CONTENTS

About the Authors iv
About the Cover viii
Preface xxi

Chapter 1 Overview of Compilation **1**

1.1 Introduction 1
1.2 Why Study Compiler Construction? 3
1.3 The Fundamental Principles of Compilation 4
1.4 Compiler Structure 5
1.5 High-Level View of Translation 8
1.5.1 Understanding the Input 8
1.5.2 Creating and Maintaining the Run-Time Environment 13
1.5.3 Improving the Code 15
1.5.4 Creating the Output Program 16
1.6 Desirable Properties of a Compiler 23
1.7 Summary and Perspective 24

Chapter 2 Scanning **27**

2.1 Introduction 27
2.2 Recognizing Words 29
2.2.1 A Formalism for Recognizers 31
2.2.2 Recognizing More-Complex Words 33
2.2.3 Automating Scanner Construction 35

2.3 Regular Expressions 36
2.3.1 Formalizing the Notation 37
2.3.2 Examples 39
2.3.3 Properties of REs 42
2.4 From Regular Expression to Scanner and Back 44
2.4.1 Nondeterministic Finite Automata 45
2.4.2 Regular Expression to NFA: Thompson's Construction 48
2.4.3 NFA to DFA: The Subset Construction 51
2.4.4 DFA to Minimal DFA: Hopcroft's Algorithm 55
2.4.5 DFA to Regular Expression 59
2.4.6 Using a DFA as a Recognizer 60
2.5 Implementing Scanners 61
2.5.1 Table-Driven Scanners 62
2.5.2 Direct-Coded Scanners 64
2.5.3 Handling Keywords 65
2.5.4 Specifying Actions 65
2.6 Advanced Topics 67
2.7 Summary and Perspective 71

Chapter 3 Parsing 73
3.1 Introduction 73
3.2 Expressing Syntax 74
3.2.1 Context-Free Grammars 75
3.2.2 Constructing Sentences 79
3.2.3 Encoding Meaning into Structure 83
3.2.4 Discovering a Specific Derivation 86
3.2.5 Context-Free Grammars versus Regular Expressions 87
3.3 Top-Down Parsing 89
3.3.1 Example 90
3.3.2 Complications in Top-Down Parsing 94
3.3.3 Eliminating Left Recursion 94
3.3.4 Eliminating the Need to Backtrack 97
3.3.5 Top-Down Recursive-Descent Parsers 101
3.4 Bottom-Up Parsing 107
3.4.1 Shift-Reduce Parsing 108

3.4.2 Finding Handles 112
3.4.3 LR(1) Parsers 115
3.5 Building LR(1) Tables 120
3.5.1 LR(1) Items 121
3.5.2 Constructing the Canonical Collection 122
3.5.3 Filling in the Tables 127
3.5.4 Errors in the Table Construction 129
3.6 Practical Issues 133
3.6.1 Error Recovery 134
3.6.2 Unary Operators 135
3.6.3 Handling Context-Sensitive Ambiguity 136
3.6.4 Left versus Right Recursion 138
3.7 Advanced Topics 140
3.7.1 Optimizing a Grammar 141
3.7.2 Reducing the Size of LR(1) Tables 143
3.8 Summary and Perspective 147

Chapter 4 Context-Sensitive Analysis 151
4.1 Introduction 151
4.2 An Introduction to Type Systems 154
4.2.1 The Purpose of Type Systems 154
4.2.2 Components of a Type System 160
4.3 The Attribute-Grammar Framework 171
4.3.1 Evaluation Methods 175
4.3.2 Circularity 176
4.3.3 Extended Examples 177
4.3.4 Problems with the Attribute-Grammar Approach 185
4.4 Ad Hoc Syntax-Directed Translation 188
4.4.1 Implementing Ad Hoc Syntax-Directed Translation 190
4.4.2 Examples 193
4.5 Advanced Topics 202
4.5.1 Harder Problems in Type Inference 202
4.5.2 Changing Associativity 204
4.6 Summary and Perspective 206

Chapter 5 Intermediate Representations **209**

5.1 Introduction 209
5.2 Taxonomy 210
5.3 Graphical IRs 213
5.3.1 Syntax-Related Trees 213
5.3.2 Graphs 218
5.4 Linear IRs 222
5.4.1 Stack-Machine Code 224
5.4.2 Three-Address Code 224
5.4.3 Representing Linear Codes 225
5.5 Static Single-Assignment Form 228
5.6 Mapping Values to Names 231
5.6.1 Naming Temporary Values 233
5.6.2 Memory Models 235
5.7 Symbol Tables 238
5.7.1 Hash Tables 239
5.7.2 Building a Symbol Table 241
5.7.3 Handling Nested Scopes 242
5.7.4 The Many Uses for Symbol Tables 246
5.8 Summary and Perspective 249

Chapter 6 The Procedure Abstraction **251**

6.1 Introduction 251
6.2 Control Abstraction 254
6.3 Name Spaces 256
6.3.1 Name Spaces of Algol-like Languages 257
6.3.2 Activation Records 262
6.3.3 Name Spaces of Object-Oriented Languages 268
6.4 Communicating Values between Procedures 275
6.4.1 Passing Parameters 275
6.4.2 Returning Values 279
6.5 Establishing Addressability 280
6.5.1 Trivial Base Addresses 280

6.5.2 Local Variables of Other Procedures 281
6.6 Standardized Linkages 286
6.7 Managing Memory 290
6.7.1 Memory Layout 290
6.7.2 Algorithms to Manage the Heap 295
6.7.3 Implicit Deallocation 299
6.8 Summary and Perspective 305

Chapter 7 Code Shape 307
7.1 Introduction 307
7.2 Assigning Storage Locations 309
7.2.1 Laying Out Data Areas 310
7.2.2 Keeping a Value in a Register 310
7.2.3 Machine Idiosyncrasies 312
7.3 Arithmetic Operators 313
7.3.1 Reducing Demand for Registers 315
7.3.2 Accessing Parameter Values 318
7.3.3 Function Calls in an Expression 319
7.3.4 Other Arithmetic Operators 319
7.3.5 Mixed-Type Expressions 320
7.3.6 Assignment as an Operator 321
7.3.7 Commutativity, Associativity, and Number Systems 321
7.4 Boolean and Relational Operators 322
7.4.1 Representations 323
7.4.2 Hardware Support for Relational Operations 328
7.4.3 Choosing a Representation 332
7.5 Storing and Accessing Arrays 333
7.5.1 Referencing a Vector Element 333
7.5.2 Array Storage Layout 335
7.5.3 Referencing an Array Element 337
7.5.4 Range Checking 343
7.6 Character Strings 344
7.6.1 String Representations 344
7.6.2 String Assignment 345
7.6.3 String Concatenation 349
7.6.4 String Length 349

7.7 Structure References 350
7.7.1 Loading and Storing Anonymous Values 351
7.7.2 Understanding Structure Layouts 352
7.7.3 Arrays of Structures 353
7.7.4 Unions and Run-Time Tags 354
7.8 Control-Flow Constructs 355
7.8.1 Conditional Execution 356
7.8.2 Loops and Iteration 359
7.8.3 Case Statements 364
7.8.4 Break Statements 368
7.9 Procedure Calls 368
7.9.1 Evaluating Actual Parameters 369
7.9.2 Procedure-Valued Parameters 370
7.9.3 Saving and Restoring Registers 370
7.9.4 Optimizations for Leaf Procedures 372
7.10 Implementing Object-Oriented Languages 373
7.10.1 Single Class, No Inheritance 373
7.10.2 Single Inheritance 375
7.11 Summary and Perspective 381

Chapter 8 Introduction to Code Optimization 383
8.1 Introduction 383
8.2 Background 386
8.2.1 An Example from LINPACK 387
8.2.2 Considerations for Optimization 388
8.2.3 Opportunities for Optimization 392
8.3 Redundant Expressions 393
8.3.1 Building a Directed Acyclic Graph 394
8.3.2 Value Numbering 398
8.3.3 Lessons from Redundancy Elimination 403
8.4 Scope of Optimization 404
8.4.1 Local Methods 404
8.4.2 Superlocal Methods 405
8.4.3 Regional Methods 405
8.4.4 Global Methods 407
8.4.5 Whole-Program Methods 407

8.5 Value Numbering Over Regions Larger Than Basic Blocks 408
8.5.1 Superlocal Value Numbering 408
8.5.2 Dominator-Based Value Numbering 413
8.6 Global Redundancy Elimination 417
8.6.1 Computing Available Expressions 419
8.6.2 Replacing Redundant Computations 421
8.6.3 Putting It Together 423
8.7 Advanced Topics 424
8.7.1 Cloning to Increase Context 425
8.7.2 Inline Substitution 427
8.8 Summary and Perspective 430

Chapter 9 Data-Flow Analysis 433

9.1 Introduction 433
9.2 Iterative Data-Flow Analysis 435
9.2.1 Live Variables 435
9.2.2 Properties of the Iterative LIVEOUT Solver 444
9.2.3 Limitations on Data-Flow Analysis 447
9.2.4 Other Data-Flow Problems 450
9.3 Static Single-Assignment Form 454
9.3.1 A Simple Method for Building SSA Form 456
9.3.2 Dominance 457
9.3.3 Placing ϕ-Functions 463
9.3.4 Renaming 466
9.3.5 Reconstructing Executable Code from SSA Form 474
9.4 Advanced Topics 479
9.4.1 Structural Data-Flow Algorithms and Reducibility 480
9.4.2 Interprocedural Analysis 483
9.5 Summary and Perspective 488

Chapter 10 Scalar Optimizations 491

10.1 Introduction 491
10.2 A Taxonomy for Transformations 494

10.2.1 Machine-Independent Transformations 494
10.2.2 Machine-Dependent Transformations 496
10.3 Example Optimizations 498
10.3.1 Eliminating Useless and Unreachable Code 498
10.3.2 Code Motion 505
10.3.3 Specialization 515
10.3.4 Enabling Other Transformations 518
10.3.5 Redundancy Elimination 523
10.4 Advanced Topics 523
10.4.1 Combining Optimizations 523
10.4.2 Strength Reduction 527
10.4.3 Other Objectives for Optimization 538
10.4.4 Choosing an Optimization Sequence 540
10.5 Summary and Perspective 542

Chapter 11 Instruction Selection 545

11.1 Introduction 545
11.1.1 Building Retargetable Compilers 548
11.2 A Simple Tree-Walk Scheme 552
11.3 Instruction Selection via Tree-Pattern Matching 558
11.3.1 Rewrite Rules 559
11.3.2 Finding a Tiling 565
11.3.3 Tools 568
11.4 Instruction Selection via Peephole Optimization 569
11.4.1 Peephole Optimization 570
11.4.2 Peephole Transformers 578
11.5 Advanced Topics 580
11.5.1 Learning Peephole Patterns 580
11.5.2 Generating Instruction Sequences 581
11.6 Summary and Perspective 582

Chapter 12 Instruction Scheduling 585

12.1 Introduction 585

12.2 The Instruction-Scheduling Problem 587
12.2.1 Other Measures of Schedule Quality 593
12.2.2 What Makes Scheduling Hard? 593
12.3 List Scheduling 595
12.3.1 Efficiency Concerns 598
12.3.2 Other Priority Schemes 599
12.3.3 Forward versus Backward List Scheduling 600
12.3.4 Why Use List Scheduling? 603
12.4 Advanced Topics 605
12.4.1 Regional Scheduling 605
12.4.2 Cloning for Context 614
12.5 Summary and Perspective 617

Chapter 13 Register Allocation 619
13.1 Introduction 619
13.2 Background Issues 620
13.2.1 Memory versus Registers 621
13.2.2 Allocation versus Assignment 622
13.2.3 Register Classes 623
13.3 Local Register Allocation and Assignment 624
13.3.1 Top-Down Local Register Allocation 625
13.3.2 Bottom-Up Local Register Allocation 626
13.4 Moving Beyond Single Blocks 629
13.4.1 Liveness and Live Ranges 630
13.4.2 Complications at Block Boundaries 631
13.5 Global Register Allocation and Assignment 633
13.5.1 Discovering Global Live Ranges 635
13.5.2 Estimating Global Spill Costs 637
13.5.3 Interferences and the Interference Graph 639
13.5.4 Top-Down Coloring 642
13.5.5 Bottom-Up Coloring 644
13.5.6 Coalescing Live Ranges to Reduce Degree 646
13.5.7 Review and Comparison 648
13.5.8 Encoding Machine Constraints in the

Interference Graph 649
13.6 Advanced Topics 651
13.6.1 Variations on Graph-Coloring Allocation 651
13.6.2 Harder Problems in Register Allocation 654
13.7 Summary and Perspective 657

Appendix A ILOC **659**
A.1 Introduction 659
A.2 Naming Conventions 662
A.3 Individual Operations 662
A.3.1 Arithmetic 662
A.3.2 Shifts 663
A.3.3 Memory Operations 664
A.3.4 Register-to-Register Copy Operations 665
A.4 An Example 666
A.5 Control-Flow Operations 667
A.5.1 Alternate Comparison and Branch Syntax 668
A.5.2 Jumps 669
A.6 Representing SSA Form 670

Appendix B Data Structures **673**
B.1 Introduction 673
B.2 Representing Sets 674
B.2.1 Representing Sets as Ordered Lists 675
B.2.2 Representing Sets as Bit Vectors 677
B.2.3 Representing Sparse Sets 678
B.3 Implementing Intermediate Representations 679
B.3.1 Graphical Intermediate Representations 679
B.3.2 Linear Intermediate Forms 684
B.4 Implementing Hash Tables 686
B.4.1 Choosing a Hash Function 688

B.4.2 Open Hashing 689
B.4.3 Open Addressing 691
B.4.4 Storing Symbol Records 693
B.4.5 Adding Nested Lexical Scopes 694
B.5 A Flexible Symbol-Table Design 698

Bibliograpy 703
Exercises 725
Index 779