

Contents

1	Introduction	1
1.1	What is a compiler?	1
1.2	The phases of a compiler	2
1.3	Interpreters	3
1.4	Why learn about compilers?	4
1.5	The structure of this book	5
1.6	To the lecturer	5
1.7	Acknowledgements	6
1.8	Permission to use	6
2	Lexical Analysis	7
2.1	Introduction	7
2.2	Regular expressions	8
2.2.1	Shorthands	10
2.2.2	Examples	11
2.3	Nondeterministic finite automata	13
2.4	Converting a regular expression to an NFA	15
2.4.1	Optimisations	16
2.5	Deterministic finite automata	19
2.6	Converting an NFA to a DFA	20
2.6.1	Solving set equations	20
2.6.2	The subset construction	23
2.7	Size versus speed	26
2.8	Minimisation of DFAs	27
2.8.1	Example	28
2.8.2	Dead states	30
2.9	Lexers and lexer generators	31
2.9.1	Lexer generators	36
2.10	Properties of regular languages	37
2.10.1	Relative expressive power	37
2.10.2	Limits to expressive power	39
2.10.3	Closure properties	39
2.11	Further reading	40

Exercises	41
3 Syntax Analysis	47
3.1 Introduction	47
3.2 Context-free grammars	48
3.2.1 How to write context free grammars	49
3.3 Derivation	51
3.3.1 Syntax trees and ambiguity	52
3.4 Operator precedence	56
3.4.1 Rewriting ambiguous expression grammars	57
3.5 Other sources of ambiguity	60
3.6 Syntax analysis	61
3.7 Predictive parsing	61
3.8 <i>Nullable</i> and <i>FIRST</i>	62
3.9 Predictive parsing revisited	65
3.10 <i>FOLLOW</i>	66
3.11 LL(1) parsing	69
3.11.1 Recursive descent	69
3.11.2 Table-driven LL(1) parsing	70
3.11.3 Conflicts	71
3.12 Rewriting a grammar for LL(1) parsing	72
3.12.1 Eliminating left-recursion	72
3.12.2 left-factorisation	74
3.12.3 Construction of LL(1) parsers summarized	75
3.13 SLR parsing	75
3.14 Constructing SLR parse tables	77
3.14.1 Conflicts in SLR parse-tables	82
3.15 Using precedence rules in LR parse tables	83
3.16 Using LR-parser generators	85
3.16.1 Declarations and actions	85
3.16.2 Abstract syntax	86
3.16.3 Conflict handling in parser generators	88
3.17 Properties of context-free languages	90
3.18 Further reading	90
Exercises	91
4 Symbol Tables	97
4.1 Introduction	97
4.2 Symbol tables	98
4.2.1 Implementation of symbol tables	98
4.2.2 Simple persistent symbol tables	99
4.2.3 A simple imperative symbol table	100
4.2.4 Efficiency issues	100

4.2.5	Shared or separate name spaces	100
4.3	Further reading	101
	Exercises	101
5	Type Checking	103
5.1	Introduction	103
5.2	Attributes	103
5.3	A small example language	105
5.4	Environments for type checking	105
5.5	Type-checking expressions	105
5.6	Type checking of function declarations	108
5.7	Type-checking a program	109
5.8	Advanced type checking	109
5.9	Further reading	112
	Exercises	112
6	Intermediate Code Generation	115
6.1	Introduction	115
6.2	Choosing an intermediate language	116
6.3	The intermediate language	117
6.4	Generating code from expressions	119
6.4.1	Examples of translation	121
6.5	Translating statements	123
6.6	Logical operators	126
6.6.1	Sequential logical operators	127
6.7	Advanced control statements	130
6.8	Translating structured data	132
6.8.1	Floating-point values	132
6.8.2	Arrays	132
6.8.3	Strings	137
6.8.4	Records/structs and unions	137
6.9	Translating declarations	138
6.9.1	Example: Simple local declarations	138
6.10	Further reading	139
	Exercises	140
7	Machine-Code Generation	143
7.1	Introduction	143
7.2	Conditional jumps	144
7.3	Constants	145
7.4	Exploiting complex machine-code instructions	145
7.4.1	Two-address instructions	147
7.5	Optimisations	149

7.6	Further reading	150
	Exercises	151
8	Register Allocation	153
8.1	Introduction	153
8.2	Liveness	154
8.3	Liveness analysis	154
8.4	Interference	157
8.5	Register allocation by graph colouring	159
8.6	Spilling	161
8.7	Heuristics	162
8.7.1	Removing redundant moves	164
8.8	Further reading	165
	Exercises	165
9	Function calls	167
9.1	Introduction	167
9.1.1	The call stack	167
9.2	Activation records	168
9.3	Prologues, epilogues and call-sequences	169
9.4	Caller-saves versus callee-saves	170
9.5	Using registers to pass parameters	173
9.6	Interaction with the register allocator	174
9.7	Accessing non-local variables	178
9.7.1	Global variables	178
9.7.2	call-by-reference parameters	179
9.7.3	Nested scopes	180
9.8	Variants	183
9.8.1	Variable-sized frames	183
9.8.2	Variable number of parameters	184
9.8.3	Direction of stack-growth and position of FP	184
9.8.4	Register stacks	185
9.9	Further reading	185
	Exercises	185
10	Bootstrapping a compiler	187
10.1	Introduction	187
10.2	Notation	187
10.3	Compiling compilers	189
10.3.1	Full bootstrap	191
10.4	Further reading	194
	Exercises	194