

Contents

	Preface	xi
1	Preliminaries	1
	1.1 Abstract Syntax Trees	1
	1.2 Grammars	3
	1.3 Pattern Matching	4
	1.4 Recursive Functions	6
	1.5 Interpreters	6
	1.6 Example Compiler: A Partial Evaluator	9
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	22
	2.4 Uniquify Variables	25
	2.5 Remove Complex Operands	27
	2.6 Explicate Control	28
	2.7 Select Instructions	30
	2.8 Assign Homes	31
	2.9 Patch Instructions	32
	2.10 Generate Prelude and Conclusion	33
	2.11 Challenge: Partial Evaluator for \mathcal{L}_{Var}	33
3	Register Allocation	35
	3.1 Registers and Calling Conventions	36
	3.2 Liveness Analysis	38
	3.3 Build the Interference Graph	42
	3.4 Graph Coloring via Sudoku	44
	3.5 Patch Instructions	49
	3.6 Prelude and Conclusion	50
	3.7 Challenge: Move Biasing	52
	3.8 Further Reading	55

4	Booleans and Conditionals	57
4.1	The \mathcal{L}_{If} Language	58
4.2	Type Checking \mathcal{L}_{If} Programs	59
4.3	The \mathcal{C}_{If} Intermediate Language	64
4.4	The x86_{If} Language	64
4.5	Shrink the \mathcal{L}_{If} Language	66
4.6	Uniquify Variables	67
4.7	Remove Complex Operands	67
4.8	Explicate Control	68
4.9	Select Instructions	74
4.10	Register Allocation	75
4.11	Patch Instructions	77
4.12	Challenge: Optimize Blocks and Remove Jumps	77
4.13	Further Reading	81
5	Loops and Dataflow Analysis	83
5.1	The $\mathcal{L}_{\text{While}}$ Language	84
5.2	Cyclic Control Flow and Dataflow Analysis	85
5.3	Mutable Variables and Remove Complex Operands	89
5.4	Uncover <code>get!</code>	91
5.5	Remove Complex Operands	92
5.6	Explicate Control and \mathcal{C}_{\circ}	93
5.7	Select Instructions	94
5.8	Register Allocation	95
6	Tuples and Garbage Collection	97
6.1	The $\mathcal{L}_{\text{Tuple}}$ Language	97
6.2	Garbage Collection	100
6.3	Expose Allocation	108
6.4	Remove Complex Operands	109
6.5	Explicate Control and the $\mathcal{C}_{\text{Tuple}}$ Language	110
6.6	Select Instructions and the $\text{x86}_{\text{Global}}$ Language	110
6.7	Register Allocation	115
6.8	Prelude and Conclusion	115
6.9	Challenge: Simple Structures	117
6.10	Challenge: Arrays	119
6.11	Uncover <code>get!</code>	123
6.12	Challenge: Generational Collection	124
6.13	Further Reading	125
7	Functions	127
7.1	The \mathcal{L}_{Fun} Language	127
7.2	Functions in x86	132
7.3	Shrink \mathcal{L}_{Fun}	135
7.4	Reveal Functions and the $\mathcal{L}_{\text{FunRef}}$ Language	135

7.5	Limit Functions	135
7.6	Remove Complex Operands	136
7.7	Explicate Control and the \mathcal{C}_{Fun} Language	136
7.8	Select Instructions and the $\text{x86}_{\text{callq*}}^{\text{Def}}$ Language	138
7.9	Register Allocation	140
7.10	Patch Instructions	141
7.11	Prelude and Conclusion	141
7.12	An Example Translation	143
8	Lexically Scoped Functions	145
8.1	The \mathcal{L}_λ Language	147
8.2	Assignment and Lexically Scoped Functions	150
8.3	Assignment Conversion	150
8.4	Closure Conversion	152
8.5	An Example Translation	154
8.6	Expose Allocation	155
8.7	Explicate Control and $\mathcal{C}_{\text{Clos}}$	155
8.8	Select Instructions	155
8.9	Challenge: Optimize Closures	158
8.10	Further Reading	160
9	Dynamic Typing	161
9.1	The \mathcal{L}_{Dyn} Language	161
9.2	Representation of Tagged Values	166
9.3	The \mathcal{L}_{Any} Language	166
9.4	Cast Insertion: Compiling \mathcal{L}_{Dyn} to \mathcal{L}_{Any}	172
9.5	Reveal Casts	173
9.6	Remove Complex Operands	174
9.7	Explicate Control and \mathcal{C}_{Any}	174
9.8	Select Instructions	174
9.9	Register Allocation for \mathcal{L}_{Any}	177
10	Gradual Typing	179
10.1	Type Checking $\mathcal{L}_?$	179
10.2	Interpreting $\mathcal{L}_{\text{Cast}}$	187
10.3	Cast Insertion	188
10.4	Lower Casts	191
10.5	Differentiate Proxies	192
10.6	Reveal Casts	194
10.7	Closure Conversion	195
10.8	Select Instructions	195
10.9	Further Reading	196
11	Generics	199
11.1	Compiling Generics	206

11.2	Resolve Instantiation	207
11.3	Erase Generic Types	207
A	Appendix	211
A.1	Interpreters	211
A.2	Utility Functions	211
A.3	x86 Instruction Set Quick Reference	212
	References	215
	Index	223