

The Art of Assembly Language

(Full Contents)

Forward Why Would Anyone Learn This Stuff?	1
1 What's Wrong With Assembly Language	1
2 What's Right With Assembly Language?	4
3 Organization of This Text and Pedagogical Concerns	5
4 Obtaining Program Source Listings and Other Materials in This Text	7
Section One:	9
Machine Organization	9
Chapter One Data Representation	11
1.0 Chapter Overview	11
1.1 Numbering Systems	11
1.1.1 A Review of the Decimal System	11
1.1.2 The Binary Numbering System	12
1.1.3 Binary Formats	13
1.2 Data Organization	13
1.2.1 Bits	14
1.2.2 Nibbles	14
1.2.3 Bytes	14
1.2.4 Words	15
1.2.5 Double Words	16
1.3 The Hexadecimal Numbering System	17
1.4 Arithmetic Operations on Binary and Hexadecimal Numbers	19
1.5 Logical Operations on Bits	20
1.6 Logical Operations on Binary Numbers and Bit Strings	22
1.7 Signed and Unsigned Numbers	23
1.8 Sign and Zero Extension	25
1.9 Shifts and Rotates	26
1.10 Bit Fields and Packed Data	28
1.11 The ASCII Character Set	28
1.12 Summary	31
1.13 Laboratory Exercises	33
1.13.1 Installing the Software	33
1.13.2 Data Conversion Exercises	34
1.13.3 Logical Operations Exercises	35
1.13.4 Sign and Zero Extension Exercises	36
1.13.5 Packed Data Exercises	37
1.14 Questions	38
1.15 Programming Projects	41
Chapter Two Boolean Algebra	43
2.0 Chapter Overview	43
2.1 Boolean Algebra	43

2.2 Boolean Functions and Truth Tables	45
2.3 Algebraic Manipulation of Boolean Expressions	48
2.4 Canonical Forms	49
2.5 Simplification of Boolean Functions	52
2.6 What Does This Have To Do With Computers, Anyway?	59
2.6.1 Correspondence Between Electronic Circuits and Boolean Functions	59
2.6.2 Combinatorial Circuits	60
2.6.3 Sequential and Clocked Logic	62
2.7 Okay, What Does It Have To Do With Programming, Then?	64
2.8 Generic Boolean Functions	65
2.9 Laboratory Exercises	69
2.9.1 Truth Tables and Logic Equations Exercises	70
2.9.2 Canonical Logic Equations Exercises	71
2.9.3 Optimization Exercises	72
2.9.4 Logic Evaluation Exercises	72
2.10 Programming Projects	77
2.11 Summary	78
2.12 Questions	80
Chapter Three System Organization	83
3.0 Chapter Overview	83
3.1 The Basic System Components	83
3.1.1 The System Bus	84
3.1.1.1 The Data Bus	84
3.1.1.2 The Address Bus	86
3.1.1.3 The Control Bus	86
3.1.2 The Memory Subsystem	87
3.1.3 The I/O Subsystem	92
3.2 System Timing	92
3.2.1 The System Clock	92
3.2.2 Memory Access and the System Clock	93
3.2.3 Wait States	95
3.2.4 Cache Memory	96
3.3 The 886, 8286, 8486, and 8686 “Hypothetical” Processors	99
3.3.1 CPU Registers	99
3.3.2 The Arithmetic & Logical Unit	100
3.3.3 The Bus Interface Unit	100
3.3.4 The Control Unit and Instruction Sets	100
3.3.5 The x86 Instruction Set	102
3.3.6 Addressing Modes on the x86	103
3.3.7 Encoding x86 Instructions	104
3.3.8 Step-by-Step Instruction Execution	107
3.3.9 The Differences Between the x86 Processors	109
3.3.10 The 886 Processor	110
3.3.11 The 8286 Processor	110
3.3.12 The 8486 Processor	116
3.3.12.1 The 8486 Pipeline	117
3.3.12.2 Stalls in a Pipeline	118
3.3.12.3 Cache, the Prefetch Queue, and the 8486	119

3.3.12.4 Hazards on the 8486	122
3.3.13 The 8686 Processor	123
3.4 I/O (Input/Output)	124
3.5 Interrupts and Polled I/O	126
3.6 Laboratory Exercises	128
3.6.1 The SIMx86 Program – Some Simple x86 Programs	128
3.6.2 Simple I/O-Mapped Input/Output Operations	131
3.6.3 Memory Mapped I/O	132
3.6.4 DMA Exercises	133
3.6.5 Interrupt Driven I/O Exercises	134
3.6.6 Machine Language Programming & Instruction Encoding Exercises	135
3.6.7 Self Modifying Code Exercises	136
3.7 Programming Projects	138
3.8 Summary	139
3.9 Questions	142
Chapter Four Memory Layout and Access	145
4.0 Chapter Overview	145
4.1 The 80x86 CPUs:A Programmer’s View	145
4.1.1 8086 General Purpose Registers	146
4.1.2 8086 Segment Registers	147
4.1.3 8086 Special Purpose Registers	148
4.1.4 80286 Registers	148
4.1.5 80386/80486 Registers	149
4.2 80x86 Physical Memory Organization	150
4.3 Segments on the 80x86	151
4.4 Normalized Addresses on the 80x86	154
4.5 Segment Registers on the 80x86	155
4.6 The 80x86 Addressing Modes	155
4.6.1 8086 Register Addressing Modes	156
4.6.2 8086 Memory Addressing Modes	156
4.6.2.1 The Displacement Only Addressing Mode	156
4.6.2.2 The Register Indirect Addressing Modes	158
4.6.2.3 Indexed Addressing Modes	159
4.6.2.4 Based Indexed Addressing Modes	160
4.6.2.5 Based Indexed Plus Displacement Addressing Mode	160
4.6.2.6 An Easy Way to Remember the 8086 Memory Addressing Modes	162
4.6.2.7 Some Final Comments About 8086 Addressing Modes	162
4.6.3 80386 Register Addressing Modes	163
4.6.4 80386 Memory Addressing Modes	163
4.6.4.1 Register Indirect Addressing Modes	163
4.6.4.2 80386 Indexed, Base/Indexed, and Base/Indexed/Disp Addressing Modes	164
4.6.4.3 80386 Scaled Indexed Addressing Modes	165
4.6.4.4 Some Final Notes About the 80386 Memory Addressing Modes	165
4.7 The 80x86 MOV Instruction	166
4.8 Some Final Comments on the MOV Instructions	169
4.9 Laboratory Exercises	169
4.9.1 The UCR Standard Library for 80x86 Assembly Language Programmers	169
4.9.2 Editing Your Source Files	170

4.9.3 The SHELL.ASM File	170
4.9.4 Assembling Your Code with MASM	172
4.9.5 Debuggers and CodeView™	173
4.9.5.1 A Quick Look at CodeView	173
4.9.5.2 The Source Window	174
4.9.5.3 The Memory Window	175
4.9.5.4 The Register Window	176
4.9.5.5 The Command Window	176
4.9.5.6 The Output Menu Item	177
4.9.5.7 The CodeView Command Window	177
4.9.5.7.1 The Radix Command (N)	177
4.9.5.7.2 The Assemble Command	178
4.9.5.7.3 The Compare Memory Command	178
4.9.5.7.4 The Dump Memory Command	180
4.9.5.7.5 The Enter Command	181
4.9.5.7.6 The Fill Memory Command	182
4.9.5.7.7 The Move Memory Command	182
4.9.5.7.8 The Input Command	183
4.9.5.7.9 The Output Command	183
4.9.5.7.10 The Quit Command	183
4.9.5.7.11 The Register Command	183
4.9.5.7.12 The Unassemble Command	184
4.9.5.8 CodeView Function Keys	184
4.9.5.9 Some Comments on CodeView Addresses	185
4.9.5.10 A Wrap on CodeView	186
4.9.6 Laboratory Tasks	186
4.10 Programming Projects	187
4.11 Summary	188
4.12 Questions	190
Section Two:	193
Basic Assembly Language	193
Chapter Five Variables and Data Structures	195
5.0 Chapter Overview	195
5.1 Some Additional Instructions: LEA, LES, ADD, and MUL	195
5.2 Declaring Variables in an Assembly Language Program	196
5.3 Declaring and Accessing Scalar Variables	197
5.3.1 Declaring and using BYTE Variables	198
5.3.2 Declaring and using WORD Variables	200
5.3.3 Declaring and using DWORD Variables	201
5.3.4 Declaring and using FWORD, QWORD, and TBYTE Variables	202
5.3.5 Declaring Floating Point Variables with REAL4, REAL8, and REAL10	202
5.4 Creating Your Own Type Names with TYPEDEF	203
5.5 Pointer Data Types	203
5.6 Composite Data Types	206
5.6.1 Arrays	206
5.6.1.1 Declaring Arrays in Your Data Segment	207
5.6.1.2 Accessing Elements of a Single Dimension Array	209
5.6.2 Multidimensional Arrays	210
5.6.2.1 Row Major Ordering	211

5.6.2.2	Column Major Ordering	215
5.6.2.3	Allocating Storage for Multidimensional Arrays	216
5.6.2.4	Accessing Multidimensional Array Elements in Assembly Language	217
5.6.3	Structures	218
5.6.4	Arrays of Structures and Arrays/Structures as Structure Fields	220
5.6.5	Pointers to Structures	221
5.7	Sample Programs	222
5.7.1	Simple Variable Declarations	222
5.7.2	Using Pointer Variables	224
5.7.3	Single Dimension Array Access	226
5.7.4	Multidimensional Array Access	227
5.7.5	Simple Structure Access	229
5.7.6	Arrays of Structures	231
5.7.7	Structures and Arrays as Fields of Another Structure	233
5.7.8	Pointers to Structures and Arrays of Structures	235
5.8	Laboratory Exercises	237
5.9	Programming Projects	238
5.10	Summary	239
5.11	Questions	241
Chapter Six	The 80x86 Instruction Set	243
6.0	Chapter Overview	243
6.1	The Processor Status Register (Flags)	244
6.2	Instruction Encodings	245
6.3	Data Movement Instructions	246
6.3.1	The MOV Instruction	246
6.3.2	The XCHG Instruction	247
6.3.3	The LDS, LES, LFS, LGS, and LSS Instructions	248
6.3.4	The LEA Instruction	248
6.3.5	The PUSH and POP Instructions	249
6.3.6	The LAHF and SAHF Instructions	252
6.4	Conversions	252
6.4.1	The MOVZX, MOVSX, CBW, CWD, CWDE, and CDQ Instructions	252
6.4.2	The BSWAP Instruction	254
6.4.3	The XLAT Instruction	255
6.5	Arithmetic Instructions	255
6.5.1	The Addition Instructions: ADD, ADC, INC, XADD, AAA, and DAA	256
6.5.1.1	The ADD and ADC Instructions	256
6.5.1.2	The INC Instruction	258
6.5.1.3	The XADD Instruction	258
6.5.1.4	The AAA and DAA Instructions	258
6.5.2	The Subtraction Instructions: SUB, SBB, DEC, AAS, and DAS	259
6.5.3	The CMP Instruction	261
6.5.4	The CMPXCHG, and CMPXCHG8B Instructions	263
6.5.5	The NEG Instruction	263
6.5.6	The Multiplication Instructions: MUL, IMUL, and AAM	264
6.5.7	The Division Instructions: DIV, IDIV, and AAD	267
6.6	Logical, Shift, Rotate and Bit Instructions	269
6.6.1	The Logical Instructions: AND, OR, XOR, and NOT	269
6.6.2	The Shift Instructions: SHL/SAL, SHR, SAR, SHLD, and SHRD	270

6.6.2.1	SHL/SAL	271
6.6.2.2	SAR	272
6.6.2.3	SHR	273
6.6.2.4	The SHLD and SHRD Instructions	274
6.6.3	The Rotate Instructions: RCL, RCR, ROL, and ROR	276
6.6.3.1	RCL	277
6.6.3.2	RCR	277
6.6.3.3	ROL	278
6.6.3.4	ROR	278
6.6.4	The Bit Operations	279
6.6.4.1	TEST	280
6.6.4.2	The Bit Test Instructions: BT, BTS, BTR, and BTC	280
6.6.4.3	Bit Scanning: BSF and BSR	281
6.6.5	The “Set on Condition” Instructions	281
6.7	I/O Instructions	284
6.8	String Instructions	284
6.9	Program Flow Control Instructions	286
6.9.1	Unconditional Jumps	286
6.9.2	The CALL and RET Instructions	289
6.9.3	The INT, INTO, BOUND, and IRET Instructions	292
6.9.4	The Conditional Jump Instructions	296
6.9.5	The JCXZ/JECXZ Instructions	299
6.9.6	The LOOP Instruction	300
6.9.7	The LOOPE/LOOPZ Instruction	300
6.9.8	The LOOPNE/LOOPNZ Instruction	301
6.10	Miscellaneous Instructions	302
6.11	Sample Programs	303
6.11.1	Simple Arithmetic I	303
6.11.2	Simple Arithmetic II	305
6.11.3	Logical Operations	306
6.11.4	Shift and Rotate Operations	308
6.11.5	Bit Operations and SETcc Instructions	310
6.11.6	String Operations	312
6.11.7	Conditional Jumps	313
6.11.8	CALL and INT Instructions	315
6.11.9	Conditional Jumps I	317
6.11.10	Conditional Jump Instructions II	318
6.12	Laboratory Exercises	320
6.12.1	The IBM/L System	320
6.12.2	IBM/L Exercises	327
6.13	Programming Projects	327
6.14	Summary	328
6.15	Questions	331
Chapter Seven	The UCR Standard Library	333
7.0	Chapter Overview	333
7.1	An Introduction to the UCR Standard Library	333
7.1.1	Memory Management Routines: MEMINIT, MALLOC, and FREE	334
7.1.2	The Standard Input Routines: GETC, GETS, GETSM	334
7.1.3	The Standard Output Routines: PUTC, PUTCR, PUTS, PUTH, PUTI, PRINT, and PRINTF	336

7.1.4 Formatted Output Routines: Putsize, Putsize, Putsize, and Putulsize	340
7.1.5 Output Field Size Routines: Isize, Usize, and Lsize	340
7.1.6 Conversion Routines: ATOx, and xTOA	341
7.1.7 Routines that Test Characters for Set Membership	342
7.1.8 Character Conversion Routines: ToUpper, ToLower	343
7.1.9 Random Number Generation: Random, Randomize	343
7.1.10 Constants, Macros, and other Miscellany	344
7.1.11 Plus more!	344
7.2 Sample Programs	344
7.2.1 Stripped SHELLASM File	345
7.2.2 Numeric I/O	345
7.3 Laboratory Exercises	348
7.3.1 Obtaining the UCR Standard Library	348
7.3.2 Unpacking the Standard Library	349
7.3.3 Using the Standard Library	349
7.3.4 The Standard Library Documentation Files	350
7.4 Programming Projects	351
7.5 Summary	351
7.6 Questions	353
Chapter Eight MASM: Directives & Pseudo-Opcodes	355
8.0 Chapter Overview	355
8.1 Assembly Language Statements	355
8.2 The Location Counter	357
8.3 Symbols	358
8.4 Literal Constants	359
8.4.1 Integer Constants	360
8.4.2 String Constants	361
8.4.3 Real Constants	361
8.4.4 Text Constants	362
8.5 Declaring Manifest Constants Using Equates	362
8.6 Processor Directives	364
8.7 Procedures	365
8.8 Segments	366
8.8.1 Segment Names	367
8.8.2 Segment Loading Order	368
8.8.3 Segment Operands	369
8.8.3.1 The ALIGN Type	369
8.8.3.2 The COMBINE Type	373
8.8.4 The CLASS Type	374
8.8.5 The Read-only Operand	375
8.8.6 The USE16, USE32, and FLAT Options	375
8.8.7 Typical Segment Definitions	376
8.8.8 Why You Would Want to Control the Loading Order	376
8.8.9 Segment Prefixes	377
8.8.10 Controlling Segments with the ASSUME Directive	377
8.8.11 Combining Segments: The GROUP Directive	380
8.8.12 Why Even Bother With Segments?	383
8.9 The END Directive	384

8.10	Variables	384
8.11	Label Types	385
8.11.1	How to Give a Symbol a Particular Type	385
8.11.2	Label Values	386
8.11.3	Type Conflicts	386
8.12	Address Expressions	387
8.12.1	Symbol Types and Addressing Modes	387
8.12.2	Arithmetic and Logical Operators	388
8.12.3	Coercion	390
8.12.4	Type Operators	392
8.12.5	Operator Precedence	396
8.13	Conditional Assembly	397
8.13.1	IF Directive	398
8.13.2	IFE directive	399
8.13.3	IFDEF and IFNDEF	399
8.13.4	IFB, IFNB	399
8.13.5	IFIDN, IFDIF, IFIDNI, and IFDIFI	400
8.14	Macros	400
8.14.1	Procedural Macros	400
8.14.2	Macros vs. 80x86 Procedures	404
8.14.3	The LOCAL Directive	406
8.14.4	The EXITM Directive	406
8.14.5	Macro Parameter Expansion and Macro Operators	407
8.14.6	A Sample Macro to Implement For Loops	409
8.14.7	Macro Functions	413
8.14.8	Predefined Macros, Macro Functions, and Symbols	414
8.14.9	Macros vs. Text Equates	418
8.14.10	Macros: Good and Bad News	419
8.15	Repeat Operations	420
8.16	The FOR and FORC Macro Operations	421
8.17	The WHILE Macro Operation	422
8.18	Macro Parameters	422
8.19	Controlling the Listing	424
8.19.1	The ECHO and %OUT Directives	424
8.19.2	The TITLE Directive	424
8.19.3	The SUBTTL Directive	424
8.19.4	The PAGE Directive	424
8.19.5	The .LIST, .NOLIST, and .XLIST Directives	425
8.19.6	Other Listing Directives	425
8.20	Managing Large Programs	425
8.20.1	The INCLUDE Directive	426
8.20.2	The PUBLIC, EXTERN, and EXTRN Directives	427
8.20.3	The EXTERNDEF Directive	428
8.21	Make Files	429
8.22	Sample Program	432
8.22.1	EX8.MAK	432
8.22.2	Matrix.A	432
8.22.3	EX8.ASM	433
8.22.4	GETL.ASM	442

8.22.5	GetArray.ASM	443
8.22.6	XProduct.ASM	445
8.23	Laboratory Exercises	447
8.23.1	Near vs. Far Procedures	447
8.23.2	Data Alignment Exercises	448
8.23.3	Equate Exercise	449
8.23.4	IFDEF Exercise	450
8.23.5	Make File Exercise	451
8.24	Programming Projects	453
8.25	Summary	453
8.26	Questions	456
Chapter Nine	Arithmetic and Logical Operations	459
9.0	Chapter Overview	459
9.1	Arithmetic Expressions	460
9.1.1	Simple Assignments	460
9.1.2	Simple Expressions	460
9.1.3	Complex Expressions	462
9.1.4	Commutative Operators	466
9.2	Logical (Boolean) Expressions	467
9.3	Multiprecision Operations	470
9.3.1	Multiprecision Addition Operations	470
9.3.2	Multiprecision Subtraction Operations	472
9.3.3	Extended Precision Comparisons	473
9.3.4	Extended Precision Multiplication	475
9.3.5	Extended Precision Division	477
9.3.6	Extended Precision NEG Operations	480
9.3.7	Extended Precision AND Operations	481
9.3.8	Extended Precision OR Operations	482
9.3.9	Extended Precision XOR Operations	482
9.3.10	Extended Precision NOT Operations	482
9.3.11	Extended Precision Shift Operations	482
9.3.12	Extended Precision Rotate Operations	484
9.4	Operating on Different Sized Operands	485
9.5	Machine and Arithmetic Idioms	486
9.5.1	Multiplying Without MUL and IMUL	487
9.5.2	Division Without DIV and IDIV	488
9.5.3	Using AND to Compute Remainders	488
9.5.4	Implementing Modulo-n Counters with AND	489
9.5.5	Testing an Extended Precision Value for 0FFFF.FFh	489
9.5.6	TEST Operations	489
9.5.7	Testing Signs with the XOR Instruction	490
9.6	Masking Operations	490
9.6.1	Masking Operations with the AND Instruction	490
9.6.2	Masking Operations with the OR Instruction	491
9.7	Packing and Unpacking Data Types	491
9.8	Tables	493
9.8.1	Function Computation via Table Look Up	493
9.8.2	Domain Conditioning	496

9.8.3	Generating Tables	497
9.9	Sample Programs	498
9.9.1	Converting Arithmetic Expressions to Assembly Language	498
9.9.2	Boolean Operations Example	500
9.9.3	64-bit Integer I/O	503
9.9.4	Packing and Unpacking Date Data Types	506
9.10	Laboratory Exercises	509
9.10.1	Debugging Programs with CodeView	509
9.10.2	Debugging Strategies	511
9.10.2.1	Locating Infinite Loops	511
9.10.2.2	Incorrect Computations	512
9.10.2.3	Illegal Instructions/Infinite Loops Part II	513
9.10.3	Debug Exercise I: Using CodeView to Find Bugs in a Calculation	513
9.10.4	Software Delay Loop Exercises	515
9.11	Programming Projects	516
9.12	Summary	516
9.13	Questions	518
Chapter 10	Control Structures	521
10.0	Chapter Overview	521
10.1	Introduction to Decisions	521
10.2	IF..THEN..ELSE Sequences	522
10.3	CASE Statements	525
10.4	State Machines and Indirect Jumps	529
10.5	Spaghetti Code	531
10.6	Loops	531
10.6.1	While Loops	532
10.6.2	Repeat..Until Loops	532
10.6.3	LOOP..ENDLOOP Loops	533
10.6.4	FOR Loops	533
10.7	Register Usage and Loops	534
10.8	Performance Improvements	535
10.8.1	Moving the Termination Condition to the End of a Loop	535
10.8.2	Executing the Loop Backwards	537
10.8.3	Loop Invariant Computations	538
10.8.4	Unraveling Loops	539
10.8.5	Induction Variables	540
10.8.6	Other Performance Improvements	541
10.9	Nested Statements	542
10.10	Timing Delay Loops	544
10.11	Sample Program	547
10.12	Laboratory Exercises	552
10.12.1	The Physics of Sound	552
10.12.2	The Fundamentals of Music	553
10.12.3	The Physics of Music	554
10.12.4	The 8253/8254 Timer Chip	555
10.12.5	Programming the Timer Chip to Produce Musical Tones	555
10.12.6	Putting it All Together	556

10.12.7 Amazing Grace Exercise	557
10.13 Programming Projects	558
10.14 Summary	559
10.15 Questions	561
Chapter 11 Procedures and Functions	565
11.0 Chapter Overview	565
11.1 Procedures	566
11.2 Near and Far Procedures	568
11.2.1 Forcing NEAR or FAR CALLs and Returns	568
11.2.2 Nested Procedures	569
11.3 Functions	572
11.4 Saving the State of the Machine	572
11.5 Parameters	574
11.5.1 Pass by Value	574
11.5.2 Pass by Reference	575
11.5.3 Pass by Value-Returned	575
11.5.4 Pass by Result	576
11.5.5 Pass by Name	576
11.5.6 Pass by Lazy-Evaluation	577
11.5.7 Passing Parameters in Registers	578
11.5.8 Passing Parameters in Global Variables	580
11.5.9 Passing Parameters on the Stack	581
11.5.10 Passing Parameters in the Code Stream	590
11.5.11 Passing Parameters via a Parameter Block	598
11.6 Function Results	600
11.6.1 Returning Function Results in a Register	601
11.6.2 Returning Function Results on the Stack	601
11.6.3 Returning Function Results in Memory Locations	602
11.7 Side Effects	602
11.8 Local Variable Storage	604
11.9 Recursion	606
11.10 Sample Program	610
11.11 Laboratory Exercises	618
11.11.1 Ex11_1.cpp	619
11.11.2 Ex11_1.asm	621
11.11.3 EX11_1a.asm	625
11.12 Programming Projects	632
11.13 Summary	633
11.14 Questions	635
Section Three:	637
Intermediate Level Assembly Language Programming	637
Chapter 12 Procedures: Advanced Topics	639
12.0 Chapter Overview	639
12.1 Lexical Nesting, Static Links, and Displays	639
12.1.1 Scope	640

12.1.2	Unit Activation, Address Binding, and Variable Lifetime	642
12.1.3	Static Links	642
12.1.4	Accessing Non-Local Variables Using Static Links	647
12.1.5	The Display	648
12.1.6	The 80286 ENTER and LEAVE Instructions	650
12.2	Passing Variables at Different Lex Levels as Parameters	652
12.2.1	Passing Parameters by Value in a Block Structured Language	652
12.2.2	Passing Parameters by Reference, Result, and Value-Result in a Block Structured Language	653
12.2.3	Passing Parameters by Name and Lazy-Evaluation in a Block Structured Language	654
12.3	Passing Parameters as Parameters to Another Procedure	655
12.3.1	Passing Reference Parameters to Other Procedures	656
12.3.2	Passing Value-Result and Result Parameters as Parameters	657
12.3.3	Passing Name Parameters to Other Procedures	657
12.3.4	Passing Lazy Evaluation Parameters as Parameters	658
12.3.5	Parameter Passing Summary	658
12.4	Passing Procedures as Parameters	659
12.5	Iterators	663
12.5.1	Implementing Iterators Using In-Line Expansion	664
12.5.2	Implementing Iterators with Resume Frames	666
12.6	Sample Programs	669
12.6.1	An Example of an Iterator	669
12.6.2	Another Iterator Example	673
12.7	Laboratory Exercises	678
12.7.1	Iterator Exercise	678
12.7.2	The 80x86 Enter and Leave Instructions	684
12.7.3	Parameter Passing Exercises	690
12.8	Programming Projects	695
12.9	Summary	697
12.10	Questions	698
Chapter 13	MS-DOS, PC-BIOS, and File I/O	699
13.0	Chapter Overview	700
13.1	The IBM PC BIOS	701
13.2	An Introduction to the BIOS' Services	701
13.2.1	INT 5- Print Screen	702
13.2.2	INT 10h - Video Services	702
13.2.3	INT 11h - Equipment Installed	704
13.2.4	INT 12h - Memory Available	704
13.2.5	INT 13h - Low Level Disk Services	704
13.2.6	INT 14h - Serial I/O	706
13.2.6.1	AH=0: Serial Port Initialization	706
13.2.6.2	AH=1: Transmit a Character to the Serial Port	707
13.2.6.3	AH=2: Receive a Character from the Serial Port	707
13.2.6.4	AH=3: Serial Port Status	707
13.2.7	INT 15h - Miscellaneous Services	708
13.2.8	INT 16h - Keyboard Services	708
13.2.8.1	AH=0: Read a Key From the Keyboard	709
13.2.8.2	AH=1: See if a Key is Available at the Keyboard	709
13.2.8.3	AH=2: Return Keyboard Shift Key Status	710
13.2.9	INT 17h - Printer Services	710

13.2.9.1	AH=0: Print a Character	711
13.2.9.2	AH=1: Initialize Printer	711
13.2.9.3	AH=2: Return Printer Status	711
13.2.10	INT 18h - Run BASIC	712
13.2.11	INT 19h - Reboot Computer	712
13.2.12	INT 1Ah - Real Time Clock	712
13.2.12.1	AH=0: Read the Real Time Clock	712
13.2.12.2	AH=1: Setting the Real Time Clock	713
13.3	An Introduction to MS-DOS™	713
13.3.1	MS-DOS Calling Sequence	714
13.3.2	MS-DOS Character Oriented Functions	714
13.3.3	MS-DOS Drive Commands	716
13.3.4	MS-DOS “Obsolete” Filing Calls	717
13.3.5	MS-DOS Date and Time Functions	718
13.3.6	MS-DOS Memory Management Functions	718
13.3.6.1	Allocate Memory	719
13.3.6.2	Deallocate Memory	719
13.3.6.3	Modify Memory Allocation	719
13.3.6.4	Advanced Memory Management Functions	720
13.3.7	MS-DOS Process Control Functions	721
13.3.7.1	Terminate Program Execution	721
13.3.7.2	Terminate, but Stay Resident	721
13.3.7.3	Execute a Program	722
13.3.8	MS-DOS “New” Filing Calls	725
13.3.8.1	Open File	725
13.3.8.2	Create File	726
13.3.8.3	Close File	727
13.3.8.4	Read From a File	727
13.3.8.5	Write to a File	728
13.3.8.6	Seek (Move File Pointer)	728
13.3.8.7	Set Disk Transfer Address (DTA)	729
13.3.8.8	Find First File	729
13.3.8.9	Find Next File	730
13.3.8.10	Delete File	730
13.3.8.11	Rename File	730
13.3.8.12	Change/Get File Attributes	731
13.3.8.13	Get/Set File Date and Time	731
13.3.8.14	Other DOS Calls	732
13.3.9	File I/O Examples	734
13.3.9.1	Example #1: A Hex Dump Utility	734
13.3.9.2	Example #2: Upper Case Conversion	735
13.3.10	Blocked File I/O	737
13.3.11	The Program Segment Prefix (PSP)	739
13.3.12	Accessing Command Line Parameters	742
13.3.13	ARGC and ARGV	750
13.4	UCR Standard Library File I/O Routines	751
13.4.1	Fopen	751
13.4.2	Fcreate	752
13.4.3	Fclose	752
13.4.4	Fflush	752
13.4.5	Fgetc	752
13.4.6	Fread	753
13.4.7	Fputc	753

13.4.8 Fwrite	753
13.4.9 Redirecting I/O Through the StdLib File I/O Routines	753
13.4.10 A File I/O Example	755
13.5 Sample Program	758
13.6 Laboratory Exercises	763
13.7 Programming Projects	768
13.8 Summary	768
13.9 Questions	770
Chapter 14 Floating Point Arithmetic	771
14.0 Chapter Overview	771
14.1 The Mathematics of Floating Point Arithmetic	771
14.2 IEEE Floating Point Formats	774
14.3 The UCR Standard Library Floating Point Routines	777
14.3.1 Load and Store Routines	778
14.3.2 Integer/Floating Point Conversion	779
14.3.3 Floating Point Arithmetic	780
14.3.4 Float/Text Conversion and Printf	780
14.4 The 80x87 Floating Point Coprocessors	781
14.4.1 FPU Registers	781
14.4.1.1 The FPU Data Registers	782
14.4.1.2 The FPU Control Register	782
14.4.1.3 The FPU Status Register	785
14.4.2 FPU Data Types	788
14.4.3 The FPU Instruction Set	789
14.4.4 FPU Data Movement Instructions	789
14.4.4.1 The FLD Instruction	789
14.4.4.2 The FST and FSTP Instructions	790
14.4.4.3 The FXCH Instruction	790
14.4.5 Conversions	791
14.4.5.1 The FILD Instruction	791
14.4.5.2 The FIST and FISTP Instructions	791
14.4.5.3 The FBLD and FBSTP Instructions	792
14.4.6 Arithmetic Instructions	792
14.4.6.1 The FADD and FADDP Instructions	792
14.4.6.2 The FSUB, FSUBP, FSUBR, and FSUBRP Instructions	793
14.4.6.3 The FMUL and FMULP Instructions	794
14.4.6.4 The FDIV, FDIVP, FDIVR, and FDIVRP Instructions	794
14.4.6.5 The FSQRT Instruction	795
14.4.6.6 The FSCALE Instruction	795
14.4.6.7 The FPREM and FPREM1 Instructions	795
14.4.6.8 The FRNDINT Instruction	796
14.4.6.9 The EXTRACT Instruction	796
14.4.6.10 The FABS Instruction	796
14.4.6.11 The FCHS Instruction	797
14.4.7 Comparison Instructions	797
14.4.7.1 The FCOM, FCOMP, and FCOMPP Instructions	797
14.4.7.2 The FUCOM, FUCOMP, and FUCOMPP Instructions	798
14.4.7.3 The FTST Instruction	798
14.4.7.4 The FXAM Instruction	798
14.4.8 Constant Instructions	798

14.4.9	Transcendental Instructions	799
14.4.9.1	The F2XMI Instruction	799
14.4.9.2	The FSIN, FCOS, and FSINCOS Instructions	799
14.4.9.3	The FPTAN Instruction	799
14.4.9.4	The FPATAN Instruction	800
14.4.9.5	The FYL2X and FYL2XP1 Instructions	800
14.4.10	Miscellaneous instructions	800
14.4.10.1	The FINIT and FNINIT Instructions	800
14.4.10.2	The FWAIT Instruction	801
14.4.10.3	The FLDCW and FSTCW Instructions	801
14.4.10.4	The FCLEX and FNCLEX Instructions	801
14.4.10.5	The FLDENV, FSTENV, and FNSTENV Instructions	801
14.4.10.6	The FSAVE, FNSAVE, and FRSTOR Instructions	802
14.4.10.7	The FSTSW and FNSTSW Instructions	803
14.4.10.8	The FINCSTP and FDECSTP Instructions	803
14.4.10.9	The FNOP Instruction	803
14.4.10.10	The FFREE Instruction	803
14.4.11	Integer Operations	803
14.5	Sample Program: Additional Trigonometric Functions	804
14.6	Laboratory Exercises	810
14.6.1	FPU vs StdLib Accuracy	811
14.7	Programming Projects	814
14.8	Summary	814
14.9	Questions	817
Chapter 15	Strings and Character Sets	819
15.0	Chapter Overview	819
15.1	The 80x86 String Instructions	819
15.1.1	How the String Instructions Operate	819
15.1.2	The REP/REPE/REPZ and REPZ/REPNE Prefixes	820
15.1.3	The Direction Flag	821
15.1.4	The MOVS Instruction	822
15.1.5	The CMPS Instruction	826
15.1.6	The SCAS Instruction	828
15.1.7	The STOS Instruction	828
15.1.8	The LODS Instruction	829
15.1.9	Building Complex String Functions from LODS and STOS	830
15.1.10	Prefixes and the String Instructions	830
15.2	Character Strings	831
15.2.1	Types of Strings	831
15.2.2	String Assignment	832
15.2.3	String Comparison	834
15.3	Character String Functions	835
15.3.1	Substr	835
15.3.2	Index	838
15.3.3	Repeat	840
15.3.4	Insert	841
15.3.5	Delete	843
15.3.6	Concatenation	844
15.4	String Functions in the UCR Standard Library	845

15.4.1	StrBDel, StrBDelm	846
15.4.2	Strcat, Strcatl, Strcatm, Strcatml	847
15.4.3	Strchr	848
15.4.4	Strcmp, Strcmpl, Stricmp, Stricmpl	848
15.4.5	Strcpy, Strcpyl, Strdup, Strdupl	849
15.4.6	Strdel, Strdelm	850
15.4.7	Strins, Strinsl, Strinsm, Strinsml	851
15.4.8	Strlen	852
15.4.9	Strlwr, Strlwm, Strupr, Struprm	852
15.4.10	Strev, Strevm	853
15.4.11	Strset, Strsetm	853
15.4.12	Strspan, Strspanl, Strcspan, Strcspanl	854
15.4.13	Strstr, Strstrl	855
15.4.14	Strtrim, Strtrimm	855
15.4.15	Other String Routines in the UCR Standard Library	856
15.5	The Character Set Routines in the UCR Standard Library	856
15.6	Using the String Instructions on Other Data Types	859
15.6.1	Multi-precision Integer Strings	859
15.6.2	Dealing with Whole Arrays and Records	860
15.7	Sample Programs	860
15.7.1	Find.asm	860
15.7.2	StrDemo.asm	862
15.7.3	Fcmp.asm	865
15.8	Laboratory Exercises	868
15.8.1	MOVS Performance Exercise #1	868
15.8.2	MOVS Performance Exercise #2	870
15.8.3	Memory Performance Exercise	872
15.8.4	The Performance of Length-Prefixed vs. Zero-Terminated Strings	874
15.9	Programming Projects	878
15.10	Summary	878
15.11	Questions	881
Chapter 16	Pattern Matching	883
16.1	An Introduction to Formal Language (Automata) Theory	883
16.1.1	Machines vs. Languages	883
16.1.2	Regular Languages	884
16.1.2.1	Regular Expressions	885
16.1.2.2	Nondeterministic Finite State Automata (NFAs)	887
16.1.2.3	Converting Regular Expressions to NFAs	888
16.1.2.4	Converting an NFA to Assembly Language	890
16.1.2.5	Deterministic Finite State Automata (DFAs)	893
16.1.2.6	Converting a DFA to Assembly Language	895
16.1.3	Context Free Languages	900
16.1.4	Eliminating Left Recursion and Left Factoring CFGs	903
16.1.5	Converting REs to CFGs	905
16.1.6	Converting CFGs to Assembly Language	905
16.1.7	Some Final Comments on CFGs	912
16.1.8	Beyond Context Free Languages	912
16.2	The UCR Standard Library Pattern Matching Routines	913
16.3	The Standard Library Pattern Matching Functions	914

16.3.1 Spncset	914
16.3.2 Brkcset	915
16.3.3 Anycset	915
16.3.4 Notanycset	916
16.3.5 MatchStr	916
16.3.6 MatchiStr	916
16.3.7 MatchToStr	917
16.3.8 MatchChar	917
16.3.9 MatchToChar	918
16.3.10 MatchChars	918
16.3.11 MatchToPat	918
16.3.12 EOS	919
16.3.13 ARB	919
16.3.14 ARBNUM	920
16.3.15 Skip	920
16.3.16 Pos	921
16.3.17 RPos	921
16.3.18 GotoPos	921
16.3.19 RGotoPos	922
16.3.20 SL_Match2	922
16.4 Designing Your Own Pattern Matching Routines	922
16.5 Extracting Substrings from Matched Patterns	925
16.6 Semantic Rules and Actions	929
16.7 Constructing Patterns for the MATCH Routine	933
16.8 Some Sample Pattern Matching Applications	935
16.8.1 Converting Written Numbers to Integers	935
16.8.2 Processing Dates	941
16.8.3 Evaluating Arithmetic Expressions	948
16.8.4 A Tiny Assembler	953
16.8.5 The “MADVENTURE” Game	963
16.9 Laboratory Exercises	979
16.9.1 Checking for Stack Overflow (Infinite Loops)	979
16.9.2 Printing Diagnostic Messages from a Pattern	984
16.10 Programming Projects	988
16.11 Summary	988
16.12 Questions	991
Section Four:	993
Advanced Assembly Language Programming	993
Chapter 17 Interrupts, Traps, and Exceptions	995
17.1 80x86 Interrupt Structure and Interrupt Service Routines (ISRs)	996
17.2 Traps	999
17.3 Exceptions	1000
17.3.1 Divide Error Exception (INT 0)	1000
17.3.2 Single Step (Trace) Exception (INT 1)	1000
17.3.3 Breakpoint Exception (INT 3)	1001
17.3.4 Overflow Exception (INT 4/INTO)	1001
17.3.5 Bounds Exception (INT 5/BOUND)	1001
17.3.6 Invalid Opcode Exception (INT 6)	1004

17.3.7 Coprocessor Not Available (INT 7)	1004
17.4 Hardware Interrupts	1004
17.4.1 The 8259A Programmable Interrupt Controller (PIC)	1005
17.4.2 The Timer Interrupt (INT 8)	1007
17.4.3 The Keyboard Interrupt (INT 9)	1008
17.4.4 The Serial Port Interrupts (INT 0Bh and INT 0Ch)	1008
17.4.5 The Parallel Port Interrupts (INT 0Dh and INT 0Fh)	1008
17.4.6 The Diskette and Hard Drive Interrupts (INT 0Eh and INT 76h)	1009
17.4.7 The Real-Time Clock Interrupt (INT 70h)	1009
17.4.8 The FPU Interrupt (INT 75h)	1009
17.4.9 Nonmaskable Interrupts (INT 2)	1009
17.4.10 Other Interrupts	1009
17.5 Chaining Interrupt Service Routines	1010
17.6 Reentrancy Problems	1012
17.7 The Efficiency of an Interrupt Driven System	1014
17.7.1 Interrupt Driven I/O vs. Polling	1014
17.7.2 Interrupt Service Time	1015
17.7.3 Interrupt Latency	1016
17.7.4 Prioritized Interrupts	1020
17.8 Debugging ISRs	1020
17.9 Summary	1021
Chapter 18 Resident Programs	1025
18.1 DOS Memory Usage and TSRs	1025
18.2 Active vs. Passive TSRs	1029
18.3 Reentrancy	1032
18.3.1 Reentrancy Problems with DOS	1032
18.3.2 Reentrancy Problems with BIOS	1033
18.3.3 Reentrancy Problems with Other Code	1034
18.4 The Multiplex Interrupt (INT 2Fh)	1034
18.5 Installing a TSR	1035
18.6 Removing a TSR	1037
18.7 Other DOS Related Issues	1039
18.8 A Keyboard Monitor TSR	1041
18.9 Semiresident Programs	1055
18.10 Summary	1064
Chapter 19 Processes, Coroutines, and Concurrency	1065
19.1 DOS Processes	1065
19.1.1 Child Processes in DOS	1065
19.1.1.1 Load and Execute	1066
19.1.1.2 Load Program	1068
19.1.1.3 Loading Overlays	1069
19.1.1.4 Terminating a Process	1069
19.1.1.5 Obtaining the Child Process Return Code	1070
19.1.2 Exception Handling in DOS: The Break Handler	1070
19.1.3 Exception Handling in DOS: The Critical Error Handler	1071
19.1.4 Exception Handling in DOS: Traps	1075
19.1.5 Redirection of I/O for Child Processes	1075

19.2 Shared Memory	1078
19.2.1 Static Shared Memory	1078
19.2.2 Dynamic Shared Memory	1088
19.3 Coroutines	1103
19.4 Multitasking	1124
19.4.1 Lightweight and HeavyWeight Processes	1124
19.4.2 The UCR Standard Library Processes Package	1125
19.4.3 Problems with Multitasking	1126
19.4.4 A Sample Program with Threads	1127
19.5 Synchronization	1129
19.5.1 Atomic Operations, Test & Set, and Busy-Waiting	1132
19.5.2 Semaphores	1134
19.5.3 The UCR Standard Library Semaphore Support	1136
19.5.4 Using Semaphores to Protect Critical Regions	1136
19.5.5 Using Semaphores for Barrier Synchronization	1140
19.6 Deadlock	1146
19.7 Summary	1147
Section Five:	1151
The PC's I/O Ports	1151
Chapter 20 The PC Keyboard	1153
20.1 Keyboard Basics	1153
20.2 The Keyboard Hardware Interface	1159
20.3 The Keyboard DOS Interface	1167
20.4 The Keyboard BIOS Interface	1168
20.5 The Keyboard Interrupt Service Routine	1174
20.6 Patching into the INT 9 Interrupt Service Routine	1184
20.7 Simulating Keystrokes	1186
20.7.1 Stuffing Characters in the Type Ahead Buffer	1186
20.7.2 Using the 80x86 Trace Flag to Simulate IN AL, 60H Instructions	1187
20.7.3 Using the 8042 Microcontroller to Simulate Keystrokes	1192
20.8 Summary	1196
Chapter 21 The PC Parallel Ports	1199
21.1 Basic Parallel Port Information	1199
21.2 The Parallel Port Hardware	1201
21.3 Controlling a Printer Through the Parallel Port	1202
21.3.1 Printing via DOS	1203
21.3.2 Printing via BIOS	1203
21.3.3 An INT 17h Interrupt Service Routine	1203
21.4 Inter-Computer Communications on the Parallel Port	1209
21.5 Summary	1222
Chapter 22 The PC Serial Ports	1223
22.1 The 8250 Serial Communications Chip	1223
22.1.1 The Data Register (Transmit/Receive Register)	1224
22.1.2 The Interrupt Enable Register (IER)	1224
22.1.3 The Baud Rate Divisor	1225

22.1.4	The Interrupt Identification Register (IIR)	1226
22.1.5	The Line Control Register	1227
22.1.6	The Modem Control Register	1228
22.1.7	The Line Status Register (LSR)	1229
22.1.8	The Modem Status Register (MSR)	1230
22.1.9	The Auxiliary Input Register	1231
22.2	The UCR Standard Library Serial Communications Support Routines	1231
22.3	Programming the 8250 (Examples from the Standard Library)	1233
22.4	Summary	1244
Chapter 23	The PC Video Display	1247
23.1	Memory Mapped Video	1247
23.2	The Video Attribute Byte	1248
23.3	Programming the Text Display	1249
23.4	Summary	1252
Chapter 24	The PC Game Adapter	1255
24.1	Typical Game Devices	1255
24.2	The Game Adapter Hardware	1257
24.3	Using BIOS' Game I/O Functions	1259
24.4	Writing Your Own Game I/O Routines	1260
24.5	The Standard Game Device Interface (SGDI)	1262
24.5.1	Application Programmer's Interface (API)	1262
24.5.2	Read4Sw	1263
24.5.3	Read4Pots:	1263
24.5.4	ReadPot	1264
24.5.5	Read4:	1264
24.5.6	CalibratePot	1264
24.5.7	TestPotCalibration	1264
24.5.8	ReadRaw	1265
24.5.9	ReadSwitch	1265
24.5.10	Read16Sw	1265
24.5.11	Remove	1265
24.5.12	TestPresence	1265
24.5.13	An SGDI Driver for the Standard Game Adapter Card	1265
24.6	An SGDI Driver for the CH Products' Flight Stick Pro™	1280
24.7	Patching Existing Games	1293
24.8	Summary	1306
Section Six:		1309
Optimization		1309
Chapter 25	Optimizing Your Programs	1311
25.0	Chapter Overview	1311
25.1	When to Optimize, When Not to Optimize	1311
25.2	How Do You Find the Slow Code in Your Programs?	1313
25.3	Is Optimization Necessary?	1314
25.4	The Three Types of Optimization	1315
25.5	Improving the Implementation of an Algorithm	1317

25.6 Summary	1341
Section Seven:	1343
Appendixes	1343
Appendix A: ASCII/IBM Character Set	1345
Appendix B: Annotated Bibliography	1347
Appendix C: Keyboard Scan Codes	1351
Appendix D: Instruction Set Reference	1361

