

Contents

	Preface	xi
1	Preliminaries	1
	1.1 Abstract Syntax Trees	1
	1.2 Grammars	3
	1.3 Pattern Matching	5
	1.4 Recursive Functions	6
	1.5 Interpreters	8
	1.6 Example Compiler: A Partial Evaluator	10
2	Integers and Variables	13
	2.1 The \mathcal{L}_{Var} Language	13
	2.2 The x86_{Int} Assembly Language	16
	2.3 Planning the Trip to x86	21
	2.4 Remove Complex Operands	23
	2.5 Select Instructions	25
	2.6 Assign Homes	26
	2.7 Patch Instructions	27
	2.8 Generate Prelude and Conclusion	27
	2.9 Challenge: Partial Evaluator for \mathcal{L}_{Var}	28
3	Parsing	29
	3.1 Lexical Analysis and Regular Expressions	29
	3.2 Grammars and Parse Trees	31
	3.3 Ambiguous Grammars	33
	3.4 From Parse Trees to Abstract Syntax Trees	34
	3.5 Earley's Algorithm	36
	3.6 The LALR(1) Algorithm	40
	3.7 Further Reading	43
4	Register Allocation	45
	4.1 Registers and Calling Conventions	46
	4.2 Liveness Analysis	49
	4.3 Build the Interference Graph	51

4.4	Graph Coloring via Sudoku	52
4.5	Patch Instructions	58
4.6	Generate Prelude and Conclusion	58
4.7	Challenge: Move Biasing	59
4.8	Further Reading	62
5	Booleans and Conditionals	65
5.1	The \mathcal{L}_{If} Language	66
5.2	Type Checking \mathcal{L}_{If} Programs	66
5.3	The \mathcal{C}_{If} Intermediate Language	72
5.4	The x86_{If} Language	72
5.5	Shrink the \mathcal{L}_{If} Language	75
5.6	Remove Complex Operands	75
5.7	Explicate Control	76
5.8	Select Instructions	82
5.9	Register Allocation	83
5.10	Patch Instructions	84
5.11	Generate Prelude and Conclusion	84
5.12	Challenge: Optimize Blocks and Remove Jumps	85
5.13	Further Reading	88
6	Loops and Dataflow Analysis	91
6.1	The $\mathcal{L}_{\text{While}}$ Language	91
6.2	Cyclic Control Flow and Dataflow Analysis	91
6.3	Remove Complex Operands	96
6.4	Explicate Control	96
6.5	Register Allocation	96
7	Tuples and Garbage Collection	99
7.1	The $\mathcal{L}_{\text{Tuple}}$ Language	99
7.2	Garbage Collection	102
7.3	Expose Allocation	109
7.4	Remove Complex Operands	110
7.5	Explicate Control and the $\mathcal{C}_{\text{Tuple}}$ Language	110
7.6	Select Instructions and the $\text{x86}_{\text{Global}}$ Language	111
7.7	Register Allocation	116
7.8	Generate Prelude and Conclusion	116
7.9	Challenge: Arrays	118
7.10	Further Reading	123
8	Functions	125
8.1	The \mathcal{L}_{Fun} Language	125
8.2	Functions in x86	130
8.3	Shrink \mathcal{L}_{Fun}	133
8.4	Reveal Functions and the $\mathcal{L}_{\text{FunRef}}$ Language	133

8.5	Limit Functions	133
8.6	Remove Complex Operands	134
8.7	Explicate Control and the \mathcal{C}_{Fun} Language	135
8.8	Select Instructions and the $\text{x86}_{\text{callq}^*}^{\text{Def}}$ Language	136
8.9	Register Allocation	138
8.10	Patch Instructions	139
8.11	Generate Prelude and Conclusion	139
8.12	An Example Translation	141
9	Lexically Scoped Functions	143
9.1	The \mathcal{L}_λ Language	145
9.2	Assignment and Lexically Scoped Functions	150
9.3	Uniquify Variables	151
9.4	Assignment Conversion	151
9.5	Closure Conversion	153
9.6	Expose Allocation	156
9.7	Explicate Control and $\mathcal{C}_{\text{Clos}}$	156
9.8	Select Instructions	157
9.9	Challenge: Optimize Closures	158
9.10	Further Reading	160
10	Dynamic Typing	161
10.1	The \mathcal{L}_{Dyn} Language	161
10.2	Representation of Tagged Values	165
10.3	The \mathcal{L}_{Any} Language	166
10.4	Cast Insertion: Compiling \mathcal{L}_{Dyn} to \mathcal{L}_{Any}	170
10.5	Reveal Casts	170
10.6	Assignment Conversion	171
10.7	Closure Conversion	171
10.8	Remove Complex Operands	172
10.9	Explicate Control and \mathcal{C}_{Any}	172
10.10	Select Instructions	172
10.11	Register Allocation for \mathcal{L}_{Any}	174
11	Gradual Typing	177
11.1	Type Checking $\mathcal{L}_?$	177
11.2	Interpreting $\mathcal{L}_{\text{Cast}}$	183
11.3	Overload Resolution	184
11.4	Cast Insertion	185
11.5	Lower Casts	187
11.6	Differentiate Proxies	188
11.7	Reveal Casts	190
11.8	Closure Conversion	191
11.9	Select Instructions	191
11.10	Further Reading	193

12	Generics	195
	12.1 Compiling Generics	201
	12.2 Resolve Instantiation	202
	12.3 Erase Generic Types	202
A	Appendix	207
	A.1 x86 Instruction Set Quick Reference	207
	References	209
	Index	217