Table of Contents

1 Introduction 1
1.1 Language Processors 1
1.2 The Structure of a Compiler 4
1.3 The Evolution of Programming Languages 12
1.4 The Science of Building a Compiler 15
1.5 Applications of Compiler Technology 17
1.6 Programming Language Basics 25
1.7 Summary of Chapter 1 36
1.8 References for Chapter 1 38

2 A Simple Syntax-Directed Translator 39
2.1 Introduction 40
2.2 Syntax Definition 42
2.3 Syntax-Directed Translation 52
2.4 Parsing 60
2.5 A Translator for Simple Expressions 68
2.6 Lexical Analysis 76
2.7 Symbol Tables 85
2.8 Intermediate Code Generation 91
2.9 Summary of Chapter 2 105

3 Lexical Analysis 109
3.1 The Role of the Lexical Analyzer 109
3.2 Input Buffering 115
3.3 Specification of Tokens 116
3.4 Recognition of Tokens 128
3.5 The Lexical-Analyzer Generator Lex 140
3.6 Finite Automata 147
3.7 From Regular Expressions to Automata 152
3.8 Design of a Lexical-Analyzer Generator 166
3.9 Optimization of DFA-Based Pattern Matchers 173
3.10 Summary of Chapter 3 187
3.11 References for Chapter 3 189

4 Syntax Analysis 191
4.1 Introduction 192
4.2 Context-Free Grammars 197
4.3 Writing a Grammar 209
4.4 Top-Down Parsing 217
4.5 Bottom-Up Parsing 233
4.6 Introduction to LR Parsing: Simple LR 241
4.7 More Powerful LR Parsers 259
4.8 Using Ambiguous Grammars 278
4.9 Parser Generators 287
4.10 Summary of Chapter 4 297
4.11 References for Chapter 4 300

5 Syntax-Directed Translation 303
5.1 Syntax-Directed Definitions 304
5.2 Evaluation Orders for SDD's 310
5.3 Applications of Syntax-Directed Translation 318
5.4 Syntax-Directed Translation Schemes 324
5.5 Implementing L-Attributed SDD's 337
5.6 Summary of Chapter 5 353
5.7 References for Chapter 5 354

6 Intermediate-Code Generation 357
6.1 Variants of Syntax Trees 358
6.2 Three-Address Code 363
6.3 Types and Declarations 370
6.4 Translation of Expressions 378
6.5 Type Checking 386
6.6 Control Flow 399
6.7 Backpatching 410
6.8 Switch-Statements 418
6.9 Intermediate Code for Procedures 422
6.10 Summary of Chapter 6 424
6.11 References for Chapter 6 425

7 Run-Time Environments 427
7.1 Storage Organization 427
7.2 Stack Allocation of Space 430
7.3 Access to Nonlocal Data on the Stack 441
7.4 Heap Management 452
7.5 Introduction to Garbage Collection 463
7.6 Introduction to Trace-Based Collection 470
7.7 Short-Pause Garbage Collection 483
7.8 Advanced Topics in Garbage Collection 494
7.9 Summary of Chapter 7 500
7.10 References for Chapter 7 502

8 Code Generation 505
8.1 Issues in the Design of a Code Generator 506
8.2 The Target Language 512
8.3 Addresses in the Target Code 518
8.4 Basic Blocks and Flow Graphs 525
8.5 Optimization of Basic Blocks 533
8.6 A Simple Code Generator 542
8.7 Peephole Optimization 549
8.8 Register Allocation and Assignment 553
8.9 Instruction Selection by Tree Rewriting 558
8.10 Optimal Code Generation for Expressions 567
8.11 Dynamic Programming Code-Generation 573
8.12 Summary of Chapter 8 578
8.13 References for Chapter 8 579

9 Machine-Independent Optimizations 583
9.1 The Principal Sources of Optimization 584
9.2 Introduction to Data-Flow Analysis 597
9.3 Foundations of Data-Flow Analysis 618
9.4 Constant Propagation 632
9.5 Partial-Redundancy Elimination 639
9.6 Loops in Flow Graphs 655
9.7 Region-Based Analysis 672
9.8 Symbolic Analysis 686
9.9 Summary of Chapter 9 700
9.10 References for Chapter 9 703

10 Instruction-Level Parallelism 707
10.1 Processor Architectures 708
10.2 Code-Scheduling Constraints 710
10.3 Basic-Block Scheduling 721
10.4 Global Code Scheduling 727
10.5 Software Pipelining 738
10.6 Summary of Chapter 10 765
10.7 References for Chapter 10 766

11 Optimizing for Parallelism and Locality 769
11.1 Basic Concepts 771
11.2 Matrix Multiply: An In-Depth Example 782
11.3 Iteration Spaces 788
11.4 Affine Array Indexes 801
11.5 Data Reuse 804
11.6 Array Data-Dependence Analysis 815
11.7 Finding Synchronization-Free Parallelism 828
11.8 Synchronization Between Parallel Loops 853
11.9 Pipelining 861
11.10 Locality Optimizations 884
11.11 Other Uses of Affine Transforms 893
11.12 Summary of Chapter 11 897
11.13 References for Chapter 11 899

12 Interprocedural Analysis Not in this edition
12.1 Basic Concepts -
12.2 Why Interprocedural Analysis? -
12.3 A Logical Representation of Data Flow -
12.4 A Simple Pointer-Analysis Algorithm -
12.5 Context-Insensitive Interprocedural Analysis -
12.6 Context-Sensitive Pointer Analysis -
12.7 Datalog Implementation by BDD's -
12.8 Summary of Chapter 12 -
12.9 References for Chapter 12 -

A A Complete Front End 903
A.1 The Source Language 903
A.2 Main 904
A.3 Lexical Analyzer 905
A.4 Symbol Tables and Types 908
A.5 Intermediate Code for Expressions 909
A.6 Jumping Code for Boolean Expressions 912
A.7 Intermediate Code for Statements 916
A.8 Parser 919
A.9 Creating the Front End 924

B Finding Linearly Independent Solutions 927
Index 931