

Introduction to Compilers and Language Design

**Prof. Douglas Thain
University of Notre Dame**

Introduction to Compilers and Language Design

Copyright © 2018 Douglas Thain.

Hardcover ISBN: 978-0-359-13804-3

Paperback ISBN: 978-0-359-14283-5

First edition.

Anyone is free to download and print the PDF edition of this book for personal use. Commercial distribution, printing, or reproduction without the author's consent is expressly prohibited. All other rights are reserved.

You can find the latest version of the PDF edition, and purchase inexpensive hardcover copies at <http://compilerbook.org>

Revision Date: May 2, 2019

For Lisa, William, Zachary, Emily, and Alia.

Contributions

I am grateful to the following people for their contributions to this book:

Andrew Litteken drafted the chapter on ARM assembly; Kevin Latimer drew the RegEx to NFA and the LR example figures; Benjamin Gunning fixed an error in LL(1) parse table construction; Tim Shaffer completed the detailed LR(1) example.

And the following people corrected typos:

John Westhoff (25), Gonzalo Martinez (25), Daniel Kerrigan (24), Brian DuSell (23), Ryan Mackey (20), Nedim Mininovic (15), Joseph Kimlinger (12), Andrew Litteken (9), Thomas Cane (9), Stéphane Massou (8), Luis Prieb (7), Jonathan Xu (6), John Johnson (4), Spencer King (2), Yaoxian Qu (2), Maria Aranguren (2), Patrick Lacher (2), Connor Higgins (2), Tango Gu (2), Andrew Syrmakesis (2), Horst von Brand (2), Benjamin Gunning (1), Charles Osborne (1), William Theisen (1), Jessica Cioffi (1), Ben Tovar (1), Ryan Michalec (1), Patrick Flynn (1), Clint Jeffery (1), Ralph Siemsen (1)

Please send any comments or corrections via email to Prof. Douglas Thain (dthain@nd.edu).

Contents

1	Introduction	1
1.1	What is a compiler?	1
1.2	Why should you study compilers?	2
1.3	What's the best way to learn about compilers?	2
1.4	What language should I use?	2
1.5	How is this book different from others?	3
1.6	What other books should I read?	4
2	A Quick Tour	5
2.1	The Compiler Toolchain	5
2.2	Stages Within a Compiler	6
2.3	Example Compilation	7
2.4	Exercises	10
3	Scanning	11
3.1	Kinds of Tokens	11
3.2	A Hand-Made Scanner	12
3.3	Regular Expressions	13
3.4	Finite Automata	15
3.4.1	Deterministic Finite Automata	16
3.4.2	Nondeterministic Finite Automata	17
3.5	Conversion Algorithms	19
3.5.1	Converting REs to NFAs	19
3.5.2	Converting NFAs to DFAs	22
3.5.3	Minimizing DFAs	24
3.6	Limits of Finite Automata	26
3.7	Using a Scanner Generator	26
3.8	Practical Considerations	29
3.9	Exercises	31
3.10	Further Reading	33
4	Parsing	35
4.1	Overview	35
4.2	Context Free Grammars	35

4.2.1	Deriving Sentences	36
4.2.2	Ambiguous Grammars	37
4.3	LL Grammars	40
4.3.1	Eliminating Left Recursion	40
4.3.2	Eliminating Common Left Prefixes	41
4.3.3	First and Follow Sets	42
4.3.4	Recursive Descent Parsing	44
4.3.5	Table Driven Parsing	46
4.4	LR Grammars	48
4.4.1	Shift-Reduce Parsing	49
4.4.2	The LR(0) Automaton	50
4.4.3	SLR Parsing	54
4.4.4	LR(1) Parsing	59
4.4.5	LALR Parsing	61
4.5	Grammar Classes Revisited	61
4.6	The Chomsky Hierarchy	62
4.7	Exercises	64
4.8	Further Reading	66
5	Parsing in Practice	67
5.1	The Bison Parser Generator	68
5.2	Expression Validator	71
5.3	Expression Interpreter	72
5.4	Expression Trees	73
5.5	Exercises	79
5.6	Further Reading	81
6	The Abstract Syntax Tree	83
6.1	Overview	83
6.2	Declarations	84
6.3	Statements	86
6.4	Expressions	88
6.5	Types	90
6.6	Putting it All Together	93
6.7	Building the AST	94
6.8	Exercises	96
7	Semantic Analysis	97
7.1	Overview of Type Systems	98
7.2	Designing a Type System	101
7.3	The B-Minor Type System	104
7.4	The Symbol Table	105
7.5	Name Resolution	109
7.6	Implementing Type Checking	111
7.7	Error Messages	115

7.8	Exercises	116
7.9	Further Reading	116
8	Intermediate Representations	117
8.1	Introduction	117
8.2	Abstract Syntax Tree	117
8.3	Directed Acyclic Graph	118
8.4	Control Flow Graph	123
8.5	Static Single Assignment Form	125
8.6	Linear IR	126
8.7	Stack Machine IR	127
8.8	Examples	128
8.8.1	GIMPLE - GNU Simple Representation	128
8.8.2	LLVM - Low Level Virtual Machine	129
8.8.3	JVM - Java Virtual Machine	130
8.9	Exercises	131
8.10	Further Reading	132
9	Memory Organization	133
9.1	Introduction	133
9.2	Logical Segmentation	133
9.3	Heap Management	136
9.4	Stack Management	138
9.4.1	Stack Calling Convention	139
9.4.2	Register Calling Convention	140
9.5	Locating Data	141
9.6	Program Loading	144
9.7	Further Reading	146
10	Assembly Language	147
10.1	Introduction	147
10.2	Open Source Assembler Tools	148
10.3	X86 Assembly Language	150
10.3.1	Registers and Data Types	150
10.3.2	Addressing Modes	152
10.3.3	Basic Arithmetic	154
10.3.4	Comparisons and Jumps	156
10.3.5	The Stack	157
10.3.6	Calling a Function	158
10.3.7	Defining a Leaf Function	160
10.3.8	Defining a Complex Function	161
10.4	ARM Assembly	165
10.4.1	Registers and Data Types	165
10.4.2	Addressing Modes	166
10.4.3	Basic Arithmetic	168

10.4.4	Comparisons and Branches	169
10.4.5	The Stack	171
10.4.6	Calling a Function	173
10.4.7	Defining a Leaf Function	173
10.4.8	Defining a Complex Function	174
10.4.9	64-bit Differences	177
10.5	Further Reading	178
11	Code Generation	179
11.1	Introduction	179
11.2	Supporting Functions	179
11.3	Generating Expressions	181
11.4	Generating Statements	186
11.5	Conditional Expressions	190
11.6	Generating Declarations	191
11.7	Exercises	192
12	Optimization	193
12.1	Overview	193
12.2	Optimization in Perspective	194
12.3	High Level Optimizations	195
12.3.1	Constant Folding	195
12.3.2	Strength Reduction	197
12.3.3	Loop Unrolling	197
12.3.4	Code Hoisting	198
12.3.5	Function Inlining	199
12.3.6	Dead Code Detection and Elimination	200
12.4	Low-Level Optimizations	202
12.4.1	Peephole Optimizations	202
12.4.2	Instruction Selection	202
12.5	Register Allocation	206
12.5.1	Safety of Register Allocation	206
12.5.2	Priority of Register Allocation	207
12.5.3	Conflicts Between Variables	207
12.5.4	Global Register Allocation	208
12.6	Optimization Pitfalls	209
12.7	Optimization Interactions	211
12.8	Exercises	212
12.9	Further Reading	213
A	Sample Course Project	215
A.1	Scanner Assignment	215
A.2	Parser Assignment	215
A.3	Pretty-Printer Assignment	216
A.4	Typechecker Assignment	216

A.5	Optional: Intermediate Representation	216
A.6	Code Generator Assignment	216
A.7	Optional: Extend the Language	217
B	The B-Minor Language	219
B.1	Overview	219
B.2	Tokens	220
B.3	Types	220
B.4	Expressions	221
B.5	Declarations and Statements	222
B.6	Functions	222
B.7	Optional Elements	223
C	Coding Conventions	225
	Index	227

List of Figures

2.1	A Typical Compiler Toolchain	5
2.2	The Stages of a Unix Compiler	6
2.3	Example AST	9
2.4	Example Intermediate Representation	9
2.5	Example Assembly Code	10
3.1	A Simple Hand Made Scanner	12
3.2	Relationship Between REs, NFAs, and DFAs	19
3.3	Subset Construction Algorithm	22
3.4	Converting an NFA to a DFA via Subset Construction	23
3.5	Hopcroft's DFA Minimization Algorithm	24
3.6	Structure of a Flex File	27
3.7	Example Flex Specification	28
3.8	Example Main Program	28
3.9	Example Token Enumeration	29
3.10	Build Procedure for a Flex Program	30
4.1	Two Derivations of the Same Sentence	38
4.2	A Recursive-Descent Parser	45
4.3	LR(0) Automaton for Grammar G_{10}	52
4.4	SLR Parse Table for Grammar G_{10}	55
4.5	Part of LR(0) Automaton for Grammar G_{11}	58
4.6	LR(1) Automaton for Grammar G_{10}	60
4.7	The Chomsky Hierarchy	63
5.1	Bison Specification for Expression Validator	69
5.2	Main Program for Expression Validator	70
5.3	Build Procedure for Bison and Flex Together	70
5.4	Bison Specification for an Interpreter	73
5.5	AST for Expression Interpreter	74
5.6	Building an AST for the Expression Grammar	76
5.7	Evaluating Expressions	78
5.8	Printing and Evaluating Expressions	79
7.1	The Symbol Structure	105

7.2	A Nested Symbol Table	107
7.3	Symbol Table API	108
7.4	Name Resolution for Declarations	110
7.5	Name Resolution for Expressions	110
8.1	Sample DAG Data Structure Definition	118
8.2	Example of Constant Folding	123
8.3	Example Control Flow Graph	124
9.1	Flat Memory Model	133
9.2	Logical Segments	134
9.3	Multiprogrammed Memory Layout	135
10.1	X86 Register Structure	151
10.2	X86 Register Structure	152
10.3	Summary of System V ABI Calling Convention	158
10.4	System V ABI Register Assignments	160
10.5	Example X86-64 Stack Layout	162
10.6	Complete X86 Example	164
10.7	ARM Addressing Modes	167
10.8	ARM Branch Instructions	170
10.9	Summary of ARM Calling Convention	172
10.10	ARM Register Assignments	172
10.11	Example ARM Stack Frame	175
10.12	Complete ARM Example	176
11.1	Code Generation Functions	180
11.2	Example of Generating X86 Code from a DAG	182
11.3	Expression Generation Skeleton	184
11.4	Generating Code for a Function Call	185
11.5	Statement Generator Skeleton	186
12.1	Timing a Fast Operation	195
12.2	Constant Folding Pseudo-Code	196
12.3	Example X86 Instruction Templates	204
12.4	Example of Tree Rewriting	205
12.5	Live Ranges and Register Conflict Graph	208
12.6	Example of Global Register Allocation	209