

# CONTENTS IN DETAIL

## ACKNOWLEDGMENTS

xxiii

## INTRODUCTION

xxv

0.1 A Brief History of the ARM CPU . . . . .	xxvi
0.2 Why Learn ARM Assembly? . . . . .	xxvii
0.3 Why Learn 64-Bit ARM? . . . . .	xxviii
0.4 Expectations and Prerequisites . . . . .	xxix
0.5 Source Code . . . . .	xxx
0.6 Typography and Pedantry . . . . .	xxxii
0.7 Organization . . . . .	xxxii

## PART I: MACHINE ORGANIZATION

1

### 1

#### HELLO, WORLD OF ASSEMBLY LANGUAGE

3

1.1 What You'll Need . . . . .	4
1.1.1 Setting Up Gas . . . . .	4
1.1.2 Setting Up a Text Editor . . . . .	4
1.1.3 Understanding C/C++ Examples . . . . .	4
1.2 The Anatomy of an Assembly Language Program . . . . .	5
1.3 Running Your First Assembly Language Program . . . . .	7
1.4 Running Your First Gas/C++ Hybrid Program . . . . .	8
1.5 The <code>aaa.inc</code> Include File . . . . .	10
1.6 The ARM64 CPU Architecture . . . . .	11
1.6.1 ARM CPU Registers . . . . .	11
1.6.2 The Memory Subsystem . . . . .	14
1.7 Declaring Memory Variables in Gas . . . . .	16
1.7.1 Associating Memory Addresses with Variables . . . . .	19
1.7.2 Aligning Variables . . . . .	19
1.7.3 Declaring Named Constants in Gas . . . . .	21
1.7.4 Creating Register Aliases in Gas and Substituting Text . . . . .	22
1.8 Basic ARM Assembly Language Instructions . . . . .	22
1.8.1 <code>ldr</code> , <code>str</code> , <code>adr</code> , and <code>adrp</code> . . . . .	23
1.8.2 <code>mov</code> . . . . .	27
1.8.3 <code>add</code> and <code>sub</code> . . . . .	28
1.8.4 <code>bl</code> , <code>blr</code> , and <code>ret</code> . . . . .	29
1.9 The ARM64 Application Binary Interface . . . . .	30
1.9.1 Register Usage . . . . .	31
1.9.2 Parameter Passing and Function Result Conventions . . . . .	32
1.10 Calling C Library Functions . . . . .	33
1.10.1 Assembling Programs Under Multiple OSes . . . . .	36
1.10.2 Writing a "Hello, World!" Program . . . . .	40

1.11	Moving On . . . . .	43
1.12	For More Information . . . . .	43

## 2

### **DATA REPRESENTATION AND OPERATIONS 45**

2.1	Numbering Systems . . . . .	46
	2.1.1 Decimal . . . . .	46
	2.1.2 Binary . . . . .	46
	2.1.3 Hexadecimal. . . . .	48
2.2	Numbers vs. Representation . . . . .	50
2.3	Data Organization. . . . .	53
	2.3.1 Bits. . . . .	53
	2.3.2 Nibbles . . . . .	54
	2.3.3 Bytes . . . . .	54
	2.3.4 Half Words . . . . .	55
	2.3.5 Words . . . . .	56
	2.3.6 Double Words and Quad Words . . . . .	57
2.4	Logical Operations on Bits . . . . .	58
	2.4.1 AND . . . . .	58
	2.4.2 OR. . . . .	59
	2.4.3 XOR. . . . .	59
	2.4.4 NOT . . . . .	60
2.5	Logical Operations on Binary Numbers and Bit Strings. . . . .	60
2.6	Signed and Unsigned Numbers. . . . .	65
2.7	Sign Extension and Zero Extension. . . . .	71
2.8	Sign Contraction and Saturation . . . . .	72
2.9	Loading and Storing Byte and Half-Word Values . . . . .	72
2.10	Control-Transfer Instructions . . . . .	74
	2.10.1 Branch . . . . .	75
	2.10.2 Instructions That Affect the Condition Code Flags. . . . .	76
	2.10.3 Conditional Branch . . . . .	77
	2.10.4 cmp and Corresponding Conditional Branches . . . . .	78
2.11	Shifts and Rotates . . . . .	82
2.12	Bit Fields and Packed Data . . . . .	85
2.13	IEEE Floating-Point Formats . . . . .	93
	2.13.1 Single-Precision Format. . . . .	94
	2.13.2 Double-Precision Format . . . . .	95
2.14	Normalized Floating-Point Values. . . . .	96
	2.14.1 Nonnumeric Values . . . . .	97
	2.14.2 Gas Support for Floating-Point Values . . . . .	97
2.15	Binary-Coded Decimal Representation . . . . .	98
2.16	Characters . . . . .	99
	2.16.1 The ASCII Character Encoding . . . . .	99
	2.16.2 Gas Support for ASCII Characters . . . . .	101
2.17	Gas Support for the Unicode Character Set. . . . .	102
2.18	Machine Code . . . . .	103
2.19	Operand2 . . . . .	106
	2.19.1 #immediate. . . . .	107
	2.19.2 #pattern . . . . .	107
	2.19.3 Register . . . . .	109
	2.19.4 Shifted Register . . . . .	109
	2.19.5 Extending Register . . . . .	110

2.20	Large Constants . . . . .	111
2.20.1	movz . . . . .	112
2.20.2	movk . . . . .	112
2.20.3	movn . . . . .	113
2.21	Moving On . . . . .	113
2.22	For More Information . . . . .	114

### **3 MEMORY ACCESS AND ORGANIZATION 119**

3.1	Runtime Memory Organization . . . . .	120
3.1.1	The .text Section . . . . .	121
3.1.2	The .data Section . . . . .	122
3.1.3	Read-Only Data Sections . . . . .	122
3.1.4	The .bss Section . . . . .	124
3.1.5	The .section Directive . . . . .	126
3.1.6	Declaration Sections . . . . .	126
3.1.7	Memory Access and MMU Pages . . . . .	127
3.1.8	PIE and ASLR . . . . .	128
3.1.9	The .pool Section . . . . .	130
3.2	Gas Storage Allocation for Variables . . . . .	131
3.3	Little-Endian and Big-Endian Data Organization . . . . .	133
3.4	Memory Access . . . . .	135
3.5	Gas Support for Data Alignment . . . . .	138
3.6	The ARM Memory Addressing Modes . . . . .	140
3.6.1	PC-Relative . . . . .	141
3.6.2	Register-Indirect . . . . .	142
3.6.3	Indirect-Plus-Offset . . . . .	143
3.6.4	Scaled Indirect-Plus-Offset . . . . .	143
3.6.5	Pre-indexed . . . . .	144
3.6.6	Post-indexed . . . . .	145
3.6.7	Scaled-indexed . . . . .	146
3.7	Address Expressions . . . . .	149
3.8	Getting the Address of a Memory Object . . . . .	153
3.9	The Push and Pop Operations . . . . .	155
3.9.1	Using Double Loads and Stores . . . . .	155
3.9.2	Executing the Basic Push Operation . . . . .	156
3.9.3	Executing the Basic Pop Operation . . . . .	157
3.9.4	Preserving at Least Two Registers . . . . .	158
3.9.5	Preserving Register Values on the Stack . . . . .	159
3.9.6	Saving Function Return Addresses on the Stack . . . . .	160
3.10	Pushing and Popping Stack Data . . . . .	161
3.10.1	Removing Data from the Stack Without Popping It . . . . .	163
3.10.2	Accessing Data Pushed onto the Stack Without Popping It . . . . .	165
3.11	Moving On . . . . .	167
3.12	For More Information . . . . .	167

### **4 CONSTANTS, VARIABLES, AND DATA TYPES 169**

4.1	Gas Constant Declarations . . . . .	170
4.2	The Location Counter Operator . . . . .	171

4.3	Data Types and Gas	172
4.4	Pointer Data Types	173
4.4.1	Pointer Usage in Assembly Language	174
4.4.2	Pointer Declarations in Gas	175
4.4.3	Pointer Constants and Expressions	175
4.4.4	Pointer Variables and Dynamic Memory Allocation	178
4.4.5	Common Pointer Problems	180
4.5	Composite Data Types	186
4.6	Character Strings	187
4.6.1	Zero-Terminated Strings	187
4.6.2	Length-Prefixed Strings	188
4.6.3	String Descriptors	189
4.6.4	Pointers to Strings	190
4.6.5	String Functions	190
4.7	Arrays	194
4.7.1	Declaring Arrays in Gas Programs	195
4.7.2	Accessing Elements of a Single-Dimensional Array	197
4.7.3	Sorting an Array of Values	198
4.7.4	Implementing Multidimensional Arrays	203
4.8	Structs	212
4.8.1	Dealing with Limited Gas Support for Structs	214
4.8.2	Initializing Structs	217
4.8.3	Creating Arrays of Structs	218
4.8.4	Aligning Fields Within a Struct	219
4.9	Unions	220
4.10	Moving On	221
4.11	For More Information	221

## PART II: BASIC ASSEMBLY LANGUAGE

**225**

### **5 PROCEDURES**

**227**

5.1	Assembly Language Programming Style	228
5.2	Gas Procedures	230
5.2.1	Gas Local Labels	234
5.2.2	bl, ret, and br	235
5.3	Saving the State of the Machine	237
5.4	Call Trees, Leaf Procedures, and the Stack	242
5.4.1	Activation Records	244
5.4.2	Objects in the Activation Record	246
5.4.3	ARM ABI Parameter-Passing Conventions	247
5.4.4	Standard Entry Sequence	248
5.4.5	Standard Exit Sequence	250
5.5	Local Variables	250
5.5.1	Low-Level Implementation of Automatic Variables	251
5.5.2	The locals Macro	253
5.6	Parameters	255
5.6.1	Passing by Value	255
5.6.2	Passing by Reference	256

5.6.3	Using Low-Level Parameter Implementation . . . . .	258
5.6.4	Accessing Reference Parameters on the Stack . . . . .	271
5.7	Functions and Function Return Results . . . . .	276
5.8	Recursion . . . . .	277
5.9	Procedure Pointers and Procedural Parameters . . . . .	284
5.10	A Program-Defined Stack . . . . .	286
5.11	Moving On . . . . .	290
5.12	For More Information . . . . .	290

## 6

### ARITHMETIC

**293**

6.1	Additional ARM Arithmetic Instructions . . . . .	293
6.1.1	Multiplication . . . . .	294
6.1.2	Division and Modulo . . . . .	294
6.1.3	cmp Revisited . . . . .	295
6.1.4	Conditional Instructions. . . . .	297
6.2	Memory Variables vs. Registers . . . . .	299
6.2.1	Volatile vs. Nonvolatile Register Usage. . . . .	300
6.2.2	Global vs. Local Variables. . . . .	300
6.2.3	Easy Access to Global Variables . . . . .	301
6.3	Arithmetic Expressions . . . . .	303
6.3.1	Simple Assignments . . . . .	304
6.3.2	Simple Expressions. . . . .	305
6.3.3	Complex Expressions . . . . .	307
6.3.4	Commutative Operators . . . . .	311
6.4	Logical Expressions . . . . .	312
6.5	Conditional Comparisons and Boolean Expressions . . . . .	314
6.5.1	Implementing Conjunction Using ccmp . . . . .	315
6.5.2	Implementing Disjunction Using ccmp. . . . .	318
6.5.3	Handling Complex Boolean Expressions. . . . .	319
6.6	Machine and Arithmetic Idioms . . . . .	319
6.6.1	Multiplying Without mul . . . . .	319
6.6.2	Dividing Without sdiv or udiv . . . . .	321
6.6.3	Implementing Modulo-N Counters with AND . . . . .	322
6.6.4	Avoiding Needlessly Complex Machine Idioms . . . . .	322
6.7	Floating-Point and Finite-Precision Arithmetic . . . . .	322
6.7.1	Basic Floating-Point Terminology . . . . .	322
6.7.2	Limited-Precision Arithmetic and Accuracy. . . . .	323
6.7.3	Errors in Floating-Point Calculations . . . . .	324
6.7.4	Floating-Point Value Comparisons . . . . .	326
6.8	Floating-Point Arithmetic on the ARM . . . . .	327
6.8.1	Neon Registers . . . . .	327
6.8.2	Control Register. . . . .	330
6.8.3	Status Register. . . . .	331
6.9	Floating-Point Instructions . . . . .	332
6.9.1	FPU Data Movement Instructions . . . . .	332
6.9.2	FPU Arithmetic Instructions. . . . .	334
6.9.3	Floating-Point Comparisons . . . . .	336
6.9.4	Floating-Point Conversion Instructions . . . . .	343
6.10	The ARM ABI and Floating-Point Registers . . . . .	346
6.11	Using C Standard Library Math Functions . . . . .	347

6.12	Moving On . . . . .	352
6.13	For More Information . . . . .	352

## 7

### **LOW-LEVEL CONTROL STRUCTURES 355**

7.1	Statement Labels . . . . .	356
7.2	Initializing Arrays with Statement Labels . . . . .	356
7.3	Unconditional Transfer of Control . . . . .	357
7.4	Register-Indirect Branches . . . . .	358
7.5	Taking the Address of Symbols in Your Code . . . . .	364
7.5.1	Revisiting the lea Macro . . . . .	365
7.5.2	Statically Computing the Address of a Symbol . . . . .	365
7.5.3	Dynamically Computing the Address of a Memory Object . . . . .	367
7.5.4	Working with Veneers . . . . .	368
7.6	Implementing Common Control Structures in Assembly Language . . . . .	371
7.6.1	Decisions . . . . .	371
7.6.2	if...then...else Sequences . . . . .	372
7.6.3	Complex if Statements Using Complete Boolean Evaluation . . . . .	378
7.6.4	Short-Circuit Boolean Evaluation . . . . .	380
7.6.5	Short-Circuit vs. Complete Boolean Evaluation . . . . .	382
7.6.6	Efficient Implementation of if Statements in Assembly Language . . . . .	384
7.6.7	switch...case Statements . . . . .	389
7.7	State Machines and Indirect Jumps . . . . .	405
7.8	Loops . . . . .	415
7.8.1	while . . . . .	415
7.8.2	repeat...until . . . . .	417
7.8.3	forever/endif . . . . .	418
7.8.4	for . . . . .	419
7.8.5	break and continue . . . . .	420
7.8.6	ARM Looping Instructions . . . . .	425
7.8.7	Register Usage and Loops . . . . .	426
7.9	Loop Performance Improvements . . . . .	428
7.9.1	Moving the Termination Condition to the End of a Loop . . . . .	428
7.9.2	Executing the Loop Backward . . . . .	430
7.9.3	Eliminating Loop-Invariant Calculations . . . . .	431
7.9.4	Unraveling Loops . . . . .	432
7.9.5	Using Induction Variables . . . . .	433
7.10	Moving On . . . . .	434
7.11	For More Information . . . . .	435

## **PART III: ADVANCED ASSEMBLY LANGUAGE 439**

### 8

#### **ADVANCED ARITHMETIC 441**

8.1	Extended-Precision Operations . . . . .	441
8.1.1	Addition . . . . .	442
8.1.2	Subtraction . . . . .	445
8.1.3	Comparisons . . . . .	446
8.1.4	Multiplication . . . . .	450

8.1.5	Division	457
8.1.6	Negation	465
8.1.7	AND	465
8.1.8	OR	466
8.1.9	XOR	466
8.1.10	NOT	467
8.1.11	Shift Operations	467
8.2	Operating on Different-Size Operands	472
8.3	Moving On	475
8.4	For More Information	475

## **9**

### **NUMERIC CONVERSION** **477**

9.1	Converting Numeric Strings to Values	478
9.1.1	Numeric Values to Hexadecimal Strings	478
9.1.2	Extended-Precision Hexadecimal Values to Strings	494
9.1.3	Unsigned Decimal Values to Strings	495
9.1.4	Signed Integer Values to Strings	509
9.1.5	Extended-Precision Unsigned Integers to Strings	510
9.1.6	Formatted Conversions	517
9.2	Converting Floating-Point Values to Strings	529
9.2.1	Floating-Point Exponent to String of Decimal Digits	530
9.2.2	Floating-Point Mantissa to String of Digits	530
9.2.3	Strings in Decimal and Exponential Format	531
9.2.4	Double-Precision Values to Strings	531
9.3	String-to-Numeric Conversions	566
9.3.1	Decimal Strings to Integers	566
9.3.2	Hexadecimal Strings to Numeric Form	578
9.3.3	String to Floating-Point	588
9.4	Other Numeric Conversions	602
9.5	Moving On	602
9.6	For More Information	603

## **10**

### **TABLE LOOKUPS** **605**

10.1	Using Tables in Assembly Language	605
10.1.1	Function Computation via Table Lookup	606
10.1.2	Function Domains and Ranges	611
10.1.3	Domain Conditioning	614
10.1.4	Table Generation	615
10.2	Table-Lookup Performance	617
10.3	Moving On	618
10.4	For More Information	618

## **11**

### **NEON AND SIMD PROGRAMMING** **621**

11.1	The History of SIMD Instruction Extensions	622
11.2	Vector Registers	623
11.3	Vector Data Movement Instructions	625
11.3.1	Data Movement Between Registers	625
11.3.2	Vector Load Immediate Instructions	628

11.3.3	Register or Lane Value Duplication . . . . .	631
11.3.4	Vector Load and Store . . . . .	632
11.3.5	Interleaved Load and Store . . . . .	632
11.3.6	Register Interleaving and Deinterleaving . . . . .	639
11.3.7	Table Lookups with <code>tbl</code> and <code>tbx</code> . . . . .	644
11.3.8	Endian Swaps with <code>rev16</code> , <code>rev32</code> , and <code>rev64</code> . . . . .	646
11.4	Vertical and Horizontal Operations . . . . .	646
11.5	SIMD Logical Operations . . . . .	647
11.6	SIMD Shift Operations . . . . .	649
11.6.1	Shift-Left Instruction . . . . .	649
11.6.2	Saturating Shift Left . . . . .	650
11.6.3	Shift-Left Long . . . . .	651
11.6.4	Shift and Insert . . . . .	652
11.6.5	Signed and Unsigned Shift Right . . . . .	653
11.6.6	Accumulating Shift Right . . . . .	654
11.6.7	Narrowing Shift Right . . . . .	655
11.6.8	Saturating Shift Right with Narrowing . . . . .	655
11.6.9	Shift by a Variable Number of Bits . . . . .	657
11.7	SIMD Arithmetic Operations . . . . .	659
11.7.1	SIMD Addition . . . . .	659
11.7.2	Subtraction . . . . .	668
11.7.3	Absolute Difference . . . . .	669
11.7.4	Vector Multiplication . . . . .	671
11.7.5	Vector Division . . . . .	679
11.7.6	Sign Operations . . . . .	680
11.7.7	Minimum and Maximum . . . . .	681
11.8	Floating-Point and Integer Conversions . . . . .	683
11.8.1	Floating-Point to Integer . . . . .	683
11.8.2	Integer to Floating-Point . . . . .	684
11.8.3	Conversion Between Floating-Point Formats . . . . .	685
11.8.4	Floating-Point Values Rounded to the Nearest Integral . . . . .	686
11.9	Vector Square-Root Instructions . . . . .	686
11.10	Vector Comparisons . . . . .	687
11.10.1	Vector Integer Comparisons . . . . .	688
11.10.2	Vector Floating-Point Comparisons . . . . .	689
11.10.3	Vector Bit Test Instructions . . . . .	691
11.10.4	Vector Comparison Results . . . . .	691
11.11	A Sorting Example Using SIMD Code . . . . .	694
11.12	A Numeric-to-Hex-String Example Using SIMD Code . . . . .	698
11.13	Use of SIMD Instructions in Real Programs . . . . .	699
11.14	Moving On . . . . .	700
11.15	For More Information . . . . .	700

## **12 BIT MANIPULATION 703**

12.1	What Is Bit Data, Anyway? . . . . .	703
12.2	Instructions That Manipulate Bits . . . . .	704
12.2.1	Isolating, Clearing, and Testing Bits . . . . .	705
12.2.2	Setting and Inserting Bits . . . . .	706
12.2.3	Clearing Bits . . . . .	708
12.2.4	Inverting Bits . . . . .	709



12.2.5	Shift and Rotate . . . . .	709
12.2.6	Conditional Instructions. . . . .	711
12.2.7	Counting Bits. . . . .	711
12.2.8	Bit Reversal . . . . .	712
12.2.9	Bit Insertion and Selection . . . . .	713
12.2.10	Bit Extraction with ubfx . . . . .	713
12.2.11	Bit Movement with ubfiz . . . . .	714
12.2.12	Bit Movement with ubfm . . . . .	714
12.2.13	Bit Extraction with extr . . . . .	715
12.2.14	Bit Testing with tbz and tbnz . . . . .	715
12.3	Flag Modification by Arithmetic and Logical Instructions . . . . .	715
12.3.1	The Zero Flag . . . . .	716
12.3.2	The Negative Flag . . . . .	718
12.3.3	The Carry and Overflow Flags . . . . .	719
12.4	Packing and Unpacking Bit Strings. . . . .	719
12.4.1	Inserting One Bit String into Another . . . . .	719
12.4.2	Extracting a Bit String . . . . .	726
12.4.3	Clearing a Bit Field . . . . .	727
12.4.4	Using bfm . . . . .	728
12.5	Common Bit Operations . . . . .	728
12.5.1	Coalescing Bit Sets and Distributing Bit Strings . . . . .	729
12.5.2	Creating Packed Arrays of Bit Strings . . . . .	731
12.5.3	Searching for Bits. . . . .	734
12.5.4	Merging Bit Strings . . . . .	735
12.5.5	Scattering Bits from a Bit String . . . . .	735
12.5.6	Searching for a Bit Pattern . . . . .	736
12.6	Moving On . . . . .	738
12.7	For More Information . . . . .	739

## **13 MACROS AND THE GAS COMPILE-TIME LANGUAGE 741**

13.1	The Gas Compile-Time Language Interpreter . . . . .	742
13.2	The C/C++ Preprocessor . . . . .	742
13.2.1	The #warning and #error Directives . . . . .	743
13.2.2	Compile-Time Constant Definition with CPP . . . . .	744
13.2.3	CPP Compile-Time Expressions. . . . .	745
13.2.4	Conditional Assembly. . . . .	746
13.2.5	CPP Macros . . . . .	749
13.3	Components of the Gas CTL . . . . .	760
13.3.1	Errors and Warnings During Assembly . . . . .	760
13.3.2	Conditional Assembly. . . . .	760
13.3.3	Compile-Time Loops . . . . .	763
13.3.4	Gas Macros . . . . .	765
13.4	The aoaa.inc Header File . . . . .	771
13.5	Generating Macros by Another Macro . . . . .	787
13.6	Choosing Between Gas Macros and CPP Macros . . . . .	790
13.7	Moving On . . . . .	792
13.8	For More Information . . . . .	792

**14**  
**STRING OPERATIONS** **795**

14.1	Zero-Terminated Strings and Functions . . . . .	796
14.2	A String Format for Assembly Language Programmers . . . . .	801
14.2.1	Dynamic String Allocation . . . . .	803
14.2.2	String Copy Function . . . . .	818
14.2.3	String Comparison Function . . . . .	824
14.2.4	Substring Function . . . . .	836
14.2.5	More String Functions . . . . .	845
14.3	The Unicode Character Set . . . . .	845
14.3.1	Unicode History . . . . .	846
14.3.2	Code Points and Code Planes . . . . .	847
14.3.3	Surrogate Code Points . . . . .	847
14.3.4	Glyphs, Characters, and Grapheme Clusters . . . . .	848
14.3.5	Normal Forms and Canonical Equivalence . . . . .	849
14.3.6	Encodings . . . . .	850
14.3.7	Combining Characters . . . . .	852
14.4	Unicode in Assembly Language . . . . .	853
14.4.1	Writing Console Applications with UTF-8 Characters . . . . .	853
14.4.2	Using Unicode String Functions . . . . .	857
14.5	Moving On . . . . .	858
14.6	For More Information . . . . .	859

**15**  
**MANAGING COMPLEX PROJECTS** **861**

15.1	The .include Directive . . . . .	862
15.2	Ignoring Duplicate Include Operations . . . . .	863
15.3	Assembly Units and External Directives . . . . .	864
15.4	Creating a String Library with Separate Compilation . . . . .	866
15.5	Introducing Makefiles . . . . .	875
15.5.1	Basic Makefile Syntax . . . . .	876
15.5.2	Make Clean and Touch . . . . .	882
15.6	Generating Library Files with the Archiver Program . . . . .	883
15.7	Managing the Impact of Object Files on Program Size . . . . .	886
15.8	Moving On . . . . .	886
15.9	For More Information . . . . .	887

**16**  
**STAND-ALONE ASSEMBLY LANGUAGE PROGRAMS** **889**

16.1	Portability Issues with System Calls . . . . .	890
16.2	Stand-Alone Code and System Calls . . . . .	891
16.3	The svc Interface and OS Portability . . . . .	894
16.3.1	Call Numbers . . . . .	895
16.3.2	API Parameters . . . . .	897
16.3.3	API Error Handling . . . . .	898
16.4	A Stand-Alone "Hello, World!" Program . . . . .	899
16.5	A Sample File I/O Program . . . . .	901
16.5.1	volatiles.S Functions . . . . .	905
16.5.2	files.S File I/O Functions . . . . .	907

16.5.3	stdio.S Functions . . . . .	915
16.5.4	File I/O Demo Application . . . . .	922
16.6	Calling System Library Functions Under macOS. . . . .	926
16.7	Creating Assembly Applications Without GCC . . . . .	928
16.8	For More Information . . . . .	930

## **PART IV: REFERENCE MATERIALS 931**

### **A THE ASCII CHARACTER SET 933**

### **B GLOSSARY 939**

### **C INSTALLING AND USING GAS 945**

C.1	macOS . . . . .	946
C.2	Linux . . . . .	946

### **D THE BASH SHELL INTERPRETER 949**

D.1	Running Bash . . . . .	950
D.2	Command Lines . . . . .	950
	D.2.1 Command Line Arguments . . . . .	951
	D.2.2 Redirection and Piping Arguments . . . . .	952
D.3	Directories, Pathnames, and Filenames. . . . .	953
D.4	Built-in and External Bash Commands. . . . .	954
D.5	Basic Unix Commands . . . . .	955
	D.5.1 man . . . . .	955
	D.5.2 cd or chdir . . . . .	955
	D.5.3 pwd . . . . .	955
	D.5.4 ls . . . . .	956
	D.5.5 file . . . . .	956
	D.5.6 cat, less, more, and tail . . . . .	956
	D.5.7 mv . . . . .	957
	D.5.8 cp . . . . .	958
	D.5.9 rm . . . . .	958
	D.5.10 mkdir . . . . .	959
	D.5.11 date . . . . .	959
	D.5.12 echo . . . . .	959
	D.5.13 chmod . . . . .	959
D.6	Shell Scripts . . . . .	960
	D.6.1 Defining Shell Script Variables and Values . . . . .	961
	D.6.2 Defining Special Shell Variables . . . . .	963
	D.6.3 Writing Your Own Shell Scripts . . . . .	963
D.7	The build Script . . . . .	964
D.8	For More Information . . . . .	968

<b>E</b>	
<b>USEFUL C LANGUAGE FUNCTIONS</b>	<b>971</b>
E.1 String Functions . . . . .	972
E.2 Other C Stdlib and Unix Functions . . . . .	975
<b>F</b>	
<b>ANSWERS TO QUESTIONS</b>	<b>977</b>
<b>INDEX</b>	<b>999</b>