

The Software Optimization Cookbook

High-Performance Recipes for IA-32
Platforms

Second Edition

Richard Gerber

Aart J.C. Bik

Kevin B. Smith

Xinmin Tian

Intel
PRESS

Copyright © 2006 Intel Corporation. All rights reserved.

ISBN 0-9764832-1-1

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4744. Requests to the Publisher for permission should be addressed to the Publisher, Intel Press, Intel Corporation, 2111 NE 25 Avenue, JF3-330, Hillsboro, OR 97124-5961. E-Mail: intelpress@intel.com.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in professional services. If professional advice or other expert assistance is required, the services of a competent professional person should be sought.

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Intel may make changes to specifications, product descriptions, and plans at any time, without notice.

Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel, Intel logo, Intel386, Intel486, Intel NetBurst, Intel XScale, Itanium, MMX, Pentium, VTune, and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

† Other names and brands may be claimed as the property of others.

This book is printed on acid-free paper. (∞)

Publisher: Richard Bowles

Editor: David B. Spencer

Content Architect: Stuart Goldstein

Text Design & Composition: IMS Pubs

Graphic Art: IMS Pubs (illustrations), Ted Cyrek (cover)

Library of Congress Cataloging in Publication Data:

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

First printing, March 2006

Contents

Preface xi

Part I Performance Tools and Concepts 1

Chapter 1 Introduction 3

Software Optimization 4

Software Optimization Pitfalls 4

The Software Optimization Process 6

Chapter 2 The Benchmark 9

The Attributes of the Benchmark 10

Repeatable (Required) 10

Representative (Required) 11

Easy to Run (Required) 11

Verifiable (Required) 11

Measure Elapsed Time (Optional) 12

Complete Coverage (Situation-dependent) 12

Precision (Situation-dependent) 12

Benchmark Examples 13

Chapter 3 Performance Tools 19

- Timing Mechanisms 19
- Optimizing Compilers 21
 - Using the Intel® C++ and Fortran Compilers 22
 - Optimizing for Specific Processors 23
 - Writing Functions Specific to One Processor 25
 - Other Compiler Optimizations 26
- Types of Software Profilers 27
- Performance Monitor 28
- VTune™ Performance Analyzer 29
 - Sampling 29
 - Call Graph Profiling 31
- Intel Compiler Codecov Profiler 33
- Microsoft Visual C++ Profiler 34
- Sampling Versus Instrumentation 36
- Trial and Error, Common Sense, and Patience 37

Chapter 4 The Hotspot 41

- What Causes Hotspots and Cold-spots? 43
- More Than Just Time 44
- Uniform Execution and No Hotspots 46

Chapter 5 Processor Architecture 51

- Functional Blocks 52
 - Two Cheeseburgers Please! 54
 - Instruction Fetch and Decode 57
 - Instruction Execution 59
 - Retirement 62
- Registers and Memory 63

Part II Performance Issues 67

Chapter 6 Algorithms 69

- Computational Complexity 69
- Choice of Instructions 70
- Data Dependencies and Instruction Parallelism 75
- Memory Requirements 79
- Parallel Algorithms 79
- Generality of Algorithms 80
- Detecting Algorithm Issues 81

Chapter 7 Branching 89

- Finding the Critical Mis-predicted Branches 92
 - Step 1: Find the Mis-predicted Branches 92
 - Step 2: Find the Time-consuming Hotspots 93
 - Step 3: Determine the Percentage of Mis-predicted Branches 94
 - Final Sanity Check 95
- The Different Types of Branches 96
- Making Branches More Predictable 99
- Removing Branches with CMOV 100
- Removing Branches with Masks 101
- Removing Branches with Min/Max Instructions 103
- Removing Branches by Doing Extra Work 103

Chapter 8 Memory 107

- Memory Overview 108
 - Main Memory and Virtual Memory 109
 - Processor Caches 109
 - Cache Details 112
 - Hardware Prefetch 114
 - Software Prefetch 114
 - Writing Data Without the Cache: Non-temporal Writes 115
- Issues Affecting Memory Performance 117
 - Cache Compulsory Loads 117
 - Cache Capacity Loads 118
 - Cache Conflict Loads 119
 - Cache Efficiency 120

Store Forwarding	120
Data Alignment	122
Compilers and Data Alignment	123
Software Prefetch	124
Detecting Memory Issues	125
Finding Page Misses	126
Finding Store Forwarding Problems	130
Finding L1 Cache Misses	131
Understanding Potential Improvement	133
Fixing Memory Problems	135

Chapter 9 Loops 143

Data Dependences	144
Loop Distribution and Fusion	146
Loop Peeling	149
Loop Unrolling and Re-rolling	149
Loop Interchanging	153
Loop Invariant Computations	155
Loop Invariant Branches	156
Loop Invariant Results	157

Chapter 10 Slow Operations 159

Slow Instructions	159
Lookup Tables	161
System Calls	164
System Idle Process	168

Chapter 11 Floating Point 175

Numeric Exceptions	176
Flush-to-Zero and Denormals are Zero	179
Precision	180
Packed and Scalar Mode	184
Float-to-Integer Conversions, Rounding	185
Floor and Ceil Functions	186
Floating-point Manipulation Tricks	186
FP-to-Integer Conversion	186
Square Root	187
Reciprocal Square Root	187

Chapter 12 SIMD Technology 191

- An Introduction to SIMD Technology 192
 - MMX™ Technology 192
 - Streaming SIMD Extensions 193
- Using SIMD Technology 194
 - Automatic Vectorization 195
 - C++ Class Libraries 196
 - Intrinsics 197
 - Inline Assembly Language 198
 - Advantages and Disadvantages of the Four Methods 199
- SIMD Technology Considerations 200
 - Determining Where to Use SIMD Technology 201
 - Memory Alignment 201
 - Data Layout 203
 - Selecting an Appropriate Packed Data Type 205
 - Compatibility of SIMD and x87 FPU Calculations 207

Chapter 13 Automatic Vectorization 211

- Compiler Switches for Vectorization 211
 - Commonly Used Compiler Switches 211
 - Compiler Switches Example 214
- Compiler Hints for Vectorization 215
 - Commonly Used Compiler Hints 215
 - Compiler Hints Examples 220
- Vectorization Guidelines 222
 - Design and Implementation Considerations 222
 - Usage of Vectorization Diagnostics 224
 - Minimize Potential Aliasing and Side Effects 228
 - Programming Style 232
 - Target Architectures 233

Chapter 14 Processor-Specific Optimizations 239

- 32-bit Intel® Architectures 239
- The Pentium® M Processor 242
- L1 Instruction Cache 243
- Instruction Decoding 243
- Instruction Latencies 244

- Instruction Set 245
- Floating-Point Control Register 246
- MXCSR Status Register 246
- L1 Data Cache 247
- Memory Prefetch 247
- Processor Events 248
- Partial Register Stalls 248
- Partial Flag Stall 249
- Pause Instruction 250

Chapter 15 Introduction to Multiprocessing 253

- Parallel Programming 254
- Thread Management 256
 - High-level Threading with OpenMP[†] 256
 - Low-level Threading 259
- Threading Goals 260
- Threading Issues 261
- Intel Compilers and Threading Tools 265

Chapter 16 Multithreading with OpenMP[†] 269

- OpenMP[†] Key Elements 269
 - Multithreading Execution Model 274
 - OpenMP[†] Memory Model 276
 - Limitations of OpenMP[†] 280
- Compiling OpenMP[†] Programs 281
- Automatic Parallelization 283
- Multithreading Guidelines 286

Chapter 17 Taskqueuing and Advanced Topics 291

- Taskqueuing—Intel Extension to OpenMP[†] 291
 - Taskqueuing Execution Model 291
 - Taskq and Task Constructs 294
 - Threading the N-Queens Program: A Case Study 297
- Thread-Level Pipeline Parallelism 302
- Exploiting Nested Parallelism 306
- Multi-level Parallelism 311
- Thread Affinity Insight 313
- Understanding Loop Scheduling 315

Part III Design and Application Optimization 321**Chapter 18 Case Study: Threading a Video Codec 323**

- Initial Performance of the H.264 Encoder 324
- Parallelization of the H.264 Encoder 325
 - Task and Data Domain Decomposition 325
 - Slice-Level Parallelism 327
 - Frame-Level Parallelism 328
 - Implementation Using Two Slice Queues 329
 - Implementation Using Task Queuing Model 331
- Performance 333
 - Tradeoff Between Increased Speed and Effective Compression 333
 - Performance on Multiprocessor with HT Technology 335
- Understanding the Performance 336
- Multithreading Overhead 340
- Further Performance Tuning 341
- Summary of Threading 342

Chapter 19 Designing for Performance 345

- Data Movement 346
- Memory and Parallelism 347
- Performance Experiments for Design 348
- Algorithms 349

Chapter 20 Putting it Together: Basic Optimizations 353

- Picking the Low-Hanging Fruit 353
- The Application 354
- Follow Along 356
- The Benchmark 356
- Interpret the Benchmark Results 357
- Improving Float-to-Long Conversions 359
- Parallelizing the Algorithm 360
- Automatic Vectorization to the Rescue 360
- Instruction Level Parallelism with the Ininsics 362
- Summary of Optimizations 363

Chapter 21 Putting It Together: The Last Ten Percent 365

Speed of Light 365

Greater SIMD Efficiency 367

One Final Optimization 370

Summary of Optimizations 371

References 373

Index 379

Preface

The first edition of *The Software Optimization Cookbook* continues to be one of the most popular books offered by Intel Press. Feedback received from readers indicates that the book fills a gap between introductory textbooks that deal with program optimizations in general and advanced manuals that deal with all aspects of the Intel[®] architecture in particular. The introduction of the Intel Extended Memory 64 Technology (Intel EM64T) and multi-core processing together with the growing popularity of the Hyper-Threading Technology, OpenMP[†], and multimedia extensions have outdated the first edition, however. The continuing demand for an intermediate level introduction to these topics has prompted Intel Press to ask three additional Intel experts to team up with the original author to provide an expanded and updated second edition of the book.

The Software Optimization Cookbook, Second Edition, provides updated recipes for high-performance applications on Intel platforms. Through simple explanations and examples, the authors show you how to address performance issues with algorithms, memory access, branch prediction, automatic vectorization, SIMD instructions, multiple threads, and floating-point calculations. Software developers learn how to take advantage of Intel EM64T, multi-core processing, Hyper-Threading Technology, OpenMP, and multimedia extensions. This book guides you through the growing collection of software tools, compiler switches, and coding optimizations, showing you efficient ways to improve the performance of software applications for Intel platforms. Software developers who want to understand the latest techniques for delivering more performance and to fine-tune their coding skills will benefit from this book.

Acknowledgements

The authors would like to thank everyone who has contributed in one way or the other to this book or to the Intel compilers and performance analysis tools. In particular, the authors are grateful for the efforts of the following people: Zia Ansari, Pete Baker, Mitch Bodart, Christopher Bord, Mark Buxton, Ryan Carlson, Yen-Kuang Chen, Joshua Chia, Martyn Corden, Robert Cox, William E. Damon III, Max J. Domeika, Milind Girkar, Koby Gottlieb, Grant Haab, Mohammad Haghighat, Jay Hoeflinger, John Holm, Alexander Isaev, Michael Julier, Wei Li, Christopher Lishka, Diana King, Knud Kirkegaard, David Kreitzer, Tim Mattson, Eric Moore, Dan Macri, Andrey Naraikin, Kannan Narayanan, Clark Nelson, John Ng, Paul Peterson, Michael Ross, Hideki Saito, Sanjiv Shah, Ernesto Su, Sara Sarmiento, William Savage, Dale Schouten, David Sehr, Ronak Singhal, Kevin J. Smith, Stacey Smith, Craig Stoller, Bob Valentine, and Ronen Zohar.

A special thanks to our evaluators, Tim Carver formerly of Intel, Walt Dixon of GE Global Research, Lars Petter Endresen at Scandpower Petroleum Technology AS, James H. Hill of the United States Postal Service, Brian Kennedy at Jeppesen/Boeing, Jitendra Maheshwari at Zeesoft Inc., Kevin Ruland at the University of Kansas Natural History Museum, and Robert van Engelen at Florida State University, for providing valuable feedback on preliminary drafts of the book.

The people of Intel Press have been very helpful during the writing process. In particular, we would like to acknowledge the help of our editor David Spencer and content architect Stuart Goldstein for their kind and professional guidance during all stages of the publishing process.

 **Readers Can Help, Too**

To prepare this second edition, the authoring team carefully reviewed the material in the first edition and updated most of it. In particular, Richard Gerber updated the putting-it-together material. Aart Bik updated Chapters 2, 9, and 12, and he added Chapter 13 on automatic vectorization. Kevin Smith updated Chapter 3 through 8, 10, 11, and 14. Xinmin Tian updated Chapter 15 and added Chapters 16, 17, and 18 on OpenMP parallelization.

Having done so, we are anxious to keep the book up-to-date. We would like to hear your questions and requests for clarification. We post contact information, errata, and additional materials on this book's Web site.

Richard Gerber
Aart J. C. Bik

Kevin B. Smith
Xinmin Tian