

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Algorithms and Programs</b>	<b>11</b>
2.1	Simple Algorithms	14
2.1.1	Getting Started with Some Basics	15
2.1.2	Recursive Functions	19
2.1.3	The Termination Problem	23
2.1.4	Symbolic Computations	24
2.1.5	Operating on Lists	30
2.2	A Word on Typing	31
2.3	Summary	34
<b>3</b>	<b>An Algorithmic Language</b>	<b>37</b>
3.1	The Syntax of AL Expressions	38
3.2	The Evaluation of AL Expressions	41
3.3	Summary	47
<b>4</b>	<b>The <math>\lambda</math>-Calculus</b>	<b>51</b>
4.1	$\lambda$ -Calculus Notation	52
4.2	$\beta$ -Reduction and $\alpha$ -Conversion	53
4.3	An Indexing Scheme for Bound Variables **	59
4.4	The Nameless $\lambda$ -Calculus	63
4.5	Reduction Sequences	68
4.6	Recursion in the $\lambda$ -Calculus	73
4.7	A Brief Outline of an Applied $\lambda$ -Calculus	78
4.8	Overview of a Typed $\lambda$ -Calculus	79
4.8.1	Monomorphic Types	81
4.8.2	Polymorphic Types	83
4.9	Summary	86

<b>5</b>	<b>The <math>\text{SE(M)CD}</math> Machine and Others</b>	89
5.1	An Outline of the Original SECD Machine	89
5.2	The $\text{SE(M)CD}$ Machine	93
5.2.1	The Traversal Mechanism	94
5.2.2	Doing $\beta$ -Reductions	96
5.2.3	Reducing a Simple Expression	98
5.3	The $\#\text{SE(M)CD}$ Machine for the Nameless $\lambda$ -Calculus	101
5.4	Implementing $\delta$ -Reductions	102
5.5	Other Weakly Normalizing Abstract Machines	105
5.5.1	The $\mathcal{K}$ -Machine	105
5.5.2	The Categorical Abstract Machine	107
5.6	Summary	108
<b>6</b>	<b>Toward Full-Fledged <math>\lambda</math>-Calculus Machines</b>	113
6.1	Berklings's String Reduction Machine	115
6.2	Wadsworth's Graph Reduction Techniques	121
6.3	The $\lambda\sigma$ -Calculus Abstract Machine	125
6.3.1	The $\lambda\sigma$ -Calculus **	126
6.3.2	The Abstract Machine **	130
6.4	Head-Order Reduction	132
6.4.1	Head Forms and Head-Order $\beta$ -Reductions	134
6.4.2	An Abstract Head-Order Reduction (HOR) Machine **	141
6.5	Summary	145
<b>7</b>	<b>Interpreted Head-Order Graph Reduction</b>	149
7.1	Graph Representation and Graph Reduction	150
7.2	Continuing with Reductions in the Head	156
7.3	Reducing the Tails	160
7.4	An Outline of the Formal Specification of $\text{G\_HOR}$	163
7.5	Garbage Collection	164
7.6	Summary	167
<b>8</b>	<b>The <math>B</math>-Machine</b>	171
8.1	The Operating Principles of the $B$ -Machine	173
8.2	The Instruction Set	174
8.2.1	Instruction Interpretation Without Sharing **	176
8.2.2	Interpretation Under Sharing in the Head **	179
8.3	Executing $B$ -Machine Code: an Example **	181
8.4	Supporting Primitive Functions	186
8.5	Summary	189
<b>9</b>	<b>The <math>G</math>-Machine</b>	193
9.1	Basic Language Issues	195
9.2	Basic Operating Principles of the $G$ -Machine	197
9.3	Compiling Supercombinators to $G$ -Machine Code	201

9.4	<i>G</i> -Code for Primitive Functions .....	204
9.5	The Controlling Instructions * .....	205
9.6	Some <i>G</i> -Code Optimizations .....	209
9.7	Summary .....	211
<b>10</b>	<b>The <math>\pi</math>-RED Machinery .....</b>	<b>215</b>
10.1	The Basic Program Execution Cycle .....	215
10.2	The Operating Principles of the Abstract Machines .....	221
10.3	The Lazy Abstract Stack Machine LASM .....	223
10.3.1	The LASM Instruction Set .....	225
10.3.2	Compilation to LASM Code * .....	228
10.3.3	Some Simple Code Optimizations .....	232
10.4	The Strict Abstract Stack Machine SASM .....	235
10.4.1	The SASM Instruction Set .....	236
10.4.2	Compilation to SASM Code * .....	237
10.4.3	Code Execution .....	240
10.5	Reducing to Full Normal Forms * .....	244
10.6	Summary .....	249
<b>11</b>	<b>Pattern Matching .....</b>	<b>253</b>
11.1	Pattern Matching in AL .....	253
11.2	Programming with Pattern Matches .....	255
11.3	Preprocessing Pattern Matches .....	258
11.4	The Pattern Matching Machinery .....	260
11.5	Compiling Pattern Matches to LASM Code * .....	263
11.6	Code Generation and Execution: an Example ** .....	265
11.7	Summary .....	268
<b>12</b>	<b>Another Functional Abstract Machine .....</b>	<b>271</b>
12.1	The Machine and How It Basically Works .....	272
12.1.1	Some Semantic Issues .....	273
12.1.2	Index Tuples and the Runtime Environment .....	275
12.2	The SECD-I Instruction Set .....	279
12.3	Compilation to SECD-I Code .....	282
12.4	Summary .....	285
<b>13</b>	<b>Imperative Abstract Machines .....</b>	<b>289</b>
13.1	Outline of an Imperative Kernel Language .....	291
13.2	An Example of an IL Program .....	294
13.3	The Runtime Environment .....	296
13.3.1	Using Static and Dynamic Links .....	298
13.3.2	Dropping Dynamic Links .....	300
13.3.3	Calculating Stack Addresses * .....	302
13.4	The Instruction Set .....	304
13.5	Compiling IL Programs to IAM Code * .....	306

13.6	Compiling the Bubble-Sort Program .....	309
13.7	Outline of a Machine for a 'Flat' Language .....	312
13.8	Summary .....	318
<b>14</b>	<b>Real Computing Machines .....</b>	<b>321</b>
14.1	A Typical CISC Architecture .....	323
14.1.1	The Register Set, Formats and Addressing in Memory .	324
14.1.2	Addressing Modes .....	326
14.1.3	Some Important Instructions .....	328
14.1.4	Implementing Procedure Calls .....	330
14.2	A Typical RISC Architecture .....	334
14.2.1	The SPARC Register Set .....	335
14.2.2	Some Important SPARC Instructions .....	339
14.2.3	The SPARC Assembler Code for Factorial .....	341
14.3	Summary .....	344
<b>A</b>	<b>Input/Output .....</b>	<b>347</b>
A.1	Functions as Input/Output Mappings .....	348
A.2	Continuation-Style Input/Output .....	354
A.3	Interactions with a File System .....	357
<b>B</b>	<b>On Theorem Proving .....</b>	<b>361</b>
	<b>References .....</b>	<b>369</b>
	<b>Index .....</b>	<b>377</b>

Abstract Computing Machines  
A Lambda Calculus Perspective

Kluge, W.

2005, XIV, 384 p. 89 illus., Hardcover

ISBN: 978-3-540-21146-4