

《高级编译器设计与实现》

图书基本信息

书名：《高级编译器设计与实现》

13位ISBN编号：9787111127710

10位ISBN编号：7111127714

出版时间：2003-9

出版社：机械工业出版社

作者：穆可尼克 (Steven S. Muchnick)

页数：856

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu000.com

《高级编译器设计与实现》

内容概要

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。

《高级编译器设计与实现》

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

书籍目录

Foreword by Susan Graham	vii
Preface	x
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
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
- 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	105
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

《高级编译器设计与实现》

精彩短评

1、清晰地描述了后端优化的诸多理论，几乎每一章后面都有一个让人蛋疼菊紧的further reading，让你清楚地认识到本书只是一个开始是Roadmap。。

1、《高级编译器设计与实现》的笔记-第3页

"中间代码的设计上在相当的程度上是一种技巧，而不是科学。有人说有多少种套件，就有多少种中间语言设计，但这种估计可能比实际少了一半左右。事实上，由于许多编译器的使用了两种以上的中间代码....."

的确，如作者所说，编译器可能会选取一种以上的中间语言。如IBM XL C/C++编译器，就有两套，第一套是基于寄存器的XIL，第二套是基于栈的WCode。而现在的IBM编译器前端吐出来的是WCode，后端会通过一个WCode2XIL的Translator翻译为XIL，再交由给后端的其它部分进行处理。

《高级编译器设计与实现》

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:www.tushu000.com