Contents

1 INTRODUCTION 1

1.1 Why Compilers? A Brief History 2
1.2 Programs Related to Compilers 4
1.3 The Translation Process 7
1.4 Major Data Structures in a Compiler 13
1.5 Other Issues in Compiler Structure 14
1.6 Bootstrapping and Porting 18
1.7 The TINY Sample Language and Compiler 22
1.8 C-Minus: A Language for a Compiler Project 26
Exercises 27 Notes and References 29

2 SCANNING 31

2.1 The Scanning Process 32
2.2 Regular Expressions 34
2.3 Finite Automata 47
2.4 From Regular Expressions to DFAs 64
2.5 Implementation of a TINY Scanner 75
2.6 Use of Lex to Generate a Scanner Automatically 81
Exercises 89 Programming Exercises 93
Notes and References 94

3 CONTEXT-FREE GRAMMARS AND PARSING 95

3.1 The Parsing Process 96
3.2 Context-Free Grammars 97
3.3 Parse Trees and Abstract Syntax Trees 106
3.4 Ambiguity 114
3.5 Extended Notations: EBNF and Syntax Diagrams 123
3.6 Formal Properties of Context-Free Languages 128
3.7 Syntax of the TINY Language 133
Exercises 138 Notes and References 142

4 TOP-DOWN PARSING 143

4.1 Top-Down Parsing by Recursive-Descent 144
4.2 LL(1) Parsing 152
4.3 First and Follow Sets 168
4.4 A Recursive-Descent Parser for the TINY Language 180
4.5 Error Recovery in Top-Down Parsers 183
Exercises 189 Programming Exercises 193
Notes and References 196

5 BOTTOM-UP PARSING 197

5.1 Overview of Bottom-Up Parsing 198
5.2 Finite Automata of LR(0) Items and LR(0) Parsing 201
5.3 SLR(1) Parsing 210
5.4 General LR(1) and LALR(1) Parsing 217
5.5 Yacc: An LALR(1) Parser Generator 226
5.6 Generation of a TINY Parser Using Yacc 243
5.7 Error Recovery in Bottom-Up Parsers 245
Exercises 250 Programming Exercises 254
Notes and References 256

6 SEMANTIC ANALYSIS 257

6.1 Attributes and Attribute Grammars 259
6.2 Algorithms for Attribute Computation 270
6.3 The Symbol Table 295
6.4 Data Types and Type Checking 313
6.5 A Semantic Analyzer for the TINY Language 334
Exercises 339 Programming Exercises 342
Notes and References 343

7 RUNTIME ENVIRONMENTS 345

7.1 Memory Organization During Program Execution 346
7.2 Fully Static Runtime Environments 349
7.3 Stack-Based Runtime Environments 352
7.4 Dynamic Memory 373
7.5 Parameter Passing Mechanisms 381
7.6 A Runtime Environment for the TINY Language 386
Exercises 388 Programming Exercises 395
Notes and References 396

8 CODE GENERATION 397

8.1 Intermediate Code and Data Structures for Code Generation 398
8.2 Basic Code Generation Techniques 407
8.3 Code Generation of Data Structure References 416
8.4 Code Generation of Control Statements and Logical Expressions 428
8.5 Code Generation of Procedure and Function Calls 436
8.6 Code Generation in Commercial Compilers: Two Case Studies 443
8.7 TM: A Simple Target Machine 453
8.8 A Code Generator for the TINY Language 459
8.9 A Survey of Code Optimization Techniques 468
8.10 Simple Optimizations for the TINY Code Generator 481
Exercises 484 Programming Exercises 488
Notes and References 489

Appendix A: A COMPILER PROJECT 491

A.1 Lexical Conventions of C− 491
A.2 Syntax and Semantics of C− 492
A.3 Sample Programs in C− 496
A.4 A TINY Machine Runtime Environment for the C− Language 497
A.5 Programming Projects Using C− and TM 500

Appendix B: TINY COMPILER LISTING 502

Appendix C: TINY MACHINE SIMULATOR LISTING 545

Bibliography 558

Index 562