

System Architecture

Wolfgang J. Paul · Christoph Baumann
Petro Lutsyk · Sabine Schmaltz

System Architecture

An Ordinary Engineering Discipline

Wolfgang J. Paul
FR 6.1 Informatik
Universität des Saarlandes
Saarbrücken, Saarland
Germany

Petro Lutsyk
FR 6.1 Informatik
Universität des Saarlandes
Saarbrücken, Saarland
Germany

Christoph Baumann
School of Computer Science
and Communication
KTH Royal Institute of Technology
Stockholm
Sweden

Sabine Schmaltz
FR 6.1 Informatik
Universität des Saarlandes
Saarbrücken, Saarland
Germany

ISBN 978-3-319-43064-5

ISBN 978-3-319-43065-2 (eBook)

DOI 10.1007/978-3-319-43065-2

Library of Congress Control Number: 2016952820

© Springer International Publishing Switzerland 2016

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

1	Introduction	1
2	Understanding Decimal Addition	7
2.1	Experience Versus Understanding	7
2.2	The Natural Numbers	8
2.2.1	$2 + 1 = 3$ is a Definition	9
2.2.2	$1 + 2 = 3$ is a Theorem	9
2.2.3	$9 + 1 = 10$ is a Brilliant Theorem	11
2.3	Final Remarks	13
2.4	Exercises	14
3	Basic Mathematical Concepts	15
3.1	Basics	16
3.1.1	Implicit Quantification	16
3.1.2	Numbers and Sets	17
3.1.3	Sequences, Their Indexing and Overloading	18
3.1.4	Bytes, Logical Connectives, and Vector Notation	20
3.2	Modulo Computation	21
3.3	Sums	24
3.3.1	Geometric Sums	24
3.3.2	Arithmetic Sums	25
3.4	Graphs	25
3.4.1	Directed Graphs	25
3.4.2	Directed Acyclic Graphs and the Depth of Nodes	27
3.4.3	Rooted Trees	28
3.5	Final Remarks	29
3.6	Exercises	30
4	Number Formats and Boolean Algebra	33
4.1	Binary Numbers	33
4.2	Two's Complement Numbers	37
4.3	Boolean Algebra	39
4.3.1	Useful Identities	42
4.3.2	Solving Equations	44

4.3.3	Disjunctive Normal Form	45
4.4	Final Remarks	47
4.5	Exercises	48
5	Hardware	51
5.1	Gates and Circuits	51
5.2	Some Basic Circuits	56
5.3	Clocked Circuits	60
5.4	Registers	66
5.5	Final Remarks	66
5.6	Exercises	67
6	Five Designs of RAM	71
6.1	Basic Random Access Memory	71
6.2	Read-Only Memory (ROM)	73
6.3	Combining RAM and ROM	73
6.4	Three-Port RAM for General-Purpose Registers	75
6.5	SPR-RAM	77
6.6	Final Remarks	79
6.7	Exercises	79
7	Arithmetic Circuits	81
7.1	Adder and Incrementer	81
7.1.1	Carry Chain Adder and Incrementer	82
7.1.2	Conditional-Sum Adders	83
7.1.3	Parallel Prefix Circuits	86
7.1.4	Carry-Look-Ahead Adders	88
7.2	Arithmetic Unit	89
7.3	Arithmetic Logic Unit (ALU)	94
7.4	Shifter	96
7.5	Branch Condition Evaluation Unit	98
7.6	Final Remarks	100
7.7	Exercises	101
8	A Basic Sequential MIPS Machine	109
8.1	Tables	110
8.1.1	I-type	110
8.1.2	R-type	111
8.1.3	J-type	111
8.2	MIPS ISA	112
8.2.1	Configuration and Instruction Fields	112
8.2.2	Instruction Decoding	114
8.2.3	ALU Operations	115
8.2.4	Shift	118
8.2.5	Branch and Jump	118

8.2.6	Loads and Stores	120
8.2.7	ISA Summary	122
8.3	A Sequential Processor Design	123
8.3.1	Hardware Configuration	123
8.3.2	Fetch and Execute Cycles	125
8.3.3	Reset	125
8.3.4	Instruction Fetch	126
8.3.5	Proof Goals for the Execute Stage	128
8.3.6	Instruction Decoder	128
8.3.7	Reading from General-Purpose Registers	132
8.3.8	Next PC Environment	133
8.3.9	ALU Environment	135
8.3.10	Shifter Environment	136
8.3.11	Jump and Link	137
8.3.12	Collecting Results	137
8.3.13	Effective Address	138
8.3.14	Memory Environment	138
8.3.15	Writing to the General-Purpose Register File	140
8.4	Final Remarks	140
8.5	Exercises	141
9	Some Assembler Programs	145
9.1	Simple MIPS Programs	146
9.2	Software Multiplication	147
9.3	Software Division	149
9.3.1	School Method for Non-negative Integer Division	149
9.3.2	‘Small’ Unsigned Integer Division	150
9.3.3	Unsigned Integer Division	153
9.3.4	Integer Division	155
9.4	Final Remarks	156
9.5	Exercises	156
10	Context-Free Grammars	159
10.1	Introduction to Context-Free Grammars	160
10.1.1	Syntax of Context-Free Grammars	160
10.1.2	Quick and Dirty Introduction to Derivation Trees	161
10.1.3	Tree Regions	163
10.1.4	Clean Definition of Derivation Trees	165
10.1.5	Composition and Decomposition of Derivation Trees	167
10.1.6	Generated Languages	168
10.2	Grammars for Expressions	168
10.2.1	Syntax of Boolean Expressions	168
10.2.2	Grammar for Arithmetic Expressions with Priorities	170
10.2.3	Proof of Lemma 66	171
10.2.4	Distinguishing Unary and Binary Minus	175

10.3	Final Remarks	175
10.4	Exercises	176
11	The Language C0	179
11.1	Grammar of C0	180
11.1.1	Names and Constants	180
11.1.2	Identifiers	182
11.1.3	Arithmetic and Boolean Expressions	183
11.1.4	Statements	183
11.1.5	Programs	184
11.1.6	Type and Variable Declarations	184
11.1.7	Function Declarations	185
11.1.8	Representing and Processing Derivation Trees in C0	186
11.1.9	Sequence Elements and Flattened Sequences in the C0 Grammar	189
11.2	Declarations	190
11.2.1	Type Tables	190
11.2.2	Global Variables	193
11.2.3	Function Tables	194
11.2.4	Variables and Subvariables of all C0 Configurations	196
11.2.5	Range of Types and Default Values	197
11.3	C0 Configurations	199
11.3.1	Variables, Subvariables, and Their Type in C0 Configurations <i>c</i>	199
11.3.2	Value of Variables, Type Correctness, and Invariants	202
11.3.3	Expressions and Statements in Function Bodies	204
11.3.4	Program Rest	207
11.3.5	Result Destination Stack	209
11.3.6	Initial Configuration	209
11.4	Expression Evaluation	210
11.4.1	Type, Right Value, and Left Value of Expressions	212
11.4.2	Constants	214
11.4.3	Variable Binding	215
11.4.4	Pointer Dereferencing	217
11.4.5	Struct Components	217
11.4.6	Array Elements	218
11.4.7	'Address of'	219
11.4.8	Unary Operators	219
11.4.9	Binary Operators	221
11.5	Statement Execution	222
11.5.1	Assignment	223
11.5.2	Conditional Statement	224
11.5.3	While Loop	224
11.5.4	'New' Statement	225
11.5.5	Function Call	227

11.5.6	Return	230
11.6	Proving the Correctness of C0 Programs	232
11.6.1	Assignment and Conditional Statement	232
11.6.2	Computer Arithmetic	234
11.6.3	While Loop	234
11.6.4	Linked Lists	236
11.6.5	Recursion	242
11.7	Final Remarks	246
11.8	Exercises	248
12	A C0-Compiler	253
12.1	Compiler Consistency	254
12.1.1	Memory Map	254
12.1.2	Size of Types, Displacement, and Base Address	256
12.1.3	Consistency for Data, Pointers, and the Result Destination Stack	260
12.1.4	Consistency for the Code	262
12.1.5	Consistency for the Program Rest	264
12.1.6	Relating the Derivation Tree and the Program Rest	270
12.2	Translation of Expressions	275
12.2.1	Sethi-Ullman Algorithm	275
12.2.2	The <i>R</i> -label	280
12.2.3	Composable MIPS Programs	286
12.2.4	Correctness of Code for Expressions	288
12.2.5	Constants	290
12.2.6	Variable Names	291
12.2.7	Struct Components	293
12.2.8	Array Elements	294
12.2.9	Dereferencing Pointers	295
12.2.10	'Address of'	296
12.2.11	Unary Operators	297
12.2.12	Binary Arithmetic Operators	298
12.2.13	Comparison	300
12.2.14	Translating Several Expressions and Maintaining the Results	301
12.3	Translation of Statements	302
12.3.1	Assignment	303
12.3.2	New Statement	304
12.3.3	While Loop	305
12.3.4	If-Then-Else	307
12.3.5	If-Then	308
12.3.6	Function Call	308
12.3.7	Return	313
12.3.8	Summary of Intermediate Results	315
12.4	Translation of Programs	316
12.4.1	Statement Sequences	316

12.4.2	Function Bodies	317
12.4.3	Function Declaration Sequences	318
12.4.4	Programs	318
12.4.5	Jumping Out of Loops and Conditional Statements	320
12.5	Compiler Correctness Revisited	322
12.5.1	Christopher Lee and the Truth About Life After Death	322
12.5.2	Consistency Points	323
12.5.3	Compiler Correctness for Optimizing Compilers	324
12.6	Final Remarks	325
12.7	Exercises	326
13	Compiler Consistency Revisited	333
13.1	Reconstructing a Well-Formed C0 Configuration	334
13.1.1	Associating Code Addresses with Statements and Functions	335
13.1.2	Reconstructing Everything Except Heap and Pointers	337
13.1.3	Reachable Subvariables	340
13.1.4	Implementation Subvariables	342
13.1.5	Heap Reconstruction Is not Unique	347
13.1.6	Heap Isomorphisms and Equivalence of C0 Configurations	348
13.1.7	Computations Starting in Equivalent Configurations	351
13.2	Garbage Collection	355
13.2.1	Pointer Chasing	356
13.2.2	Garbage-Collected Equivalent Configurations	360
13.2.3	Construction of a Garbage-Collected MIPS Configuration	362
13.3	C0 + Assembly	365
13.3.1	Syntax	365
13.3.2	Compilation	366
13.3.3	Semantics and Compiler Correctness	367
13.4	Final Remarks	372
13.5	Exercises	373
14	Operating System Support	375
14.1	Interrupts	376
14.1.1	Types of Interrupts	376
14.1.2	Special Purpose Registers and New Instructions	376
14.1.3	MIPS ISA with Interrupts	378
14.1.4	Specification of Most Internal Interrupt Event Signals	381
14.1.5	Hardware	381
14.1.6	Hardware Correctness	384
14.2	Address Translation	384
14.2.1	Specification	385
14.2.2	Hardware	388
14.3	Disks	391
14.3.1	Hardware Model of a Disk	392
14.3.2	Accessing a Device with Memory-Mapped I/O	395

14.3.3	Nondeterminism Revisited	397
14.3.4	Nondeterministic ISA for Processor + Disk	400
14.3.5	Hardware Correctness	404
14.3.6	Order Reduction	406
14.3.7	Disk Liveness in Reordered Computations	410
14.3.8	C0 + Assembly + Disk + Interrupts	414
14.3.9	Hard Disk Driver	417
14.4	Final Remarks	421
14.5	Exercises	424

15 A Generic Operating System Kernel 429

15.1	Physical and Virtual Machines	431
15.1.1	Physical Machines	431
15.1.2	Virtual Machines	431
15.2	Communicating Virtual Machines	434
15.2.1	Configurations	434
15.2.2	Semantics	435
15.3	Concrete Kernel	438
15.3.1	Data Structures	439
15.3.2	Virtual Address Translation via C Data Structures	441
15.3.3	Simulation Relation for Virtual Machines	442
15.3.4	Encoding an Abstract Kernel by a Concrete Kernel	444
15.3.5	Simulation Relation for the Abstract Kernel of CVM	446
15.3.6	Technical Invariants	449
15.3.7	Formulating the Correctness Theorem	450
15.4	The <i>runvm</i> Primitive	452
15.4.1	Overview of the <i>runvm</i> Primitive	453
15.4.2	Program Annotations	454
15.4.3	Maintaining <i>k-consis</i>	455
15.4.4	Maintaining <i>consis</i>	458
15.4.5	Maintaining <i>inv-vm</i>	459
15.5	Simulation of CVM Steps	461
15.5.1	Scratch Memory	461
15.5.2	C0 Step of the Kernel	463
15.5.3	ISA Step of a User Without Interrupt	464
15.5.4	Restoring a User Process	466
15.5.5	Testing for Reset	470
15.5.6	Boot Loader and Initialization	472
15.5.7	Process Save	474
15.6	Page Fault Handling	478
15.6.1	Testing for <i>ipf</i> Page Fault	478
15.6.2	Auxiliary Data Structures	481
15.6.3	Swapping a Page Out	485
15.6.4	Swapping a Page In	487
15.6.5	Liveness	490

XII CONTENTS

15.7 Other CVM Primitives and Dispatching	490
15.7.1 Accessing Registers of User Processes	491
15.7.2 Accessing User Pages	491
15.7.3 <i>free</i> and <i>alloc</i>	492
15.7.4 Application Binary Interface and Dispatcher for the Ab- stract Kernel	497
15.8 Final Remarks	499
15.9 Exercises	500
References	503
Index	507