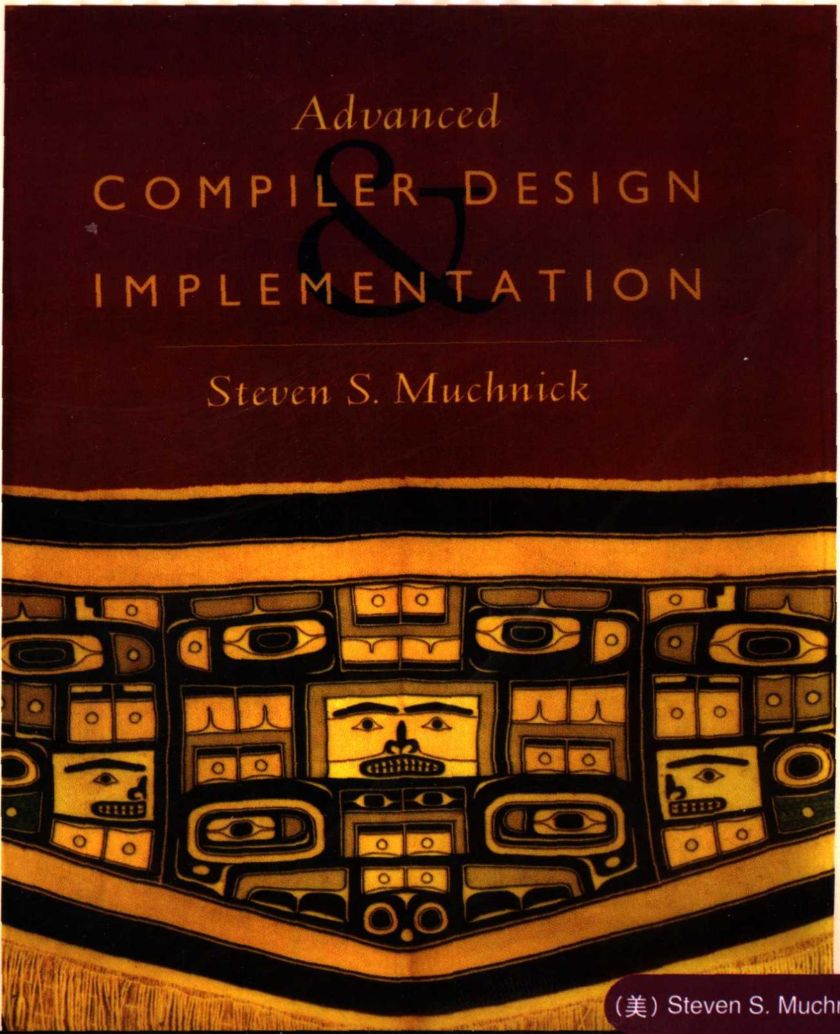
HZ BOOKS
华章教育

经　典　原　版　书　库

高级编译器设计与实现

（英文版）



（美）Steven S. Muchnick 著

机械工业出版社
China Machine Press

ELSEVIER

高级编译器设计与实现（英文版）

Advanced Compiler Design and Implementation

本书是经典的编译器著作，与“龙书”齐名。书中针对现代语言和体系结构全面介绍了编译器设计与实现的高级论题，从编译器的基础领域中的高级问题开始，然后深入讨论了各种重要的代码优化。本书专为编译器专业人士和计算机专业本科生、研究生编写，在设计和实现高度优化的编译器以及确定优化的重要性和实现优化的最有效的方法等方面，为读者提供了非常有价值的指导。

本书的主要特点

- 为理解高级编译器设计的主要问题奠定了基础。
- 深入阐述了优化问题。
- 使用四个商业编译器（包括 Sun 的 SPARC、IBM 的 POWER 和 PowerPC、DEC 的 Alpha 以及 Intel 的 Pentium 和相关处理器上的编译器）的实例分析说明了编译器结构、中间代码设计和优化的各种方法。
- 给出了大量的定义明确的代码生成、优化和其他方面的算法。
- 用 Informal Compiler Algorithm Notation（ICAN）（一种由作者设计的语言）来以清晰、简洁的方式表达算法。

作者简介

Steven S. Muchnick 具有丰富而广博的经验。他曾经是计算机科学教授，后来他将自己的知识和经验应用于编译器设计，成为两种计算机体系结构（惠普的 PA-RISC 和 Sun 的 SPARC）开发团队的核心成员，并担任这些系统的高级编译器设计与实现的领导人。他的研究和开发经验对于指导读者做出编译器设计决策极具价值。

网上购书：www.china-pub.com

ISBN 7-111-12771-4

9 787111 127710

HZ BOOKS
华章图书

北京市西城区百万庄南街 1 号　100037
读者服务热线：(010)68995259, 68995264
读者服务信箱：hzedu@hzbook.com
http://www.hzbook.com

限中国大陆地区销售

ISBN 7-111-12771-4/TP · 2860
定价：80.00 元

经 典 原 版 书 库

高级编译器设计与实现

（英文版）

Advanced Compiler Design and Implementation

(美) Steven S. Muchnick 著

机 械 工 业 出 版 社
China Machine Press

Steven S. Muchnick: Advanced Compiler Design and Implementation (ISBN: 1-55860-320-4).

Original English language edition copyright © 1997 by Academic Press.

All rights reserved.

Authorized English reprints copyright © 2003 by Elsevier (Singapore) Pte Ltd.

本书仅限于中华人民共和国境内（不包括中国香港、澳门特别行政区和中国台湾）销售发行。

版权所有，侵权必究。

本书版权登记号：图字：01-2003-5837

图书在版编目（CIP）数据

高级编译器设计与实现（英文版）/（美）穆可尼克（Muchnick, S. S.）著. –北京：机械工业出版社，2003.9

（经典原版书库）

书名原文：Advanced Compiler Design and Implementation

ISBN 7-111-12771-4

Ⅰ. 高…　Ⅱ. 穆…　Ⅲ. 编译程序–程序设计–英文　Ⅳ. TP314

中国版本图书馆CIP数据核字（2003）第068441号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：迟振春

北京瑞德印刷有限公司印刷·新华书店北京发行所发行

2003年9月第1版第1次印刷

787mm × 1092mm 1/16 · 55.5印张

印数：0 001- 3 000册

定价：80.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国

试读结束，需要全本PDF请购买 www.ertongbook.com

家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzedu@hzbook.com
联系电话：（010）68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范　明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅　宏	程　旭	程时端	谢希仁
裘宗燕	戴　葵			

Contents

Foreword by Susan Graham vii

Preface x

1 Introduction to Advanced Topics 1

1.1 Review of Compiler Structure 1
1.2 Advanced Issues in Elementary Topics 3
1.3 The Importance of Code Optimization 6
1.4 Structure of Optimizing Compilers 7
1.5 Placement of Optimizations in Aggressive Optimizing Compilers 11
1.6 Reading Flow Among the Chapters 14
1.7 Related Topics Not Covered in This Text 16
1.8 Target Machines Used in Examples 16
1.9 Number Notations and Data Sizes 16
1.10 Wrap-Up 17
1.11 Further Reading 18
1.12 Exercises 18

2 Informal Compiler Algorithm Notation (ICAN) 19

2.1 Extended Backus-Naur Form Syntax Notation 19
2.2 Introduction to ICAN 20
2.3 A Quick Overview of ICAN 23
2.4 Whole Programs 25
2.5 Type Definitions 25
2.6 Declarations 26

2.7 Data Types and Expressions 27
2.8 Statements 36
2.9 Wrap-Up 41
2.10 Further Reading 41
2.11 Exercises 41

3 Symbol-Table Structure 43
3.1 Storage Classes, Visibility, and Lifetimes 43
3.2 Symbol Attributes and Symbol-Table Entries 45
3.3 Local Symbol-Table Management 47
3.4 Global Symbol-Table Structure 49
3.5 Storage Binding and Symbolic Registers 54
3.6 Approaches to Generating Loads and Stores 59
3.7 Wrap-Up 64
3.8 Further Reading 64
3.9 Exercises 64

4 Intermediate Representations 67
4.1 Issues in Designing an Intermediate Language 67
4.2 High-Level Intermediate Languages 69
4.3 Medium-Level Intermediate Languages 71
4.4 Low-Level Intermediate Languages 71
4.5 Multi-Level Intermediate Languages 72
4.6 Our Intermediate Languages: MIR, HIR, and LIR 73
4.7 Representing MIR, HIR, and LIR in ICAN 81
4.8 ICAN Naming of Data Structures and Routines that Manipulate Intermediate Code 92
4.9 Other Intermediate-Language Forms 96
4.10 Wrap-Up 101
4.11 Further Reading 102
4.12 Exercises 102

5 Run-Time Support 105
5.1 Data Representations and Instructions 106
5.2 Register Usage 109

5.3 The Local Stack Frame 111
5.4 The Run-Time Stack 114
5.5 Parameter-Passing Disciplines 116
5.6 Procedure Prologues, Epilogues, Calls, and Returns 119
5.7 Code Sharing and Position-Independent Code 127
5.8 Symbolic and Polymorphic Language Support 131
5.9 Wrap-Up 133
5.10 Further Reading 134
5.11 Exercises 135

6 Producing Code Generators Automatically 137

6.1 Introduction to Automatic Generation of Code Generators 138
6.2 A Syntax-Directed Technique 139
6.3 Introduction to Semantics-Directed Parsing 159
6.4 Tree Pattern Matching and Dynamic Programming 160
6.5 Wrap-Up 165
6.6 Further Reading 166
6.7 Exercises 166

7 Control-Flow Analysis 169

7.1 Approaches to Control-Flow Analysis 172
7.2 Depth-First Search, Preorder Traversal, Postorder Traversal, and Breadth-First Search 177
7.3 Dominators and Postdominators 181
7.4 Loops and Strongly Connected Components 191
7.5 Reducibility 196
7.6 Interval Analysis and Control Trees 197
7.7 Structural Analysis 202
7.8 Wrap-Up 214
7.9 Further Reading 214
7.10 Exercises 215

8 Data-Flow Analysis 217

8.1 An Example: Reaching Definitions 218
8.2 Basic Concepts: Lattices, Flow Functions, and Fixed Points 223

8.3 Taxonomy of Data-Flow Problems and Solution Methods 228
8.4 Iterative Data-Flow Analysis 231
8.5 Lattices of Flow Functions 235
8.6 Control-Tree-Based Data-Flow Analysis 236
8.7 Structural Analysis 236
8.8 Interval Analysis 249
8.9 Other Approaches 250
8.10 Du-Chains, Ud-Chains, and Webs 251
8.11 Static Single-Assignment (SSA) Form 252
8.12 Dealing with Arrays, Structures, and Pointers 258
8.13 Automating Construction of Data-Flow Analyzers 259
8.14 More Ambitious Analyses 261
8.15 Wrap-Up 263
8.16 Further Reading 264
8.17 Exercises 265

9 Dependence Analysis and Dependence Graphs 267

9.1 Dependence Relations 267
9.2 Basic-Block Dependence DAGs 269
9.3 Dependences in Loops 274
9.4 Dependence Testing 279
9.5 Program-Dependence Graphs 284
9.6 Dependences Between Dynamically Allocated Objects 286
9.7 Wrap-Up 288
9.8 Further Reading 289
9.9 Exercises 290

10 Alias Analysis 293

10.1 Aliases in Various Real Programming Languages 297
10.2 The Alias Gatherer 302
10.3 The Alias Propagator 307
10.4 Wrap-Up 314
10.5 Further Reading 315
10.6 Exercises 316

试读结束，需要全本PDF请购买 www.ertongbook.com

11 Introduction to Optimization 319

11.1 Global Optimizations Discussed in Chapters 12 Through 18 321
11.2 Flow Sensitivity and May vs. Must Information 323
11.3 Importance of Individual Optimizations 323
11.4 Order and Repetition of Optimizations 325
11.5 Further Reading 328
11.6 Exercises 328

12 Early Optimizations 329

12.1 Constant-Expression Evaluation (Constant Folding) 329
12.2 Scalar Replacement of Aggregates 331
12.3 Algebraic Simplifications and Reassociation 333
12.4 Value Numbering 343
12.5 Copy Propagation 356
12.6 Sparse Conditional Constant Propagation 362
12.7 Wrap-Up 371
12.8 Further Reading 373
12.9 Exercises 374

13 Redundancy Elimination 377

13.1 Common-Subexpression Elimination 378
13.2 Loop-Invariant Code Motion 397
13.3 Partial-Redundancy Elimination 407
13.4 Redundancy Elimination and Reassociation 415
13.5 Code Hoisting 417
13.6 Wrap-Up 420
13.7 Further Reading 422
13.8 Exercises 422

14 Loop Optimizations 425

14.1 Induction-Variable Optimizations 425
14.2 Unnecessary Bounds-Checking Elimination 454
14.3 Wrap-Up 457
14.4 Further Reading 459
14.5 Exercises 460

15 Procedure Optimizations 461

15.1 Tail-Call Optimization and Tail-Recursion Elimination 461
15.2 Procedure Integration 465
15.3 In-Line Expansion 470
15.4 Leaf-Routine Optimization and Shrink Wrapping 472
15.5 Wrap-Up 476
15.6 Further Reading 478
15.7 Exercises 478

16 Register Allocation 481

16.1 Register Allocation and Assignment 482
16.2 Local Methods 483
16.3 Graph Coloring 485
16.4 Priority-Based Graph Coloring 524
16.5 Other Approaches to Register Allocation 525
16.6 Wrap-Up 526
16.7 Further Reading 528
16.8 Exercises 529

17 Code Scheduling 531

17.1 Instruction Scheduling 532
17.2 Speculative Loads and Boosting 547
17.3 Speculative Scheduling 548
17.4 Software Pipelining 548
17.5 Trace Scheduling 569
17.6 Percolation Scheduling 571
17.7 Wrap-Up 573
17.8 Further Reading 575
17.9 Exercises 576

18 Control-Flow and Low-Level Optimizations 579

18.1 Unreachable-Code Elimination 580
18.2 Straightening 583
18.3 If Simplifications 585
18.4 Loop Simplifications 586

18.5 Loop Inversion 587
18.6 Unswitching 588
18.7 Branch Optimizations 589
18.8 Tail Merging or Cross Jumping 590
18.9 Conditional Moves 591
18.10 Dead-Code Elimination 592
18.11 Branch Prediction 597
18.12 Machine Idioms and Instruction Combining 599
18.13 Wrap-Up 602
18.14 Further Reading 604
18.15 Exercises 605

19 Interprocedural Analysis and Optimization 607

19.1 Interprocedural Control-Flow Analysis: The Call Graph 609
19.2 Interprocedural Data-Flow Analysis 619
19.3 Interprocedural Constant Propagation 637
19.4 Interprocedural Alias Analysis 641
19.5 Interprocedural Optimizations 656
19.6 Interprocedural Register Allocation 659
19.7 Aggregation of Global References 663
19.8 Other Issues in Interprocedural Program Management 663
19.9 Wrap-Up 664
19.10 Further Reading 666
19.11 Exercises 667

20 Optimization for the Memory Hierarchy 669

20.1 Impact of Data and Instruction Caches 670
20.2 Instruction-Cache Optimization 672
20.3 Scalar Replacement of Array Elements 682
20.4 Data-Cache Optimization 687
20.5 Scalar vs. Memory-Oriented Optimizations 700
20.6 Wrap-Up 700
20.7 Further Reading 703
20.8 Exercises 704

21 Case Studies of Compilers and Future Trends 705

21.1 The Sun Compilers for SPARC 707
21.2 The IBM XL Compilers for the POWER and PowerPC Architectures 716
21.3 Digital Equipment's Compilers for Alpha 726
21.4 The Intel Reference Compilers for the Intel 386 Architecture Family 734
21.5 Wrap-Up 744
21.6 Future Trends in Compiler Design and Implementation 745
21.7 Further Reading 746

App. A Guide to Assembly Languages Used in This Book 747

A.1 Sun SPARC Versions 8 and 9 Assembly Language 747
A.2 IBM POWER and PowerPC Assembly Language 749
A.3 DEC Alpha Assembly Language 750
A.4 Intel 386 Architecture Assembly Language 752
A.5 Hewlett-Packard's PA-RISC Assembly Language 753

App. B Representation of Sets, Sequences, Trees, DAGs, and Functions 757

B.1 Representation of Sets 759
B.2 Representation of Sequences 763
B.3 Representation of Trees and DAGs 763
B.4 Representation of Functions 764
B.5 Further Reading 765

App. C Software Resources 767

C.1 Finding and Accessing Software on the Internet 767
C.2 Machine Simulators 767
C.3 Compilers 768
C.4 Code-Generator Generators: BURG and IBURG 769
C.5 Profiling Tools 770

List of Illustrations 773

List of Tables 797

Bibliography 801

Technical Index of Mathematical Formulas and ICAN Procedures and Major Data Structures 821

Subject Index 827

Introduction to Advanced Topics

We begin by reviewing the structure of compilers and then proceed to lay the groundwork for our exploration of the advanced topics in compiler design and implementation discussed in the remainder of the book. In particular, we first review the basics of compiler structure and then give an overview of the advanced material about symbol-table structure and access, intermediate-code forms, run-time representations, and automatic generation of code generators contained in Chapters 3 through 6. Next we describe the importance of optimization to achieving fast code, possible structures for optimizing compilers, and the organization of optimizations in an aggressive optimizing compiler. Then we discuss the reading flow relationships among the chapters. We conclude with a list of related topics not covered in this book, two short sections on target machines used in examples and notations for numbers and the names we use for various data sizes, and a wrap-up of this chapter, followed by a Further Reading section and exercises, as will be found at the end of all but the last chapter.

1.1 Review of Compiler Structure

Strictly speaking, compilers are software systems that translate programs written in higher-level languages into equivalent programs in object code or machine language for execution on a computer. Thus, a particular compiler might run on an IBM-compatible personal computer and translate programs written in Fortran 77 into Intel 386–architecture object code to be run on a PC. The definition can be widened to include systems that translate from one higher-level language to another, from one machine language to another, from a higher-level language to an intermediate-level form, etc. With the wider definition, we might have, for example, a compiler that runs on an Apple Macintosh and translates Motorola M68000 Macintosh object code to PowerPC object code to be run on a PowerPC-based Macintosh.

试读结束，需要全本PDF请购买 www.ertongbook.com