大学计算机教育国外著名教材系列（影印版）

PRENTICE HALL PTR

Itanium Architecture for Programmers

Understanding 64-Bit Processors and EPIC Principles

安腾体系结构

理解64位处理器和EPIC原理

James S. Evans
Gregory L. Trimper 著

Pearson Education

清华大学出版社

[illegible]

Itanium Architecture for Programmers

Understanding 64-Bit Processors and EPIC Principles

安腾体系结构

理解64位处理器和EPIC原理

James S. Evans
Gregory L. Trimper 著

清华大学出版社

大学计算机教育国外著名教材系列（影印版）

Itanium Architecture for Programmers

Understanding 64-Bit Processors and EPIC Principles

安 腾 体 系 结 构

理解 64 位处理器和 EPIC 原理

James S. Evans
Lawrence University
Gregory L. Trimper
Viika

清 华 大 学 出 版 社
北　京

图书在版编目(CIP)数据

安腾体系结构：理解 64 位处理器和 EPIC 原理 = Itanium Architecture for Programmers: Understanding 64-Bit Processors and EPIC Principles /（英）埃文斯（Evans, J. S.），（英）特林普尔（Trimper, G. L.）著. —影印本. —北京：清华大学出版社，2004.5
（大学计算机教育国外著名教材系列）
ISBN 7-302-08486-6

Ⅰ. 安…　Ⅱ. ①埃…　②特…　Ⅲ. 微处理器－高等学校－教材－英文　Ⅳ. TP332

中国版本图书馆 CIP 数据核字（2004）第 033621 号

出 版 者：清华大学出版社　　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　　**邮　　编**：100084
社总机：（010）6277 0175　　　**客户服务**：（010）6277 6969
责任编辑：龙啟铭
印 刷 者：北京牛山世兴印刷厂
装 订 者：三河市李旗庄少明装订厂
发 行 者：新华书店总店北京发行所
开　　本：185×230　**印张**：35.5
版　　次：2004 年 5 月第 1 版　2004 年 5 月第 1 次印刷
书　　号：ISBN 7-302-08486-6/TP・6092
印　　数：1~4000
定　　价：49.00 元

"If you've never written assembler before, this book is a good, solid introduction to a part of computer science you really should know.

If you have written assembler before, and want to learn about the Itananium instruction set, the authors take a complicated machine that uses some new ways of doing things, and build up your knowledge in manageable increments.

Having the sample assembler code to study and ponder is very useful—something that's normally considered pretty esoteric becomes much more concrete. And, the code actually does useful things!

As someone who's been writing assembler off and on for over 20 years, it's the book I wish I'd learned from."

—Al Stone, Senior Software Engineer, HP

出版说明

进入 21 世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从 1996 年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书（影印版）”等一系列引进图书，受到国内读者的欢迎和支持。跨入 21 世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高校本科及研究生计算机教育的国外经典教材或著名教材，组成本套“大学计算机教育国外著名教材系列（影印版）”，以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材系列（影印版）”做得更好，更适合高校师生的需要。

清华大学出版社

Preface

This book will assist both computer professionals and college-level learners to comprehend the specific capabilities of the 64-bit Intel® Itanium® architecture, within the wider context of contemporary architectural principles. This is accomplished through a guided investigation of Itanium assembly language, using standard command-line tools and illustrative programs.

The Itanium architecture is distinct from previously available architectures. This explicitly parallel instruction set computer (EPIC) viably introduces a versatile register stack and thorough use of predication. The stature of its two developers (Hewlett-Packard® and Intel) positions the new architecture to win widespread acceptance from technical and financial decision-makers.

Design and development of processor architectures is very expensive. Announced consolidations in the industry have pointed to the phase-out of server and graphical scientific workstation lines based upon three 64-bit RISC designs (Alpha™, MIPS®, and PA-RISC®) that have supported Unix® or Linux® operating systems; those platforms are to be superseded by new product lines based upon Itanium processors.

Except for the briefly marketed 64-bit version of Windows NT® for Alpha-based systems, Microsoft® had not promoted 64-bit development for servers. Moreover, Intel had not marketed a commercial 64-bit platform prior to the Itanium processors. Moore's law implies that a switchover from 32- to 64-bit addressing in the much larger consumer and commercial desktop market is inevitable; indeed, one can install a gigabyte of physical RAM into a high-end notebook computer, thereby using one-quarter of the 32-bit addressing capability.

This book is the second in a line of works to discuss computer architecture and assembly language programming for modern 64-bit processors. We have chosen the Itanium architecture because it represents a thoroughly new approach, and because we anticipate that it will attain wide commercial and educational adoption. In fact, the Itanium processor line should thrive as a platform for Microsoft operating systems, Hewlett-Packard's HP-UX® implementation of Unix, numerous Linux distributions, and even ports of FreeBSD and OpenVMS™.

In writing this book, we have brought forward the collective teaching and practical experience from several preceding works:

Eckhouse, Richard H. and L. Robert Morris, *Minicomputer Systems: Organization, Programming, and Applications (PDP-11).* Englewood Cliffs, N.J.: Prentice Hall, Inc., 1979.

Levy, Henry M. and Richard H. Eckhouse, *Computer Programming and Architecture: The VAX,* 2nd ed. Bedford, Mass.: Digital Press, 1989.

Evans, James S. and Richard H. Eckhouse, *Alpha RISC Architecture for Programmers.* Upper Saddle River, N.J.: Prentice Hall PTR, 1999.

These prior books built a tradition of discussing the general principles of computer architecture through a pedagogically tested experience in register-level analysis and programming, using one specific contemporary architecture each time. In this new book, we continue that tradition by focusing on the Itanium architecture, contrasting it with other designs as appropriate.

We envision a diverse readership for this book. Computer professionals, especially those who want to gain familiarity with 64-bit systems, can use it for individual study and reference. Undergraduate or graduate classes in computer architecture and/or assembly language can use it as the primary text, or advanced classes in computer science may use it as a supplement. We have striven to keep both our discussions and many of the suggested exercises to a degree of transparency that can be worked through with pencil and paper, for we feel that a mature understanding of the complex or the subtle is best built on a foundation of confidence in the simple.

Our book is about the design and capabilities of the Itanium architecture from the programmer's perspective. Hands-on exposure to command-line programming environments is recommended and illustrated to make sample programs come alive for the reader. Therefore we also describe how to work within standard command-line programming environments, principally HP-UX and Linux.

In a course on computer architecture at Lawrence University, we introduce the first simple programming illustrations early in a ten-week term. Additional brief demonstration programs are presented as illustrations during subsequent class meetings, and student assignments frequently involve adaptations and extensions of such models.

These illustrative programs are available as source text on the Web site associated with this book, http://www.viika.com/itanium/. As well as being compatible with the HP-UX and Linux programming environments on Itanium workstations or servers, most of these programs can also be explored using Hewlett-Packard's Ski simulator for Itanium architecture, a free download for 32-bit Linux systems as mentioned in Appendix B in this book.

Increasingly powerful techniques for input and output are introduced throughout the text, beginning with simple debugging techniques and continuing with the use of sequential files for input and output. A symbolic debugger is introduced and illustrated as a versatile tool for routine use.

The chapters of this book contain more material than some instructors may be able to present in their courses, particularly if comparisons among multiple architectures or concepts of hardware organization are also studied. Chapter 8, on floating-point operations, may be omitted without loss of continuity, although it is helpful as background for working through Chapter 11.

We know that some would prefer to put a thorough treatment of procedure calls at an earlier point than we do, but this may be chiefly in order to accomplish input/output. We prefer to use a debugger at first (Chapters 3–6), and introduce procedure calls based on high-level language library routines at the end of Chapter 6. We describe details of procedure-calling mechanisms more fully in Chapter 7, after much of the Itanium instruction set and many fundamental topics—e.g., addressing modes, stacks, and predication—have been discussed. Later, we continue to develop the principles of register-level programming through examples drawing upon some of the functions that the C language supplies for input and output.

We have found that exploring machine code produced by high-level language compilers proves to be remarkably illuminating. Indeed, it is surprising how much overhead may be required by the incorporation of a high degree of proceduralization in programs. Such "Eureka!" experiences become rewards for comprehending assembly language and computer architecture.

Chapters 10 and 11 comprise an introductory unit on optimization techniques, first with a focus on the intrinsic capabilities of the architecture, and then with observations of output from high-level language compilers. We use the techniques of an experimentalist more than the perspective of a theorist when dealing with performance-related concerns. These two chapters represent a distinctive feature of this book.

The final chapters take up additional topics related to Itanium architecture, including "parallel" instructions that can accelerate calculations with data less than 64 bits in width, the provision for executing applications using 32-bit Intel instructions, and an overview of extensions added to computer architectures in later implementations.

Each chapter includes a list of related references, both print and electronic, with the caveat that electronic resources, by their very nature, may vanish without warning.

Exercises vary widely in type (numeric, essay, programming) and degree of difficulty. Answers or hints for many of them are provided at the back of the book.

Several appendices contain material that is useful for setting up a computer system to utilize the programs in the book, reference material on numerous features of the Itanium architecture, a treatment of the macro capabilities of the GNU assembler for Linux, and an introduction to inline assembly.

Acknowledgments

Many professional colleagues, former students, and associates have cumulatively contributed to our understanding of computer architecture and thereby to the making of this book. We appreciate their help and encouragement.

Some of the teaching examples in our book developed out of ideas from Professor Thomas L. Naps (University of Wisconsin–Oshkosh), author or co-author of several computer science texts.

We wish to thank Walter Triebel (formerly of Intel Corporation, now at Fairleigh Dickinson University) for first suggesting a project to prepare this book, Richard Eckhouse (University of Massachusetts–Boston) for hearty endorsement, Barry Levine (School of Biosciences, University of Birmingham) for professional hospitality and enthusiastic encouragement, Joey Lawrance for energetic advocacy of Linux systems, and Fred Bartels for discussions of pedagogy from elementary school through postgr duate levels.

Mike Perman, a regional representative for Prentice-Hall®, kindly provided several reference books used during the project.

We appreciate having had remote access to two Itanium-based systems, running different releases of Linux, through the Test Drive program of Compaq® Corporation managed by Tim Regan.

We especially thank Hewlett-Packard Company (HP) for extensive assistance, without which we could not have proceeded.

Early encouragement from Mark Gibson (HP Laboratories) led to crucially important access arrangements: first, with Brian Lynn for secure remote access to a testbed Itanium-based system at HP Laboratories with both the GNU software development tools and the Intel assembler, C/C++ compiler, and FORTRAN compiler; and second, with Dan Marcek (HP University Affairs) for indirect access to an Itanium workstation with HP-UX software that had been contributed to the laboratory of Andrea Arpaci-Dusseau at the University of Wisconsin–Madison.

Later, Dan Marcek also arranged contact with Rick Hank, who provided tests of certain programs and commands using prerelease HP-UX software. Dan Marcek, Bob Kennedy, and Vicki Niccum ultimately made available an Itanium workstation to Lawrence University in direct support of our book project. Rick Hank ensured that we had all Itanium development software available from HP.

Initial inquiries to Pat Pekary (publisher of HP Books) led to further contacts in several directions: with her colleague Walter Bruce, who became our liaison as regards our book being sponsored in the HP Professional Books series; with Jill Harry and Mark Taub, who became our principal contacts at Prentice Hall PTR and who recommended our project for a publishing arrangement; and with several Hewlett-Packard technical professionals for whom we have developed deep respect and to whom we owe many thanks indeed.

Dale Morris responded to our outlined book proposal with some leads to useful technical information and with detailed comments that reinforced our confidence in the viability of an Itanium book project. Whenever we have had questions or concerns, Dale and other Hewlett-Packard professionals have always responded courteously, promptly, and with interest in the success of our endeavor. We have received graciously offered and very timely help from Eddie Gornish, Sverre Jarp, Peter Markstein, Marcel Moolenaar, and David Mosberger.

Jim Hull (HP Laboratories) ably assisted with a meticulous technical review for the publisher and forwarded copious suggestions to the authors, most of which we took on board. Donna Cullen-Dolce coordinated production work on the book for the publisher and coached the authors through the many steps of the modern publishing process.

Trademarks

No endorsement of any product mentioned in this book is implied by the authors or the publisher. Generally we have used the ™ and ® symbols denoting trademark status in the US only at the earliest occurrence of each trademarked word or phrase in the book.

Macintosh and PowerBook are trademarks of Apple Computer, Inc., registered in the U.S. and other countries.

BBEdit and TextWrangler are trademarks of Bare Bones Software, Inc.

BSD is a registered trademark of Berkeley Software Design, Inc.

Alpha and OpenVMS are trademarks and Compaq is a registered trademark of Compaq Computer Corporation in the US and other countries, which later merged with Hewlett-Packard Company.

Connectix and Connectix Virtual PC are trademarks of Connectix Corporation, many of whose products were acquired by Microsoft Corporation.

CRAY is a registered trademark of Cray, Inc.

Digital, PDP, and VAX are registered trademarks of Digital Equipment Corporation, which was acquired by Compaq Computer Corporation, which later merged with Hewlett-Packard Company.

Hewlett-Packard, HP-UX, and PA-RISC are registered trademarks of Hewlett-Packard Company.

IEEE is a registered trademark of the Institute of Electrical and Electronic Engineers, Inc.

Intel386, Intel486, and MMX are trademarks and Intel, Itanium, and Pentium are registered trademarks of Intel Corporation or its subsidiaries in the US and other countries.

System/36, System/38, and System/360 are trademarks and AS/400, IBM, PowerPC, and ThinkPad are registered trademarks of International Business Machines Corporation.

ISO is a registered trademark of the International Organization for Standardization.

CONTENTS

List of Figures xxiii

List of Tables xxv

Preface xxvii

Acknowledgments xxix

Trademarks xxxiii

Chapter 1 Architecture and Implementation 1

1.1 Analogy: Piano Architecture 2

1.2 Types of Computer Languages 3

1.3 Why Study Assembly Language? 4

1.4 Prefixes for Binary Multiples 5

1.5 Instruction Set Architectures 5

1.6 The Life Cycle of Computer Architectures 6

1.6.1 The 32-Bit Intel® Architecture and Its Predecessors 7

1.6.2 The Alpha™ Architecture and Its Predecessors 7

1.6.3 The Itanium® Architecture and Its Predecessors 9

1.6.4 The Naming of Architectures and Implementations 10

1.7 SQUARES: A First Programming Example 12

1.7.1 C, FORTRAN, and COBOL 12

1.7.2 Assembly Language for Itanium Architecture 14

1.8 Review of Number Systems 16

1.8.1 Positional Coefficients and Weights 16

1.8.2 Binary and Hexadecimal Representations 17

1.8.3 Signed Integers 18

Summary 20

References 20

Exercises 22

Chapter 2 Computer Structures and Data Representations 25

2.1 Computer Structures 26
2.1.1 The Central Processing Unit 26
2.1.2 The Memory 27
2.1.3 The Input/Output System 29
2.2 Instruction Execution 30
2.3 Classes of Instruction Set Architectures 32
2.4 Migration to 64-Bit Architectures 34
2.5 Itanium Information Units and Data Types 35
2.5.1 Integers 37
2.5.2 Floating-Point Numbers 37
2.5.3 Alphanumeric Characters 41
Summary 44
References 45
Exercises 46

Chapter 3 The Program Assembler and Debugger 49

3.1 Programming Environments 50
3.2 Program Development Steps 50
3.3 Comparing Variants of a Source File 54
3.4 Assembler Statement Types 54
3.4.1 Statement Format 55
3.4.2 Symbolic Addresses 56
3.4.3 Classes of Assembly Language Operators 56
3.5 The Functions of a Symbolic Assembler 57
3.5.1 Constants 58
3.5.2 Symbols or Identifiers 58
3.5.3 Storage Allocation 59
3.5.4 The Location Counter 61
3.5.5 Expressions 62
3.5.6 Control Statements 64
3.5.7 Elements of a Listing File 64
3.6 The Assembly Process 66
3.7 The Linking Process 68
3.8 The Program Debugger 69
3.8.1 Capabilities of Debugger Programs 71
3.8.2 Running SQUARES using gdb (Linux® and HP-UX®) 71
3.8.3 Running SQUARES using adb (HP-UX) 73
3.8.4 Examples of Debugger Commands 74

3.9 Conventions for Writing Programs 77
Summary 78
References 79
Exercises 80

Chapter 4 Itanium Instruction Formats and Addressing 83

4.1 Overview of Itanium Instruction Formats 84
4.1.1 Instruction Bundles 85
4.1.2 Instruction Bit-Field Layouts 85
4.1.3 Classes of Itanium Instructions 86
4.2 Integer Arithmetic Instructions 88
4.2.1 Addition and Subtraction 88
4.2.2 Arithmetic Overflow 89
4.2.3 Shift Left and Add Instruction 90
4.2.4 Special-Case Arithmetic Operations 91
4.2.5 Multiplication of 16-Bit Signed Integers 92
4.2.6 Full-Width Multiplication and Division 93
4.3 Bit Encoding for Itanium Instructions 93
4.4 HEXNUM: Using Arithmetic Instructions 96
4.5 Data Access Instructions 98
4.5.1 Itanium Cache Structures 98
4.5.2 Integer Store Instructions 101
4.5.3 Integer Load Instructions 101
4.5.4 Move Long Immediate Instruction 103
4.5.5 Accessing Simple Record Structures 104
4.5.6 Access to Specialized CPU Registers 105
4.6 Other ALU Instructions 105
4.6.1 Sign-Extend Instruction 106
4.6.2 Zero-Extend Instruction 106
4.6.3 Instructions for Quantities Less Than 64 Bits in Width 107
4.7 DOTPROD: Using Data Access Instructions 107
4.8 Itanium Addressing Modes 110
4.8.1 Immediate Addressing 110
4.8.2 Register Direct Addressing 110
4.8.3 Register Indirect Addressing 111
4.8.4 Autoincrement Addressing 111
4.8.5 Summary of Itanium Addressing Modes 111
4.8.6 Addressing Details in Previous Programs 112

4.9 Addressing in Other Architectures 116
4.9.1 Modes Built on Register Indirect Addressing 116
4.9.2 Modes Built on Displacement Addressing 117
4.9.3 Comparison of Modes Across Architectures 117
Summary 118
References 119
Exercises 119

Chapter 5 Comparison, Branches, and Predication 123

5.1 Hardware Basis for Control of Flow 123
5.1.1 Condition Codes 124
5.1.2 State-Management Approaches 125
5.1.3 Predicate Registers 126
5.2 Integer Compare Instructions 126
5.2.1 Signed Comparison and Equality 127
5.2.2 Unsigned Comparison 128
5.3 Program Branching 129
5.3.1 Ordinary Branch Instructions 130
5.3.2 Timing Considerations for Branches 131
5.3.3 If…Then…Else Structures 131
5.3.4 Loop Structures 134
5.3.5 Branch Addressing Range 135
5.3.6 Locality and Program Performance 136
5.4 DOTLOOP: Using a Counted Loop 136
5.5 Stops, Instruction Groups, and Performance 137
5.5.1 Study of Stops and Groups in DOTLOOP 138
5.5.2 Simplified Rules for Data Dependency 140
5.5.3 How Itanium Assemblers Handle Stops 142
5.5.4 Local Labels for Loops 142
5.5.5 Loops, Branches, and Overall Performance 143
5.6 DOTCLOOP: Using the Loop Count Register 143
5.7 Other Structured Programming Constructs 145
5.7.1 Unconditional Compare Instructions 146
5.7.2 Nested If...Then...Else Structures 146
5.7.3 Multiway Branching 147
5.7.4 Simple Case Structures 148
5.8 MAXIMUM: Using Conditional Instructions 150