Streaming SIMD Extensions	179
■ Chapter 8: X86-SSE programming – Scalar Floating-Point	207
■Chapter 9: X86-SSE Programming – Packed Floating-Point	237
■Chapter 10: X86-SSE Programming – Packed Integers	273
■Chapter 11: X86-SSE Programming – Text Strings	303
■ Chapter 12: Advanced Vector Extensions (AVX)	327
■ Chapter 13: X86-AVX Programming - Scalar Floating-Point	35 1
■ Chapter 14: X86-AVX Programming - Packed Floating-Point	377
■Chapter 15: X86-AVX Programming - Packed Integers	405

■ CONTENTS AT A GLANCE

Chapter 16: X86-AVX Programming - New Instructions	439
Chapter 17: X86-64 Core Architecture	491
Chapter 18: X86-64 Core Programming	503
Chapter 19: X86-64 SIMD Architecture	557
Chapter 20: X86-64 SIMD Programming	563
■Chapter 21: Advanced Topics and Optimization Techniqu	es 623
Chapter 22: Advanced Topics Programming	637
Index	657

Contents

About the Author	XİX
About the Technical Reviewer	xxi
Acknowledgments	xxiii
Introduction	xxv
■Chapter 1: X86-32 Core Architecture	1
Historical Overview	1
Data Types	4
Fundamental Data Types	4
Numerical Data Types	5
Packed Data Types	6
Miscellaneous Data Types	7
Internal Architecture	8
Segment Registers	9
General-Purpose Registers	9
EFLAGS Register	11
Instruction Pointer	13
Instruction Operands	13
Memory Addressing Modes	14
Instruction Set Overview	15
Data Transfer	18
Binary Arithmetic	18
Data Comparison	20

Data Conversion	20
Logical	21
Rotate and Shift	21
Byte Set and Bit String	22
String	22
Flag Manipulation	23
Control Transfer	24
Miscellaneous	25
Summary	25
Chapter 2: X86-32 Core Programming	27
Getting Started	28
First Assembly Language Function	28
Integer Multiplication and Division	32
X86-32 Programming Fundamentals	36
Calling Convention	37
Memory Addressing Modes	41
Integer Addition	46
Condition Codes	49
Arrays	54
One-Dimensional Arrays	55
Two-Dimensional Arrays	60
Structures	67
Simple Structures	67
Dynamic Structure Creation	70
Strings	73
Counting Characters	74
String Concatenation	76

Comparing Arrays	80
Array Reversal	83
Summary	86
■Chapter 3: X87 Floating-Point Unit	87
X87 FPU Core Architecture	87
Data Registers	88
X87 FPU Special-Purpose Registers	88
X87 FPU Operands and Encodings	91
X87 FPU Instruction Set	95
Data Transfer	95
Basic Arithmetic	96
Data Comparison	98
Transcendental	100
Constants	101
Control	101
Summary	102
■Chapter 4: X87 FPU Programming	103
X87 FPU Programming Fundamentals	103
Simple Arithmetic	104
Floating-Point Compares	108
X87 FPU Advanced Programming	112
Floating-Point Arrays	112
Transcendental Instructions	120
Advanced Stack Usage	124
Summary	130

4	Chapter 5: MMX Technology	133
	SIMD Processing Concepts	133
	Wraparound vs. Saturated Arithmetic	135
	MMX Execution Environment	137
	MMX Instruction Set	138
	Data Transfer	
	Arithmetic	139
	Comparison	142
	Conversion	142
	Logical and Shift	142
	Unpack and Shuffle	143
	Insertion and Extraction	144
	State and Cache Control	145
	Summary	145
	Chapter 6: MMX Technology Programming	147
	Chapter 6: MMX Technology Programming MMX Programming Fundamentals	
		147
	MMX Programming Fundamentals	147 149
	MMX Programming Fundamentals Packed Integer Addition	147 149 156
	MMX Programming Fundamentals Packed Integer Addition Packed Integer Shifts	147 149 156
	MMX Programming Fundamentals Packed Integer Addition Packed Integer Shifts Packed Integer Multiplication	147149156160
	MMX Programming Fundamentals Packed Integer Addition Packed Integer Shifts Packed Integer Multiplication MMX Advanced Programming	147156160164
	MMX Programming Fundamentals Packed Integer Addition Packed Integer Shifts Packed Integer Multiplication MMX Advanced Programming Integer Array Processing	147156160164172
	MMX Programming Fundamentals Packed Integer Addition Packed Integer Shifts Packed Integer Multiplication MMX Advanced Programming Integer Array Processing Using MMX and the x87 FPU	147156160164172
	MMX Programming Fundamentals Packed Integer Addition Packed Integer Shifts Packed Integer Multiplication MMX Advanced Programming Integer Array Processing Using MMX and the x87 FPU Summary	147156160164172178
	MMX Programming Fundamentals Packed Integer Addition Packed Integer Shifts Packed Integer Multiplication MMX Advanced Programming Integer Array Processing Using MMX and the x87 FPU Summary Chapter 7: Streaming SIMD Extensions X86-SSE Overview	147156160164172178
	MMX Programming Fundamentals Packed Integer Addition Packed Integer Shifts Packed Integer Multiplication MMX Advanced Programming Integer Array Processing Using MMX and the x87 FPU Summary Chapter 7: Streaming SIMD Extensions	147156160164172178179

X86-SSE Data Types	181
X86-SSE Control-Status Register	183
X86-SSE Processing Techniques	184
X86-SSE Instruction Set Overview	188
Scalar Floating-Point Data Transfer	190
Scalar Floating-Point Arithmetic	190
Scalar Floating-Point Comparison	191
Scalar Floating-Point Conversion	191
Packed Floating-Point Data Transfer	192
Packed Floating-Point Arithmetic	193
Packed Floating-Point Comparison	195
Packed Floating-Point Conversion	195
Packed Floating-Point Shuffle and Unpack	196
Packed Floating-Point Insertion and Extraction	197
Packed Floating-Point Blend	197
Packed Floating-Point Logical	198
Packed Integer Extensions	198
Packed Integer Data Transfer	199
Packed Integer Arithmetic	199
Packed Integer Comparison	200
Packed Integer Conversion	201
Packed Integer Shuffle and Unpack	202
Packed Integer Insertion and Extraction	202
Packed Integer Blend	203
Packed Integer Shift	203
Text String Processing	204
Non-Temporal Data Transfer and Cache Control	204
Miscellaneous	205
Summary	206

■ Chapter 8: X86-SSE programming – Scalar Floating-Point	207
Scalar Floating-Point Fundamentals	207
Scalar Floating-Point Arithmetic	207
Scalar Floating-Point Compare	212
Scalar Floating-Point Conversions	217
Advanced Scalar Floating-Point Programming	225
Scalar Floating-Point Spheres	225
Scalar Floating-Point Parallelograms	228
Summary	236
■ Chapter 9: X86-SSE Programming – Packed Floating-Point	237
Packed Floating-Point Fundamentals	237
Packed Floating-Point Arithmetic	238
Packed Floating-Point Compare	244
Packed Floating-Point Conversions	248
Advanced Packed Floating-Point Programming	254
Packed Floating-Point Least Squares	254
Packed Floating-Point 4 × 4 Matrices	260
Summary	271
■ Chapter 10: X86-SSE Programming – Packed Integers	
Packed Integer Fundamentals	
Advanced Packed Integer Programming	
Packed Integer Histogram	
Packed Integer Threshold	
Summary	302

■ Chapter 11: X86-SSE Programming – Text Strings	303
Text String Fundamentals	303
Text String Programming	311
Text String Calculate Length	311
Text String Replace Characters	316
Summary	325
■Chapter 12: Advanced Vector Extensions (AVX)	327
X86-AVX Overview	
X86-AVX Execution Environment	
X86-AVX Register Set	
X86-AVX Data Types	329
X86-AVX Instruction Syntax	330
X86-AVX Feature Extensions	332
X86-AVX Instruction Set Overview	333
Promoted x86-SSE Instructions	333
New Instructions	336
Feature Extension Instructions	342
Summary	349
■Chapter 13: X86-AVX Programming - Scalar Floating-Point	351
Programming Fundamentals	351
Scalar Floating-Point Arithmetic	351
Scalar Floating-Point Compares	355
Advanced Programming	360
Roots of a Quadratic Equation	360
Spherical Coordinates	368
Summary	376

Chapter 14: X86-AVX Programming - Packed Floating-Point	377
Programming Fundamentals	377
Packed Floating-Point Arithmetic	378
Packed Floating-Point Compares	385
Advanced Programming	389
Correlation Coefficient	389
Matrix Column Means	396
Summary	403
Chapter 15: X86-AVX Programming - Packed Integers	405
Packed Integer Fundamentals	405
Packed Integer Arithmetic	405
Packed Integer Unpack Operations	412
Advanced Programming	417
Image Pixel Clipping	417
Image Threshold Part Deux	425
Summary	437
Chapter 16: X86-AVX Programming - New Instructions	439
Detecting Processor Features (CPUID)	439
Data-Manipulation Instructions	447
Data Broadcast	447
Data Blend	453
Data Permute	458
Data Gather	463
Fused-Multiply-Add Programming	470
General-Purpose Register Instructions	482
Flagless Multiplication and Bit Shifts	482
Enhanced Bit Manipulation	486
Summary	490

	491
Internal Architecture	491
General-Purpose Registers	492
RFLAGS Register	494
Instruction Pointer Register	494
Instruction Operands	494
Memory Addressing Modes	495
Differences Between X86-64 and X86-32	497
Instruction Set Overview	499
Basic Instruction Use	499
Invalid Instructions	500
New Instructions	500
Deprecated Resources	502
Summary	502
■Chapter 18: X86-64 Core Programming	503
X86-64 Programming Fundamentals	503
Integer Arithmetic	504
Memory Addressing	
Memory AddressingInteger Operands	511
·	511 514
Integer Operands	511 514 519
Integer OperandsFloating-Point Arithmetic	511 514 519 523
Integer Operands	511 514 519 523
Integer Operands Floating-Point Arithmetic X86-64 Calling Convention Basic Stack Frames	511514519523524
Integer Operands Floating-Point Arithmetic X86-64 Calling Convention Basic Stack Frames Using Non-Volatile Registers	511514519523524528
Integer Operands Floating-Point Arithmetic X86-64 Calling Convention Basic Stack Frames Using Non-Volatile Registers Using Non-Volatile XMM Registers	511514519523524528533
Integer Operands Floating-Point Arithmetic X86-64 Calling Convention Basic Stack Frames Using Non-Volatile Registers Using Non-Volatile XMM Registers Macros for Prologs and Epilogs	511514519523524528533539
Integer Operands Floating-Point Arithmetic X86-64 Calling Convention Basic Stack Frames Using Non-Volatile Registers Using Non-Volatile XMM Registers Macros for Prologs and Epilogs X86-64 Arrays and Strings	511514519523524533539546

■ Chapter 19: X86-64 SIMD Architecture	557
X86-SSE-64 Execution Environment	557
X86-SSE-64 Register Set	557
X86-SSE-64 Data Types	559
X86-SSE-64 Instruction Set Overview	559
X86-AVX Execution Environment	560
X86-AVX-64 Register Set	560
X86-AVX-64 Data Types	561
X86-AVX-64 Instruction Set Overview	562
Summary	562
■Chapter 20: X86-64 SIMD Programming	563
X86-SSE-64 Programming	563
Image Histogram	563
Image Conversion	571
Vector Arrays	580
X86-AVX-64 Programming	590
Ellipsoid Calculations	590
RGB Image Processing	595
Matrix Inverse	602
Miscellaneous Instructions	617
Summary	622
■Chapter 21: Advanced Topics and Optimization Techniqu	ıes 623
Processor Microarchitecture	623
Multi-Core Processor Overview	624
Microarchitecture Pipeline Functionality	626
Execution Engine	628

Optimizing Assembly Language Code	629
Basic Optimizations	630
Floating-Point Arithmetic	631
Program Branches	631
Data Alignment	633
SIMD Techniques	634
Summary	635
■Chapter 22: Advanced Topics Programming	637
Non-Temporal Memory Stores	637
Data Prefetch	645
Summary	656
Index	657

About the Author



Daniel Kusswurm has over 30 years of professional experience as a software developer and computer scientist. During his career, he has developed innovative software for medical devices, scientific instruments, and image processing applications. On many of these projects, he successfully employed x86 assembly language to significantly improve the performance of computationally-intense algorithms or solve unique programming challenges. His educational background includes a BS in Electrical Engineering Technology from Northern Illinois University along with an MS and PhD in Computer Science from DePaul University.

About the Technical Reviewer



Paul Cohen joined Intel Corporation during the very early days of the x86 architecture, starting with the 8086, and retired from Intel after 26 years in sales/marketing/management. He is currently partnered with Douglas Technology Group, focusing on the creation of technology books on behalf of Intel and other corporations. Paul also teaches a class that transforms middle and high school students into real, confident entrepreneurs, in conjunction with the Young Entrepreneurs Academy (YEA) and is a Traffic Commissioner for the City of Beaverton, Oregon and on the Board of Directors of multiple non-profit organizations.

Acknowledgments

The production of a motion picture and the publication of a book are somewhat analogous. Movie trailers extol the performances of the lead actors. The front cover of a book trumpets the authors' names. Actors and authors ultimately receive public acclamation for their efforts. It is, however, impossible to produce a movie or publish a book without the dedication, expertise, and creativity of a professional behind-the-scenes team. This book is no exception.

I would like to thank Patrick Hauke for his valuable advice and championing of the book project during its conceptual stage. I am indebted to Steve Weiss for his editorial savvy and guidance through the book publishing jungle. I am extremely appreciative of Melissa Maldonado's efforts to keep me and everyone else focused and on schedule. Paul Cohen deserves kudos for his meticulous technical review and practical suggestions. Copy editor Kezia Endsley and proofreader Ed Kusswurm merit applause and recognition for their hard work and constructive feedback. I accept full responsibility for any remaining imperfections.

I would also like to thank Dhaneesh Kumar and the entire production staff at Apress for their contributions, Vyacheslav Klochkov and Mitch Bodart for their help in clarifying how to effectively use the FMA instructions, and my professional colleagues for their support and encouragement. Finally, I would like to recognize parental nodes Armin (RIP) and Mary along with sibling nodes Mary, Tom, Ed, and John for their inspiration during the writing of this book

Introduction

Since the invention of the personal computer, software developers have used assembly language to create innovative solutions for a wide variety of algorithmic challenges. During the early days of the PC era, it was common practice to code large portions of a program or complete applications using x86 assembly language. Even as the use of high-level languages such as C, C++, and C# became more prevalent, many software developers continued to employ assembly language to code performance-critical sections of their programs. And while compilers have improved remarkably over the years in terms of generating machine code that is both spatially and temporally efficient, situations still exist where it makes sense for software developers to exploit the benefits of assembly language programming.

The inclusion of single-instruction multiple-data (SIMD) architectures in modern x86 processors provides another reason for the continued interest in assembly language programming. A SIMD-capable processor includes computational resources that facilitate concurrent calculations using multiple data values, which can significantly improve the performance of applications that must deliver real-time responsiveness. SIMD architectures are also well-suited for computationally-intense problem domains such as image processing, audio and video encoding, computer-aided design, computer graphics, and data mining. Unfortunately, many high-level languages and development tools are unable to fully (or even partially) exploit the SIMD capabilities of a modern x86 processor. Assembly language, on the other hand, enables the software developer to take full advantage of a processor's entire computational resource suite.

Modern X86 Assembly Language Programming

Modern X86 Assembly Language Programming is an edifying text on the subject of x86 assembly language programming. Its primary purpose is to teach you how to code functions using x86 assembly language that can be invoked from a high-level language. The book includes informative material that explains the internal architecture of an x86 processor as viewed from the perspective of an application program. It also contains an abundance of sample code that is structured to help you quickly understand x86 assembly language programming and the computational resources of the x86 platform. Major topics of the book include the following:

- X86 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 the fundamentals of packed integer arithmetic
- Streaming SIMD extensions (SSE) and Advanced Vector Extensions (AVX), including internal registers, packed integer and floating-point arithmetic, and associated instruction sets
- X86 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 microarchitecture and assembly language optimization techniques

Before proceeding I should also explicitly mention some of the topics that are not covered. This book does not examine legacy aspects of x86 assembly language programming such as 16-bit real-mode applications or segmented memory models. Except for a few historical observations and comparisons, all of the discussions and sample code emphasize x86 protected-mode programming using a flat linear memory model. This book does not discuss x86 instructions or architectural features that are managed by operating systems or require elevated privileges. It also doesn't explore how to use x86 assembly language to develop software that is intended for operating systems or device drivers. However, if your ultimate goal is to use x86 assembly language to create software for one of these environments, you will need to thoroughly understand the material presented in this book.

While it is still theoretically possible to write an entire application program using assembly language, the demanding requirements of contemporary software development make such an approach impractical and ill advised. Instead, this book concentrates on creating x86 assembly language modules and functions that are callable from C++. All of the sample code and programing examples presented in this book use Microsoft Visual C++ and Microsoft Macro Assembler. Both of these tools are included with Microsoft's Visual Studio development tool.

Target Audience

The target audience for this book is software developers, including:

- Software developers who are creating application programs for Windows-based platforms and want to learn how to write performance-enhancing algorithms and functions using x86 assembly language.
- Software developers who are creating application programs for non-Windows environments and want to learn x86 assembly language programming.

- Software developers who have a basic understanding of x86 assembly language programming and want to learn how to use the x86's SSE and AVX instruction sets.
- Software developers and computer science students who want or need to gain a better understanding of the x86 platform, including its internal architecture and instruction sets.

The principal audience for Modern X86 *Assembly Language Programming* is Windows software developers since the sample code uses Visual C++ and Microsoft Macro Assembler. It is important to note, however, that this is not a book on how to use the Microsoft development tools. Software developers who are targeting non-Windows platforms also can learn from the book since most of the informative content is organized and communicated independent of any specific operating system. In order to understand the book's subject material and sample code, a background that includes some programming experience using C or C++ will be helpful. Prior experience with Visual Studio or knowledge of a particular Windows API is not a prerequisite to benefit from the book.

Outline of Book

The primary objective of this book is to help you learn x86 assembly language programming. In order to achieve this goal, you must also thoroughly understand the internal architecture and execution environment of an x86 processor. The book's chapters and content are organized with this in mind. The following paragraphs summarize the book's major topics and each chapter's content.

X86-32 Core Architecture—Chapter 1 covers the core architecture of the x86-32 platform. It includes a discussion of the platform's fundamental data types, internal architecture, instruction operands, and memory addressing modes. This chapter also presents an overview of the core x86-32 instruction set. Chapter 2 explains the fundamentals of x86-32 assembly language programming using the core x86-32 instruction set and common programming constructs. All of the sample code discussed in Chapter 2 (and subsequent chapters) is packaged as working programs, which means that you can run, modify, or otherwise experiment with the code in order to enhance your learning experience.

X87 Floating-Point Unit—Chapter 3 surveys the architecture of the x87 floating-point unit (FPU) and includes operational descriptions of the x87 FPU's register stack, control word register, status word register, and instruction set. This chapter also delves into the binary encodings that are used to represent floating-point numbers and certain special values. Chapter 4 contains an assortment of sample code that demonstrates how to perform floating-point calculations using the x87 FPU instruction set. Readers who need to maintain an existing x87 FPU code base or are targeting processors that lack the scalar floating-point capabilities of x86-SSE and x86-AVX (e.g., Intel's Quark) will benefit the most from this chapter.

MMX Technology—Chapter 5 describes the x86's first SIMD extension, which is called MMX technology. It examines the architecture of MMX technology including its register set, operand types, and instruction set. This chapter also discusses a number of related topics, including SIMD processing concepts and the mechanics of packed-

integer arithmetic. Chapter 6 includes sample code that illustrates basic MMX operations, including packed-integer arithmetic (both wraparound and saturated), integer array processing, and how to properly handle transitions between MMX and x87 FPU code.

Streaming SIMD Extensions—Chapter 7 focuses on the architecture of Streaming SIMD Extensions (SSE). X86-SSE adds a new set of 128-bit wide registers to the x86 platform and incorporates several instruction set additions that support computations using packed integers, packed floating-point (both single and double precision), and text strings. Chapter 7 also discusses the scalar floating-point capabilities of x86-SSE, which can be used to both simplify and improve the performance of algorithms that require scalar floating-point arithmetic. Chapters 8 - 11 contain an extensive collection of sample code that highlights use of the x86-SSE instruction set. Included in this chapter are several examples that demonstrate using the packed-integer capabilities of x86-SSE to perform common image-processing tasks, such as histogram construction and pixel thresholding. These chapters also include sample code that illustrates how to use the packed floating-point, scalar floating-point, and text string-processing instructions of x86-SSE.

Advanced Vector Extensions—Chapter 12 explores the x86's most recent SIMD extension, which is called Advanced Vector Extensions (AVX). This chapter explains the x86-AVX execution environment, its data types and register sets, and the new three-operand instruction syntax. It also discusses the data broadcast, gather, and permute capabilities of x86-AVX along with several x86-AVX concomitant extensions, including fused-multiply-add (FMA), half-precision floating-point, and new general-purpose register instructions. Chapters 13 - 16 contain sample code that depicts use of the various x86-AVX computational resources. Examples include using the x86-AVX instruction set with packed integers, packed floating-point, and scalar floating-point operands. These chapters also contain sample code that explicates use of the data broadcast, gather, permute, and FMA instructions.

X86-64 Core Architecture—Chapter 17 peruses the x86-64 platform and includes a discussion of the platform's core architecture, supported data types, general purpose registers, and status flags. It also explains the enhancements made to the x86-32 platform in order to support 64-bit operands and memory addressing. The chapter concludes with a discussion of the x86-64 instruction set, including those instructions that have been deprecated or are no longer available. Chapter 18 explores the fundamentals x86-64 assembly language programming using a variety of sample code. Examples include how to perform integer calculations using operands of various sizes, memory addressing modes, scalar floating-point arithmetic, and common programming constructs. Chapter 18 also explains the calling convention that must be observed in order to invoke an x86-64 assembly language function from C++.

X86-64 SSE and AVX—Chapter 19 describes the enhancements to x86-SSE and x86-AVX that are available on the x86-64 platform. This includes a discussion of the respective execution environments and extended data register sets. Chapter 20 contains sample code that highlights use of the x86-SSE and x86-AVX instruction sets with the x86-64 core architecture.

Advanced Topics—The last two chapters of this book consider advanced topics and optimization techniques related to x86 assembly language programming. Chapter 21 examines key elements of an x86 processor's microarchitecture, including its front-end pipelines, out-of-order execution model, and internal execution units. It also includes a discussion of programming techniques that you can employ to write x86 assembly

language code that is both spatially and temporally efficient. Chapter 22 contains sample code that illustrates several advanced assembly language programming techniques.

Appendices—The final section of the book includes several appendices. Appendix A contains a brief tutorial on how to use Microsoft's Visual C++ and Macro Assembler. Appendix B summarizes the x86-32 and x86-64 calling conventions that assembly language functions must observe in order to be invoked from a Visual C++ function. Appendix C contains a list of references and resources that you can consult for more information about x86 assembly language programming.

Sample Code Requirements

You can download the sample code for this book from the Apress website at http://www.apress.com/9781484200650. The following hardware and software is required to build and run the sample code:

- A PC with an x86 processor that is based on a recent microarchitecture. All of the x86-32, x87 FPU, MMX, and x86-SSE sample code can be executed using a processor based on the Nehalem (or later) microarchitecture. PCs with processors based on earlier microarchitectures also can be used to run many of the sample code programs. The AVX and AXV2 sample code requires a processor based on the Sandy Bridge or Haswell microarchitecture, respectively.
- Microsoft Windows 8.x or Windows 7 with Service Pack 1. A 64-bit version of Windows is required to run the x86-64 sample code.
- Visual Studio Professional 2013 or Visual Studio Express 2013 for Windows Desktop. The Express edition can be freely downloaded from the following Microsoft website: http://msdn. microsoft.com/en-us/vstudio. Update 3 is recommended for both Visual Studio editions.

■ **Caution** The primary purpose of the sample code is to elucidate the topics and technologies presented in this book. Minimal attention is given to important software engineering concerns such as robust error handling, security risks, numerical stability, rounding errors, or ill-conditioned functions. You are responsible for addressing these issues should you decide to use any of the sample code in your own programs.

Terminology and Conventions

The following paragraphs define the meaning of common terms and expressions used throughout this book. A *function, subroutine,* or *procedure* is a self-contained unit of executable code that accepts zero or more arguments, performs an operation, and optionally returns a value. Functions are typically invoked using the processor's call instruction. A *thread* is the smallest unit of execution that is managed and scheduled by an operating system. A *task* or *process* is a collection of one or more threads that share the same logical memory space. An *application* or *program* is a complete software package that contains at least one task.

The terms *x86-32* and *x86-64* are used respectively to describe 32-bit and 64-bit aspects, resources, or capabilities of a processor; *x86* is employed for features that are common to both 32-bit and 64-bit architectures. The expressions *x86-32 mode* and *x86-64 mode* denote a specific processor execution environment with the primary difference being the latter mode's support of 64-bit registers, operands, and memory addressing. Common capabilities of the *x86*'s SIMD extensions are described using the terms *x86-SSE* for Streaming SIMD Extensions or *x86-AVX* for Advanced Vector Extensions. When discussing aspects or instructions of a specific SIMD enhancement, the original acronyms (e.g., SSE, SSE2, SSE3, SSSE3, SSSE4, AVX, and AVX2) are used.

Additional Resources

An extensive set of x86-related documentation is available from both Intel and AMD. Appendix C lists a number of resources that both aspiring and experienced x86 assembly language programmers will find useful. Of all the resources listed Appendix C, the most important tome is Volume 2 of the reference manual entitled *Intel 64 and IA-32 Architectures Software Developer's Manual—Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C (Order Number: 325462).* This volume contains comprehensive information for each processor instruction, including detailed operational descriptions, lists of valid operands, affected status flags, and potential exceptions. You are strongly encouraged to consult this documentation when developing your own x86 assembly language functions in order to verify correct instruction usage.