Contents

About the Author xix

About the Technical Reviewer xxi

Acknowledgments xxiii

Introduction xxv

■Part I: Assembly Language and Computer Architecture 1

■Chapter 1: Basic Computer Architecture 3

1.1 The Core Architecture 3

1.1.1 Model of Computation 3

1.1.2 von Neumann Architecture 3

1.2 Evolution 5

1.2.1 Drawbacks of von Neumann Architecture 5

1.2.2 Intel 64 Architecture 6

1.2.3 Architecture Extensions 6

1.3 Registers 7

1.3.1 General Purpose Registers 8

1.3.2 Other Registers 11

1.3.3 System Registers 12

1.4 Protection Rings 14

1.5 Hardware Stack 14

1.6 Summary 16

Chapter 2: Assembly Language 17
2.1 Setting Up the Environment 17
2.1.1 Working with Code Examples 18
2.2 Writing "Hello, world" 18
2.2.1 Basic Input and Output 18
2.2.2 Program Structure 19
2.2.3 Basic Instructions 20
2.3 Example: Output Register Contents 22
2.3.1 Local Labels 23
2.3.2 Relative Addressing 23
2.3.3 Order of Execution 24
2.4 Function Calls 25
2.5 Working with Data 28
2.5.1 Endianness 28
2.5.2 Strings 29
2.5.3 Constant Precomputation 30
2.5.4 Pointers and Different Addressing Types 30
2.6 Example: Calculating String Length 32
2.7 Assignment: Input/Output Library 34
2.7.1 Self-Evaluation 35
2.8 Summary 36
Chapter 3: Legacy 39
3.1 Real mode 39
3.2 Protected Mode 40
3.3 Minimal Segmentation in Long Mode 44
3.4 Accessing Parts of Registers 45
3.4.1 An Unexpected Behavior 45
3.4.2 CISC and RISC 45
3.4.3 Explanation 46
3.5 Summary 46

Chapter 4: Virtual Memory **47**

4.1 Caching 47

4.2 Motivation 47

4.3 Address Spaces 48

4.4 Features 49

4.5 Example: Accessing Forbidden Address 50

4.6 Efficiency 52

4.7 Implementation 52

4.7.1 Virtual Address Structure 53

4.7.2 Address Translation in Depth 53

4.7.3 Page Sizes 56

4.8 Memory Mapping 56

4.9 Example: Mapping File into Memory 57

4.9.1 Mnemonic Names for Constants 57

4.9.2 Complete Example 58

4.10 Summary 60

Chapter 5: Compilation Pipeline **63**

5.1 Preprocessor 64

5.1.1 Simple Substitutions 64

5.1.2 Substitutions with Arguments 65

5.1.3 Simple Conditional Substitution 66

5.1.4 Conditioning on Definition 67

5.1.5 Conditioning on Text Identity 67

5.1.6 Conditioning on Argument Type 68

5.1.7 Evaluation Order: Define, xdefine, Assign 69

5.1.8 Repetition 70

5.1.9 Example: Computing Prime Numbers 71

5.1.10 Labels Inside Macros 72

5.1.11 Conclusion 73

5.2 Translation 74
5.3 Linking 74
5.3.1 Executable and Linkable Format 74
5.3.2 Relocatable Object Files 76
5.3.3 Executable Object Files 80
5.3.4 Dynamic Libraries 81
5.3.5 Loader 85
5.4 Assignment: Dictionary 87
5.5 Summary 89
Chapter 6: Interrupts and System Calls 91
6.1 Input and Output 91
6.1.1 TR register and Task State Segment 92
6.2 Interrupts 94
6.3 System Calls 97
6.3.1 Model-Specific Registers 97
6.3.2 syscall and sysret 97
6.4 Summary 99
Chapter 7: Models of Computation 101
7.1 Finite State Machines 101
7.1.1 Definition 101
7.1.2 Example: Bits Parity 103
7.1.3 Implementation in Assembly Language 103
7.1.4 Practical Value 105
7.1.5 Regular Expressions 106
7.2 Forth Machine 109
7.2.1 Architecture 109
7.2.2 Tracing an Exemplary Forth Program 111
7.2.3 Dictionary 112
7.2.4 How Words Are Implemented 112
7.2.5 Compiler 117

7.3 Assignment: Forth Compiler and Interpreter........ 118
7.3.1 Static Dictionary, Interpreter........ 118
7.3.2 Compilation........ 121
7.3.3 Forth with Bootstrap........ 123
7.4 Summary........ 125
Part II: The C Programming Language........ 127
Chapter 8: Basics........ 129
8.1 Introduction........ 129
8.2 Program Structure........ 130
8.2.1 Data Types........ 132
8.3 Control Flow........ 133
8.3.1 if........ 134
8.3.2 while........ 135
8.3.3 for........ 135
8.3.4 goto........ 136
8.3.5 switch........ 137
8.3.6 Example: Divisor........ 138
8.3.7 Example: Is It a Fibonacci Number?........ 138
8.4 Statements and Expressions........ 139
8.4.1 Statement Types........ 140
8.4.2 Building Expressions........ 141
8.5 Functions........ 142
8.6 Preprocessor........ 144
8.7 Summary........ 146
Chapter 9: Type System........ 147
9.1 Basic Type System of C........ 147
9.1.1 Numeric Types........ 147
9.1.2 Type Casting........ 149
9.1.3 Boolean Type........ 150
9.1.4 Implicit Conversions........ 150

9.1.5 Pointers 151
9.1.6 Arrays 153
9.1.7 Arrays as Function Arguments 153
9.1.8 Designated Initializers in Arrays 154
9.1.9 Type Aliases 155
9.1.10 The Main Function Revisited 156
9.1.11 Operator sizeof 157
9.1.12 Const Types 158
9.1.13 Strings 160
9.1.14 Functional Types 160
9.1.15 Coding Well 162
9.1.16 Assignment: Scalar Product 166
9.1.17 Assignment: Prime Number Checker 167

9.2 Tagged Types 167

9.2.1 Structures 167
9.2.2 Unions 169
9.2.3 Anonymous Structures and Unions 170
9.2.4 Enumerations 171

9.3 Data Types in Programming Languages 172

9.3.1 Kinds of Typing 172
9.3.2 Polymorphism 174

9.4 Polymorphism in C 175

9.4.1 Parametric Polymorphism 175
9.4.2 Inclusion 177
9.4.3 Overloading 178
9.4.4 Coercions 179

9.5 Summary 179

Chapter 10: Code Structure 181

10.1 Declarations and Definitions 181

10.1.1 Function Declarations 182
10.1.2 Structure Declarations 183

10.2 Accessing Code from Other Files 184
10.2.1 Functions from Other Files 184
10.2.2 Data in Other Files 185
10.2.3 Header Files 187
10.3 Standard Library 188
10.4 Preprocessor 190
10.4.1 Include Guard 192
10.4.2 Why Is Preprocessor Evil? 194
10.5 Example: Sum of a Dynamic Array 195
10.5.1 Sneak Peek into Dynamic Memory Allocation 195
10.5.2 Example 195
10.6 Assignment: Linked List 197
10.6.1 Assignment 197
10.7 The Static Keyword 198
10.8 Linkage 199
10.9 Summary 200
Chapter 11: Memory 201
11.1 Pointers Revisited 201
11.1.1 Why Do We Need Pointers? 201
11.1.2 Pointer Arithmetic 202
11.1.3 The void* Type 203
11.1.4 NULL 203
11.1.5 A Word on ptrdiff_t 204
11.1.6 Function Pointers 205
11.2 Memory Model 206
11.2.1 Memory Allocation 207
11.3 Arrays and Pointers 209
11.3.1 Syntax Details 210
11.4 String Literals 211
11.4.1 String Interning 213

11.5 Data Models 213
11.6 Data Streams 215
11.7 Assignment: Higher-Order Functions and Lists 217
11.7.1 Common Higher-Order Functions 217
11.7.2 Assignment 218
11.8 Summary 220
Chapter 12: Syntax, Semantics, and Pragmatics 221
12.1 What Is a Programming Language? 221
12.2 Syntax and Formal Grammars 222
12.2.1 Example: Natural Numbers 223
12.2.2 Example: Simple Arithmetics 224
12.2.3 Recursive Descent 224
12.2.4 Example: Arithmetics with Priorities 227
12.2.5 Example: Simple Imperative Language 229
12.2.6 Chomsky Hierarchy 229
12.2.7 Abstract Syntax Tree 230
12.2.8 Lexical Analysis 231
12.2.9 Summary on Parsing 231
12.3 Semantics 231
12.3.1 Undefined Behavior 232
12.3.2 Unspecified Behavior 233
12.3.3 Implementation-Defined Behavior 234
12.3.4 Sequence Points 234
12.4 Pragmatics 235
12.4.1 Alignment 235
12.4.2 Data Structure Padding 235
12.5 Alignment in C11 238
12.6 Summary 239

Chapter 13: Good Code Practices 241
13.1 Making Choices 241
13.2 Code Elements 242
13.2.1 General Naming 242
13.2.2 File Structure 243
13.2.3 Types 243
13.2.4 Variables 244
13.2.5 On Global Variables 245
13.2.6 Functions 246
13.3 Files and Documentation 246
13.4 Encapsulation 248
13.5 Immutability 251
13.6 Assertions 251
13.7 Error Handling 252
13.8 On Memory Allocation 254
13.9 On Flexibility 255
13.10 Assignment: Image Rotation 256
13.10.1 BMP File Format 256
13.10.2 Architecture 258
13.11 Assignment: Custom Memory Allocator 259
13.12 Summary 262

Part III: Between C and Assembly 263

Chapter 14: Translation Details 265
14.1 Function Calling Sequence 265
14.1.1 XMM Registers 265
14.1.2 Calling Convention 266
14.1.3 Example: Simple Function and Its Stack 268
14.1.4 Red Zone 271
14.1.5 Variable Number of Arguments 271
14.1.6 vprintf and Friends 273

14.2 volatile 273
14.2.1 Lazy Memory Allocation 274
14.2.2 Generated Code 274
14.3 Non-Local jumps–setjmp 276
14.3.1 Volatile and setjmp 277
14.4 inline 280
14.5 restrict 281
14.6 Strict Aliasing 283
14.7 Security Issues 284
14.7.1 Stack Buffer Overrun 284
14.7.2 return-to-libc 285
14.7.3 Format Output Vulnerabilities 285
14.8 Protection Mechanisms 287
14.8.1 Security Cookie 287
14.8.2 Address Space Layout Randomization 288
14.8.3 DEP 288
14.9 Summary 288
Chapter 15: Shared Objects and Code Models 291
15.1 Dynamic Loading 291
15.2 Relocations and PIC 293
15.3 Example: Dynamic Library in C 293
15.4 GOT and PLT 294
15.4.1 Accessing External Variables 294
15.4.2 Calling External Functions 297
15.4.3 PLT Example 299
15.5 Preloading 301
15.6 Symbol Addressing Summary 302
15.7 Examples 303
15.7.1 Calling a Function 303
15.7.2 On Various Dynamic Linkers 305

15.7.3 Accessing an External Variable.......306
15.7.4 Complete Assembly Example.......307
15.7.5 Mixing C and Assembly.......308
15.8 Which Objects Are Linked?.......310
15.9 Optimizations.......313
15.10 Code Models.......315
15.10.1 Small Code Model (No PIC).......317
15.10.2 Large Code Model (No PIC).......318
15.10.3 Medium Code Model (No PIC).......318
15.10.4 Small PIC Code Model.......319
15.10.5 Large PIC Code Model.......320
15.10.6 Medium PIC Code Model.......322
15.11 Summary.......324
Chapter 16: Performance.......327
16.1 Optimizations.......327
16.1.1 Myth About Fast Languages.......327
16.1.2 General Advice.......328
16.1.3 Omit Stack Frame Pointer.......329
16.1.4 Tail recursion.......330
16.1.5 Common Subexpressions Elimination.......333
16.1.6 Constant Propagation.......334
16.1.7 (Named) Return Value Optimization.......336
16.1.8 Influence of Branch Prediction.......338
16.1.9 Influence of Execution Units.......338
16.1.10 Grouping Reads and Writes in Code.......340
16.2 Caching.......340
16.2.1 How Do We use Cache Effectively?.......340
16.2.2 Prefetching.......341
16.2.3 Example: Binary Search with Prefetching.......342
16.2.4 Bypassing Cache.......345
16.2.5 Example: Matrix Initialization.......346

16.3 SIMD Instruction Class ... 348
16.4 SSE and AVX Extensions ... 349
16.4.1 Assignment: Sepia Filter ... 351
16.5 Summary ... 354

Chapter 17: Multithreading ... 357

17.1 Processes and Threads ... 357
17.2 What Makes Multithreading Hard? ... 358
17.3 Execution Order ... 358
17.4 Strong and Weak Memory Models ... 359
17.5 Reordering Example ... 360
17.6 What Is Volatile and What Is Not ... 362
17.7 Memory Barriers ... 363
17.8 Introduction to pthreads ... 365
17.8.1 When to Use Multithreading ... 365
17.8.2 Creating Threads ... 366
17.8.3 Managing Threads ... 369
17.8.4 Example: Distributed Factorization ... 370
17.8.5 Mutexes ... 374
17.8.6 Deadlocks ... 377
17.8.7 Livelocks ... 378
17.8.8 Condition Variables ... 379
17.8.9 Spinlocks ... 381
17.9 Semaphores ... 382
17.10 How Strong Is Intel 64? ... 385
17.11 What Is Lock-Free Programming? ... 388
17.12 C11 Memory Model ... 390
17.12.1 Overview ... 390
17.12.2 Atomics ... 390
17.12.3 Memory Orderings in C11 ... 392
17.12.4 Operations ... 392
17.13 Summary ... 394

Part IV: Appendices 397

Chapter 18: Appendix A. Using gdb 399

Chapter 19: Appendix B. Using Make 409

19.1 Simple Makefile 409

19.2 Throwing in Variables 410

19.3 Automatic Variables 412

Chapter 20: Appendix C. System Calls 415

20.1 read 415

20.1.1 Arguments 416

20.2 write 416

20.2.1 Arguments 416

20.3 open 416

20.3.1 Arguments 417

20.3.2 Flags 417

20.4 close 417

20.4.1 Arguments 418

20.5 mmap 418

20.5.1 Arguments 418

20.5.2 Protection Flags 419

20.5.3 Behavior Flags 419

20.6 munmap 419

20.6.1 Arguments 419

20.7 exit 420

20.7.1 Arguments 420

Chapter 21: Appendix D. Performance Tests Information 421

Chapter 22: Bibliography 425

Index 429