

Preface	xi
Preface to the First Edition	xvii

1 Welcome Aboard 1

1.1	What We Will Try to Do	1
1.2	How We Will Get There	2
1.3	Two Recurring Themes	3
1.3.1	The Notion of Abstraction	3
1.3.2	Hardware versus Software	5
1.4	A Computer System	7
1.5	Two Very Important Ideas	9
1.6	Computers as Universal Computational Devices	9
1.7	How Do We Get the Electrons to Do the Work?	12
1.7.1	The Statement of the Problem	13
1.7.2	The Algorithm	13
1.7.3	The Program	14
1.7.4	The ISA	14
1.7.5	The Microarchitecture	15
1.7.6	The Logic Circuit	16
1.7.7	The Devices	16
1.7.8	Putting It Together	16
	Exercises	17

2 Bits, Data Types, and Operations 21

2.1	Bits and Data Types	21
2.1.1	The Bit as the Unit of Information	21
2.1.2	Data Types	22
2.2	Integer Data Types	23
2.2.1	Unsigned Integers	23
2.2.2	Signed Integers	23
2.3	2's Complement Integers	25
2.4	Binary-Decimal Conversion	27

2.4.1	Binary to Decimal Conversion	27
2.4.2	Decimal to Binary Conversion	28
2.5	Operations on Bits—Part I: Arithmetic	29
2.5.1	Addition and Subtraction	29
2.5.2	Sign-Extension	30
2.5.3	Overflow	31
2.6	Operations on Bits—Part II: Logical Operations	33
2.6.1	The AND Function	33
2.6.2	The OR Function	34
2.6.3	The NOT Function	35
2.6.4	The Exclusive-OR Function	35
2.7	Other Representations	36
2.7.1	The Bit Vector	36
2.7.2	Floating Point Data Type	37
2.7.3	ASCII Codes	40
2.7.4	Hexadecimal Notation	41
	Exercises	43

3 Digital Logic Structures 51

3.1	The Transistor	51
3.2	Logic Gates	53
3.2.1	The NOT Gate (Inverter)	53
3.2.2	OR and NOR Gates	54
3.2.3	AND and NAND Gates	56
3.2.4	DeMorgan's Law	58
3.2.5	Larger Gates	58
3.3	Combinational Logic Circuits	59
3.3.1	Decoder	59
3.3.2	Mux	60
3.3.3	Full Adder	61
3.3.4	The Programmable Logic Array (PLA)	63
3.3.5	Logical Completeness	64
3.4	Basic Storage Elements	64
3.4.1	The R-S Latch	64
3.4.2	The Gated D Latch	66
3.4.3	A Register	66

3.5	The Concept of Memory	67
3.5.1	Address Space	68
3.5.2	Addressability	68
3.5.3	A 2 ² -by-3-Bit Memory	68
3.6	Sequential Logic Circuits	70
3.6.1	A Simple Example: The Combination Lock	71
3.6.2	The Concept of State	72
3.6.3	Finite State Machines	74
3.6.4	An Example: The Complete Implementation of a Finite State Machine	77
3.7	The Data Path of the LC-3	80
	Exercises	82

4 The von Neumann Model 97

4.1	Basic Components	97
4.1.1	Memory	98
4.1.2	Processing Unit	99
4.1.3	Input and Output	100
4.1.4	Control Unit	100
4.2	The LC-3: An Example von Neumann Machine	101
4.3	Instruction Processing	103
4.3.1	The Instruction	103
4.3.2	The Instruction Cycle	104
4.4	Changing the Sequence of Execution	107
4.4.1	Control of the Instruction Cycle	108
4.5	Stopping the Computer	110
	Exercises	111

5 The LC-3 115

5.1	The ISA: Overview	115
5.1.1	Memory Organization	116
5.1.2	Registers	116
5.1.3	The Instruction Set	117
5.1.4	Opcodes	117
5.1.5	Data Types	118
5.1.6	Addressing Modes	118
5.1.7	Condition Codes	120
5.2	Operate Instructions	120
5.3	Data Movement Instructions	123
5.3.1	PC-Relative Mode	124
5.3.2	Indirect Mode	125
5.3.3	Base+offset Mode	127
5.3.4	Immediate Mode	128
5.3.5	An Example	129

5.4	Control Instructions	130
5.4.1	Conditional Branches	131
5.4.2	An Example	132
5.4.3	Two Methods for Loop Control	135
5.4.4	Example: Adding a Column of Numbers Using a Sentinel	135
5.4.5	The JMP Instruction	136
5.4.6	The TRAP Instruction	137
5.5	Another Example: Counting Occurrences of a Character	138
5.6	The Data Path Revisited	141
5.6.1	Basic Components of the Data Path	141
5.6.2	The Instruction Cycle	144
	Exercises	145

6 Programming 155

6.1	Problem Solving	155
6.1.1	Systematic Decomposition	155
6.1.2	The Three Constructs: Sequential, Conditional, Iterative	156
6.1.3	LC-3 Control Instructions to Implement the Three Constructs	157
6.1.4	The Character Count Example from Chapter 5, Revisited	158
6.2	Debugging	162
6.2.1	Debugging Operations	163
6.2.2	Examples: Use of the Interactive Debugger	164
	Exercises	172

7 Assembly Language 177

7.1	Assembly Language Programming—Moving Up a Level	177
7.2	An Assembly Language Program	178
7.2.1	Instructions	179
7.2.2	Pseudo-ops (Assembler Directives)	182
7.2.3	Example: The Character Count Example of Section 5.5, Revisited	183
7.3	The Assembly Process	185
7.3.1	Introduction	185
7.3.2	A Two-Pass Process	185
7.3.3	The First Pass: Creating the Symbol Table	186
7.3.4	The Second Pass: Generating the Machine Language Program	187

- 7.4 Beyond the Assembly of a Single Assembly Language Program 188
 - 7.4.1 The Executable Image 189
 - 7.4.2 More than One Object File 189
- Exercises 190

8 I/O 199

- 8.1 I/O Basics 199
 - 8.1.1 Device Registers 199
 - 8.1.2 Memory-Mapped I/O versus Special Input/Output Instructions 200
 - 8.1.3 Asynchronous versus Synchronous 200
 - 8.1.4 Interrupt-Driven versus Polling 202
- 8.2 Input from the Keyboard 202
 - 8.2.1 Basic Input Registers (the KBDR and the KBSR) 202
 - 8.2.2 The Basic Input Service Routine 202
 - 8.2.3 Implementation of Memory-Mapped Input 203
- 8.3 Output to the Monitor 204
 - 8.3.1 Basic Output Registers (the DDR and the DSR) 204
 - 8.3.2 The Basic Output Service Routine 205
 - 8.3.3 Implementation of Memory-Mapped Output 206
 - 8.3.4 Example: Keyboard Echo 207
- 8.4 A More Sophisticated Input Routine 207
- 8.5 Interrupt-Driven I/O 209
 - 8.5.1 What Is Interrupt-Driven I/O? 209
 - 8.5.2 Why Have Interrupt-Driven I/O? 210
 - 8.5.3 Generation of the Interrupt Signal 211
- 8.6 Implementation of Memory-Mapped I/O, Revisited 214
- Exercises 215

9 TRAP Routines and Subroutines 219

- 9.1 LC-3 TRAP Routines 219
 - 9.1.1 Introduction 219
 - 9.1.2 The TRAP Mechanism 220
 - 9.1.3 The TRAP Instruction 221
 - 9.1.4 The Complete Mechanism 222

- 9.1.5 TRAP Routines for Handling I/O 225
- 9.1.6 TRAP Routine for Halting the Computer 225
- 9.1.7 Saving and Restoring Registers 229
- 9.2 Subroutines 230
 - 9.2.1 The Call/Return Mechanism 230
 - 9.2.2 The JSR(R) Instruction 232
 - 9.2.3 The TRAP Routine for Character Input, Revisited 233
 - 9.2.4 PUTS: Writing a Character String to the Monitor 235
 - 9.2.5 Library Routines 235
- Exercises 240

10 And, Finally . . . The Stack 251

- 10.1 The Stack: Its Basic Structure 251
 - 10.1.1 The Stack—An Abstract Data Type 251
 - 10.1.2 Two Example Implementations 252
 - 10.1.3 Implementation in Memory 253
 - 10.1.4 The Complete Picture 257
- 10.2 Interrupt-Driven I/O (Part 2) 258
 - 10.2.1 Initiate and Service the Interrupt 259
 - 10.2.2 Return from the Interrupt 261
 - 10.2.3 An Example 262
- 10.3 Arithmetic Using a Stack 264
 - 10.3.1 The Stack as Temporary Storage 264
 - 10.3.2 An Example 265
 - 10.3.3 OpAdd, OpMult, and OpNeg 265
- 10.4 Data Type Conversion 272
 - 10.4.1 Example: The Bogus Program: $2 + 3 = e$ 272
 - 10.4.2 ASCII to Binary 273
 - 10.4.3 Binary to ASCII 276
- 10.5 Our Final Example: The Calculator 278
- Exercises 283

11 Introduction to Programming in C 289

- 11.1 Our Objective 289
- 11.2 Bridging the Gap 290
- 11.3 Translating High-Level Language Programs 292

- 11.3.1 Interpretation 292
- 11.3.2 Compilation 293
- 11.3.3 Pros and Cons 293
- 11.4 The C Programming Language 293
 - 11.4.1 The C Compiler 295
- 11.5 A Simple Example 297
 - 11.5.1 The Function `main` 297
 - 11.5.2 Formatting, Comments, and Style 299
 - 11.5.3 The C Preprocessor 300
 - 11.5.4 Input and Output 301
- 11.6 Summary 304
- Exercises 305

12 Variables and Operators 307

- 12.1 Introduction 307
- 12.2 Variables 308
 - 12.2.1 Three Basic Data Types: `int`, `char`, `double` 308
 - 12.2.2 Choosing Identifiers 310
 - 12.2.3 Scope: Local versus Global 311
 - 12.2.4 More Examples 313
- 12.3 Operators 314
 - 12.3.1 Expressions and Statements 315
 - 12.3.2 The Assignment Operator 316
 - 12.3.3 Arithmetic Operators 317
 - 12.3.4 Order of Evaluation 318
 - 12.3.5 Bitwise Operators 319
 - 12.3.6 Relational Operators 320
 - 12.3.7 Logical Operators 322
 - 12.3.8 Increment/Decrement Operators 322
 - 12.3.9 Expressions with Multiple Operators 324
- 12.4 Problem Solving Using Operators 324
- 12.5 Tying it All Together 326
 - 12.5.1 Symbol Table 326
 - 12.5.2 Allocating Space for Variables 328
 - 12.5.3 A Comprehensive Example 331
- 12.6 Additional Topics 332
 - 12.6.1 Variations of the Three Basic Types 332
 - 12.6.2 Literals, Constants, and Symbolic Values 334
 - 12.6.3 Storage Class 335
 - 12.6.4 Additional C Operators 336
- 12.7 Summary 337
- Exercises 338

13 Control Structures 343

- 13.1 Introduction 343
- 13.2 Conditional Constructs 344
 - 13.2.1 The `if` Statement 344
 - 13.2.2 The `if-else` Statement 347
- 13.3 Iteration Constructs 350
 - 13.3.1 The `while` Statement 350
 - 13.3.2 The `for` Statement 353
 - 13.3.3 The `do-while` Statement 358
- 13.4 Problem Solving Using Control Structures 359
 - 13.4.1 Problem 1: Approximating the Value of π 360
 - 13.4.2 Problem 2: Finding Prime Numbers Less than 100 362
 - 13.4.3 Problem 3: Analyzing an E-mail Address 366
- 13.5 Additional C Control Structures 368
 - 13.5.1 The `switch` Statement 368
 - 13.5.2 The `break` and `continue` Statements 370
 - 13.5.3 An Example: Simple Calculator 370
- 13.6 Summary 372
- Exercises 372

14 Functions 379

- 14.1 Introduction 379
- 14.2 Functions in C 380
 - 14.2.1 A Function with a Parameter 380
 - 14.2.2 Example: Area of a Ring 384
- 14.3 Implementing Functions in C 385
 - 14.3.1 Run-Time Stack 385
 - 14.3.2 Getting It All to Work 388
 - 14.3.3 Tying It All Together 393
- 14.4 Problem Solving Using Functions 394
 - 14.4.1 Problem 1: Case Conversion 395
 - 14.4.2 Problem 2: Pythagorean Triples 397
- 14.5 Summary 398
- Exercises 399

15 Testing and Debugging 407

- 15.1 Introduction 407
- 15.2 Types of Errors 408
 - 15.2.1 Syntactic Errors 409

- 15.2.2 Semantic Errors 409
- 15.2.3 Algorithmic Errors 411
- 15.3 Testing 412
 - 15.3.1 Black-Box Testing 412
 - 15.3.2 White-Box Testing 413
- 15.4 Debugging 414
 - 15.4.1 Ad Hoc Techniques 414
 - 15.4.2 Source-Level Debuggers 415
- 15.5 Programming for Correctness 417
 - 15.5.1 Nailing Down the Specification 417
 - 15.5.2 Modular Design 418
 - 15.5.3 Defensive Programming 418
- 15.6 Summary 419
- Exercises 421

16 Pointers and Arrays 427

- 16.1 Introduction 427
- 16.2 Pointers 428
 - 16.2.1 Declaring Pointer Variables 429
 - 16.2.2 Pointer Operators 430
 - 16.2.3 Passing a Reference Using Pointers 432
 - 16.2.4 Null Pointers 433
 - 16.2.5 Demystifying the Syntax 434
 - 16.2.6 An Example Problem Involving Pointers 434
- 16.3 Arrays 436
 - 16.3.1 Declaring and Using Arrays 436
 - 16.3.2 Examples Using Arrays 438
 - 16.3.3 Arrays as Parameters 440
 - 16.3.4 Strings in C 441
 - 16.3.5 The Relationship Between Arrays and Pointers in C 446
 - 16.3.6 Problem Solving: Insertion Sort 446
 - 16.3.7 Common Pitfalls with Arrays in C 449
- 16.4 Summary 451
- Exercises 451

17 Recursion 457

- 17.1 Introduction 457
- 17.2 What Is Recursion? 458
- 17.3 Recursion versus Iteration 459
- 17.4 Towers of Hanoi 460

- 17.5 Fibonacci Numbers 464
- 17.6 Binary Search 468
- 17.7 Integer to ASCII 471
- 17.8 Summary 473
- Exercises 473

18 I/O in C 481

- 18.1 Introduction 481
- 18.2 The C Standard Library 481
- 18.3 I/O, One Character at a Time 482
 - 18.3.1 I/O Streams 482
 - 18.3.2 `putchar` 483
 - 18.3.3 `getchar` 483
 - 18.3.4 Buffered I/O 483
- 18.4 Formatted I/O 485
 - 18.4.1 `printf` 485
 - 18.4.2 `scanf` 487
 - 18.4.3 Variable Argument Lists 489
- 18.5 I/O from Files 491
- 18.6 Summary 493
- Exercises 494

19 Data Structures 497

- 19.1 Introduction 497
- 19.2 Structures 498
 - 19.2.1 `typedef` 500
 - 19.2.2 Implementing Structures in C 501
- 19.3 Arrays of Structures 502
- 19.4 Dynamic Memory Allocation 504
 - 19.4.1 Dynamically Sized Arrays 506
- 19.5 Linked Lists 508
 - 19.5.1 An Example 510
- 19.6 Summary 516
- Exercises 517

A The LC-3 ISA 521

- A.1 Overview 521
- A.2 Notation 523
- A.3 The Instruction Set 523
- A.4 Interrupt and Exception Processing 543
 - A.4.1 Interrupts 543
 - A.4.2 Exceptions 544

B From LC-3 to x86 547

- B.1 LC-3 Features and Corresponding x86 Features 548
 - B.1.1 Instruction Set 548
 - B.1.2 Memory 553
 - B.1.3 Internal State 553
- B.2 The Format and Specification of x86 Instructions 557
 - B.2.1 Prefix 558
 - B.2.2 Opcode 559
 - B.2.3 ModR/M Byte 559
 - B.2.4 SIB Byte 560
 - B.2.5 Displacement 560
 - B.2.6 Immediate 560
- B.3 An Example 562

C The Microarchitecture of the LC-3 565

- C.1 Overview 565
- C.2 The State Machine 567
- C.3 The Data Path 569
- C.4 The Control Structure 569
- C.5 Memory-Mapped I/O 575
- C.6 Interrupt and Exception Control 576
 - C.6.1 Initiating an Interrupt 579
 - C.6.2 Returning from an Interrupt, RTI 581
 - C.6.3 The Illegal Opcode Exception 582
- C.7 Control Store 583

D The C Programming Language 585

- D.1 Overview 585
- D.2 C Conventions 585
 - D.2.1 Source Files 585
 - D.2.2 Header Files 585
 - D.2.3 Comments 586
 - D.2.4 Literals 586
 - D.2.5 Formatting 588
 - D.2.6 Keywords 588
- D.3 Types 589
 - D.3.1 Basic Data Types 589
 - D.3.2 Type Qualifiers 590
 - D.3.3 Storage Class 591
 - D.3.4 Derived Types 592
 - D.3.5 `typedef` 594

- D.4 Declarations 595
 - D.4.1 Variable Declarations 595
 - D.4.2 Function Declarations 596
- D.5 Operators 596
 - D.5.1 Assignment Operators 597
 - D.5.2 Arithmetic Operators 597
 - D.5.3 Bit-wise Operators 598
 - D.5.4 Logical Operators 598
 - D.5.5 Relational Operators 599
 - D.5.6 Increment/Decrement Operators 599
 - D.5.7 Conditional Expression 600
 - D.5.8 Pointer, Array, and Structure Operators 600
 - D.5.9 `sizeof` 601
 - D.5.10 Order of Evaluation 602
 - D.5.11 Type Conversions 602
- D.6 Expressions and Statements 603
 - D.6.1 Expressions 603
 - D.6.2 Statements 604
- D.7 Control 604
 - D.7.1 If 604
 - D.7.2 If-else 605
 - D.7.3 Switch 605
 - D.7.4 While 606
 - D.7.5 For 607
 - D.7.6 Do-while 607
 - D.7.7 Break 608
 - D.7.8 `continue` 608
 - D.7.9 `return` 609
- D.8 The C Preprocessor 609
 - D.8.1 Macro substitution 609
 - D.8.2 File inclusion 610
- D.9 Some Standard Library Functions 610
 - D.9.1 I/O Functions 611
 - D.9.2 String Functions 612
 - D.9.3 Math Functions 613
 - D.9.4 Utility Functions 613

E Useful Tables 615

- E.1 Commonly Used Numerical Prefixes 615
- E.2 Standard ASCII codes 616
- E.3 Powers of 2 617

F Solutions to Selected Exercises 619