

CONTENTS IN DETAIL

FOREWORD **xxiii**

ACKNOWLEDGMENTS **xxv**

INTRODUCTION **xxvii**

PART I: MACHINE ORGANIZATION **1**

1 HELLO, WORLD OF ASSEMBLY LANGUAGE **3**

1.1 What You'll Need	4
1.2 Setting Up MASM on Your Machine	4
1.3 Setting Up a Text Editor on Your Machine	5
1.4 The Anatomy of a MASM Program	5
1.5 Running Your First MASM Program	6
1.6 Running Your First MASM/C++ Hybrid Program	7
1.7 An Introduction to the Intel x86-64 CPU Family	9
1.8 The Memory Subsystem	13
1.9 Declaring Memory Variables in MASM	14
1.9.1 Associating Memory Addresses with Variables	16
1.9.2 Associating Data Types with Variables	17
1.10 Declaring (Named) Constants in MASM.	18
1.11 Some Basic Machine Instructions	18
1.11.1 The mov Instruction	18
1.11.2 Type Checking on Instruction Operands	20
1.11.3 The add and sub Instructions.	21
1.11.4 The lea Instruction	22
1.11.5 The call and ret Instructions and MASM Procedures.	22
1.12 Calling C/C++ Procedures.	24
1.13 Hello, World!	25
1.14 Returning Function Results in Assembly Language	27
1.15 Automating the Build Process	33
1.16 Microsoft ABI Notes.	35
1.16.1 Variable Size	35
1.16.2 Register Usage	38
1.16.3 Stack Alignment	39
1.17 For More Information.	39
1.18 Test Yourself	40

2

COMPUTER DATA REPRESENTATION AND OPERATIONS

43

2.1	Numbering Systems	44
2.1.1	A Review of the Decimal System	44
2.1.2	The Binary Numbering System	44
2.1.3	Binary Conventions	45
2.2	The Hexadecimal Numbering System	46
2.3	A Note About Numbers vs. Representation	48
2.4	Data Organization	50
2.4.1	Bits	51
2.4.2	Nibbles	51
2.4.3	Bytes	52
2.4.4	Words	53
2.4.5	Double Words	54
2.4.6	Quad Words and Octal Words	55
2.5	Logical Operations on Bits	55
2.5.1	The AND Operation	55
2.5.2	The OR Operation	56
2.5.3	The XOR Operation	57
2.5.4	The NOT Operation	57
2.6	Logical Operations on Binary Numbers and Bit Strings	57
2.7	Signed and Unsigned Numbers	62
2.8	Sign Extension and Zero Extension	67
2.9	Sign Contraction and Saturation	68
2.10	Brief Detour: An Introduction to Control Transfer Instructions	69
2.10.1	The jmp Instruction	69
2.10.2	The Conditional Jump Instructions	70
2.10.3	The cmp Instruction and Corresponding Conditional Jumps	72
2.10.4	Conditional Jump Synonyms	73
2.11	Shifts and Rotates	74
2.12	Bit Fields and Packed Data	79
2.13	IEEE Floating-Point Formats	86
2.13.1	Single-Precision Format	87
2.13.2	Double-Precision Format	88
2.13.3	Extended-Precision Format	89
2.13.4	Normalized Floating-Point Values	89
2.13.5	Non-Numeric Values	90
2.13.6	MASM Support for Floating-Point Values	90
2.14	Binary-Coded Decimal Representation	91
2.15	Characters	92
2.15.1	The ASCII Character Encoding	93
2.15.2	MASM Support for ASCII Characters	95
2.16	The Unicode Character Set	96
2.16.1	Unicode Code Points	96
2.16.2	Unicode Code Planes	97
2.16.3	Unicode Encodings	97
2.17	MASM Support for Unicode	98
2.18	For More Information	99
2.19	Test Yourself	99

3

MEMORY ACCESS AND ORGANIZATION

105

3.1 Runtime Memory Organization	106
3.1.1 The .code Section	108
3.1.2 The .data Section	108
3.1.3 The .const Section	109
3.1.4 The .data? Section	110
3.1.5 Organization of Declaration Sections Within Your Programs	110
3.1.6 Memory Access and 4K Memory Management Unit Pages	111
3.2 How MASM Allocates Memory for Variables	113
3.3 The Label Declaration	114
3.4 Little-Endian and Big-Endian Data Organization	114
3.5 Memory Access.	116
3.6 MASM Support for Data Alignment	119
3.7 The x86-64 Addressing Modes	122
3.7.1 x86-64 Register Addressing Modes	122
3.7.2 x86-64 64-Bit Memory Addressing Modes	123
3.7.3 Large Address Unaware Applications	127
3.8 Address Expressions	130
3.9 The Stack Segment and the push and pop Instructions	134
3.9.1 The Basic push Instruction	134
3.9.2 The Basic pop Instruction	135
3.9.3 Preserving Registers with the push and pop Instructions	137
3.10 The Stack Is a LIFO Data Structure	137
3.11 Other push and pop Instructions	140
3.12 Removing Data from the Stack Without Popping It	140
3.13 Accessing Data You've Pushed onto the Stack Without Popping It	142
3.14 Microsoft ABI Notes	144
3.15 For More Information	144
3.16 Test Yourself	145

4

CONSTANTS, VARIABLES, AND DATA TYPES

147

4.1 The imul Instruction	148
4.2 The inc and dec Instructions	149
4.3 MASM Constant Declarations	149
4.3.1 Constant Expressions	152
4.3.2 this and \$ Operators	154
4.3.3 Constant Expression Evaluation	156
4.4 The MASM typedef Statement	156
4.5 Type Coercion	157
4.6 Pointer Data Types	161
4.6.1 Using Pointers in Assembly Language	162
4.6.2 Declaring Pointers in MASM	163
4.6.3 Pointer Constants and Pointer Constant Expressions	164
4.6.4 Pointer Variables and Dynamic Memory Allocation	166
4.6.5 Common Pointer Problems	167
4.7 Composite Data Types	174
4.8 Character Strings	174
4.8.1 Zero-Terminated Strings	174
4.8.2 Length-Prefixed Strings	175

4.8.3	String Descriptors	176
4.8.4	Pointers to Strings	177
4.8.5	String Functions	177
4.9	Arrays	181
4.9.1	Declaring Arrays in Your MASM Programs	182
4.9.2	Accessing Elements of a Single-Dimensional Array.	183
4.9.3	Sorting an Array of Values	185
4.10	Multidimensional Arrays.	189
4.10.1	Row-Major Ordering	190
4.10.2	Column-Major Ordering.	193
4.10.3	Allocating Storage for Multidimensional Arrays.	194
4.10.4	Accessing Multidimensional Array Elements in Assembly Language	196
4.11	Records/Structs	197
4.11.1	MASM Struct Declarations	198
4.11.2	Accessing Record/Struct Fields	199
4.11.3	Nesting MASM Structs	200
4.11.4	Initializing Struct Fields.	200
4.11.5	Arrays of Structs	203
4.11.6	Aligning Fields Within a Record	204
4.12	Unions	206
4.12.1	Anonymous Unions	208
4.12.2	Variant Types	209
4.13	Microsoft ABI Notes.	210
4.14	For More Information	210
4.15	Test Yourself	210

PART II: ASSEMBLY LANGUAGE PROGRAMMING

213

5 PROCEDURES

215

5.1	Implementing Procedures	216
5.1.1	The call and ret Instructions.	218
5.1.2	Labels in a Procedure.	219
5.2	Saving the State of the Machine	220
5.3	Procedures and the Stack	224
5.3.1	Activation Records.	228
5.3.2	The Assembly Language Standard Entry Sequence	231
5.3.3	The Assembly Language Standard Exit Sequence	233
5.4	Local (Automatic) Variables.	234
5.4.1	Low-Level Implementation of Automatic (Local) Variables.	235
5.4.2	The MASM Local Directive	237
5.4.3	Automatic Allocation	240
5.5	Parameters	240
5.5.1	Pass by Value	241
5.5.2	Pass by Reference	241
5.5.3	Low-Level Parameter Implementation	243
5.5.4	Declaring Parameters with the proc Directive	255
5.5.5	Accessing Reference Parameters on the Stack	256

5.6 Calling Conventions and the Microsoft ABI	261
5.7 The Microsoft ABI and Microsoft Calling Convention	263
5.7.1 Data Types and the Microsoft ABI	263
5.7.2 Parameter Locations.	264
5.7.3 Volatile and Nonvolatile Registers	265
5.7.4 Stack Alignment	267
5.7.5 Parameter Setup and Cleanup (or “What’s with These Magic Instructions?”).	268
5.8 Functions and Function Results.	270
5.9 Recursion	271
5.10 Procedure Pointers	278
5.11 Procedural Parameters	280
5.12 Saving the State of the Machine, Part II	280
5.13 Microsoft ABI Notes.	283
5.14 For More Information	284
5.15 Test Yourself	284

6 ARITHMETIC 287

6.1 x86-64 Integer Arithmetic Instructions.	287
6.1.1 Sign- and Zero-Extension Instructions	288
6.1.2 The mul and imul Instructions.	289
6.1.3 The div and idiv Instructions	291
6.1.4 The cmp Instruction, Revisited	293
6.1.5 The setcc Instructions	295
6.1.6 The test Instruction	297
6.2 Arithmetic Expressions	299
6.2.1 Simple Assignments	299
6.2.2 Simple Expressions	300
6.2.3 Complex Expressions	302
6.2.4 Commutative Operators	307
6.3 Logical (Boolean) Expressions	308
6.4 Machine and Arithmetic Idioms.	310
6.4.1 Multiplying Without mul or imul.	310
6.4.2 Dividing Without div or idiv	312
6.4.3 Implementing Modulo-N Counters with AND.	312
6.5 Floating-Point Arithmetic	313
6.5.1 Floating-Point on the x86-64	317
6.5.2 FPU Registers	317
6.5.3 FPU Data Types.	324
6.5.4 The FPU Instruction Set	325
6.5.5 FPU Data Movement Instructions	326
6.5.6 Conversions	328
6.5.7 Arithmetic Instructions.	330
6.5.8 Comparison Instructions	350
6.5.9 Constant Instructions	360
6.5.10 Transcendental Instructions	361
6.5.11 Miscellaneous Instructions.	363
6.6 Converting Floating-Point Expressions to Assembly Language	364
6.6.1 Converting Arithmetic Expressions to Postfix Notation	366
6.6.2 Converting Postfix Notation to Assembly Language	367

6.7 SSE Floating-Point Arithmetic	369
6.7.1 SSE MXCSR Register	369
6.7.2 SSE Floating-Point Move Instructions	370
6.7.3 SSE Floating-Point Arithmetic Instructions	371
6.7.4 SSE Floating-Point Comparisons	372
6.7.5 SSE Floating-Point Conversions	373
6.8 For More Information	374
6.9 Test Yourself	375

7

LOW-LEVEL CONTROL STRUCTURES 377

7.1 Statement Labels	378
7.1.1 Using Local Symbols in Procedures	378
7.1.2 Initializing Arrays with Label Addresses	381
7.2 Unconditional Transfer of Control (jmp)	382
7.2.1 Register-Indirect Jumps	383
7.2.2 Memory-Indirect Jumps	389
7.3 Conditional Jump Instructions	390
7.4 Trampolines	393
7.5 Conditional Move Instructions	394
7.6 Implementing Common Control Structures in Assembly Language	396
7.6.1 Decisions	396
7.6.2 if/then/else Sequences	397
7.6.3 Complex if Statements Using Complete Boolean Evaluation	400
7.6.4 Short-Circuit Boolean Evaluation	401
7.6.5 Short-Circuit vs. Complete Boolean Evaluation	403
7.6.6 Efficient Implementation of if Statements in Assembly Language	405
7.6.7 switch/case Statements	410
7.7 State Machines and Indirect Jumps	424
7.8 Loops	433
7.8.1 while Loops	433
7.8.2 repeat/until Loops	434
7.8.3 forever/endfor Loops	436
7.8.4 for Loops	437
7.8.5 The break and continue Statements	438
7.8.6 Register Usage and Loops	442
7.9 Loop Performance Improvements	443
7.9.1 Moving the Termination Condition to the End of a Loop	443
7.9.2 Executing the Loop Backward	445
7.9.3 Using Loop-Invariant Computations	446
7.9.4 Unraveling Loops	447
7.9.5 Using Induction Variables	448
7.10 For More Information	450
7.11 Test Yourself	450

8

ADVANCED ARITHMETIC 453

8.1 Extended-Precision Operations	454
8.1.1 Extended-Precision Addition	454
8.1.2 Extended-Precision Subtraction	457
8.1.3 Extended-Precision Comparisons	458

8.1.4	Extended-Precision Multiplication	461
8.1.5	Extended-Precision Division	466
8.1.6	Extended-Precision Negation Operations	477
8.1.7	Extended-Precision AND Operations	479
8.1.8	Extended-Precision OR Operations	479
8.1.9	Extended-Precision XOR Operations	480
8.1.10	Extended-Precision NOT Operations	480
8.1.11	Extended-Precision Shift Operations	480
8.1.12	Extended-Precision Rotate Operations	484
8.2	Operating on Different-Size Operands	485
8.3	Decimal Arithmetic	486
8.3.1	Literal BCD Constants	487
8.3.2	Packed Decimal Arithmetic Using the FPU	488
8.4	For More Information	489
8.5	Test Yourself	489

9

NUMERIC CONVERSION **491**

9.1	Converting Numeric Values to Strings	491
9.1.1	Converting Numeric Values to Hexadecimal Strings	492
9.1.2	Converting Extended-Precision Hexadecimal Values to Strings	499
9.1.3	Converting Unsigned Decimal Values to Strings	500
9.1.4	Converting Signed Integer Values to Strings	507
9.1.5	Converting Extended-Precision Unsigned Integers to Strings	508
9.1.6	Converting Extended-Precision Signed Decimal Values to Strings	513
9.1.7	Formatting Conversions	514
9.1.8	Converting Floating-Point Values to Strings	519
9.2	String-to-Numeric Conversion Routines	546
9.2.1	Converting Decimal Strings to Integers	546
9.2.2	Converting Hexadecimal Strings to Numeric Form	556
9.2.3	Converting Unsigned Decimal Strings to Integers	563
9.2.4	Converting of Extended-Precision String to Unsigned Integer	566
9.2.5	Converting of Extended-Precision Signed Decimal String to Integer	569
9.2.6	Converting of Real String to Floating-Point	570
9.3	For More Information	581
9.4	Test Yourself	581

10

TABLE LOOKUPS **583**

10.1	Tables	583
10.1.1	Function Computation via Table Lookup	584
10.1.2	Generating Tables	590
10.1.3	Table-Lookup Performance	593
10.2	For More Information	593
10.3	Test Yourself	593

11

SIMD INSTRUCTIONS **595**

11.1	The SSE/AVX Architectures	596
11.2	Streaming Data Types	596

11.3	Using cpuid to Differentiate Instruction Sets	599
11.4	Full-Segment Syntax and Segment Alignment	604
11.5	SSE, AVX, and AVX2 Memory Operand Alignment	606
11.6	SIMD Data Movement Instructions	609
11.6.1	The (v)movd and (v)movq Instructions	609
11.6.2	The (v)movaps, (v)movapd, and (v)movdqa Instructions	610
11.6.3	The (v)movups, (v)movupd, and (v)movdqu Instructions	612
11.6.4	Performance of Aligned and Unaligned Moves	612
11.6.5	The (v)movlps and (v)movlpd Instructions	615
11.6.6	The movhps and movhpd Instructions	617
11.6.7	The vmovhps and vmovhpd Instructions	618
11.6.8	The movlhps and vmovlhps Instructions	619
11.6.9	The movhlps and vmovhlps Instructions	619
11.6.10	The (v)movshdup and (v)movsldup Instructions	620
11.6.11	The (v)movddup Instruction	621
11.6.12	The (v)lddqu Instruction	622
11.6.13	Performance Issues and the SIMD Move Instructions	622
11.6.14	Some Final Comments on the SIMD Move Instructions	624
11.7	The Shuffle and Unpack Instructions	625
11.7.1	The (v)pshufb Instructions	625
11.7.2	The (v)pshufd Instructions	626
11.7.3	The (v)pshufhw and (v)pshufhw Instructions	628
11.7.4	The shufps and shufpd Instructions	630
11.7.5	The vshufps and vshufpd Instructions	632
11.7.6	The (v)unpcklps, (v)unpckhps, (v)unpcklpd, and (v)unpckhpd Instructions	633
11.7.7	The Integer Unpack Instructions	637
11.7.8	The (v)pextrb, (v)pextrw, (v)pextrd, and (v)pextrq Instructions	641
11.7.9	The (v)pinsrb, (v)pinsrw, (v)pinsrd, and (v)pinsrq Instructions	642
11.7.10	The (v)extractps and (v)insertps Instructions	643
11.8	SIMD Arithmetic and Logical Operations	644
11.9	The SIMD Logical (Bitwise) Instructions	645
11.9.1	The (v)ptest Instructions	646
11.9.2	The Byte Shift Instructions	646
11.9.3	The Bit Shift Instructions	647
11.10	The SIMD Integer Arithmetic Instructions	648
11.10.1	SIMD Integer Addition	648
11.10.2	Horizontal Additions	650
11.10.3	Double-Word-Sized Horizontal Additions	652
11.10.4	SIMD Integer Subtraction	653
11.10.5	SIMD Integer Multiplication	654
11.10.6	SIMD Integer Averages	657
11.10.7	SIMD Integer Minimum and Maximum	657
11.10.8	SIMD Integer Absolute Value	659
11.10.9	SIMD Integer Sign Adjustment Instructions	659
11.10.10	SIMD Integer Comparison Instructions	660
11.10.11	Integer Conversions	664
11.11	SIMD Floating-Point Arithmetic Operations	668
11.12	SIMD Floating-Point Comparison Instructions	671
11.12.1	SSE and AVX Comparisons	671
11.12.2	Unordered vs. Ordered Comparisons	673
11.12.3	Signaling and Quiet Comparisons	673

11.12.4	Instruction Synonyms	673
11.12.5	AVX Extended Comparisons	674
11.12.6	Using SIMD Comparison Instructions	676
11.12.7	The (v)movmskps, (v)movmskpd Instructions	676
11.13	Floating-Point Conversion Instructions	679
11.14	Aligning SIMD Memory Accesses	681
11.15	Aligning Word, Dword, and Qword Object Addresses	683
11.16	Filling an XMM Register with Several Copies of the Same Value	684
11.17	Loading Some Common Constants Into XMM and YMM Registers	685
11.18	Setting, Clearing, Inverting, and Testing a Single Bit in an SSE Register	687
11.19	Processing Two Vectors by Using a Single Incremented Index	688
11.20	Aligning Two Addresses to a Boundary	690
11.21	Working with Blocks of Data Whose Length Is Not a Multiple of the SSE/AVX Register Size	691
11.22	Dynamically Testing for a CPU Feature	692
11.23	The MASM Include Directive	702
11.24	And a Whole Lot More	703
11.25	For More Information	703
11.26	Test Yourself	705

12

BIT MANIPULATION

707

12.1	What Is Bit Data, Anyway?	707
12.2	Instructions That Manipulate Bits	708
12.2.1	The and Instruction	709
12.2.2	The or Instruction	710
12.2.3	The xor Instruction	712
12.2.4	Flag Modification by Logical Instructions	712
12.2.5	The Bit Test Instructions	715
12.2.6	Manipulating Bits with Shift and Rotate Instructions	716
12.3	The Carry Flag as a Bit Accumulator	716
12.4	Packing and Unpacking Bit Strings	717
12.5	BMI1 Instructions to Extract Bits and Create Bit Masks	723
12.6	Coalescing Bit Sets and Distributing Bit Strings	728
12.7	Coalescing and Distributing Bit Strings Using BMI2 Instructions	731
12.8	Packed Arrays of Bit Strings	733
12.9	Searching for a Bit	736
12.10	Counting Bits	739
12.11	Reversing a Bit String	739
12.12	Merging Bit Strings	741
12.13	Extracting Bit Strings	742
12.14	Searching for a Bit Pattern	743
12.15	For More Information	744
12.16	Test Yourself	744

13

MACROS AND THE MASM COMPILE-TIME LANGUAGE

747

13.1	Introduction to the Compile-Time Language	748
13.2	The echo and .err Directives	748
13.3	Compile-Time Constants and Variables	750
13.4	Compile-Time Expressions and Operators	750

13.4.1	The MASM Escape (!) Operator	750
13.4.2	The MASM Evaluation (%) Operator	750
13.4.3	The catstr Directive	751
13.4.4	The instr Directive	751
13.4.5	The sizestr Directive	752
13.4.6	The substr Directive	752
13.5	Conditional Assembly (Compile-Time Decisions)	752
13.6	Repetitive Assembly (Compile-Time Loops)	756
13.7	Macros (Compile-Time Procedures)	760
13.8	Standard Macros	760
13.9	Macro Parameters	762
13.9.1	Standard Macro Parameter Expansion	762
13.9.2	Optional and Required Macro Parameters	766
13.9.3	Default Macro Parameter Values	768
13.9.4	Macros with a Variable Number of Parameters	769
13.9.5	The Macro Expansion (&) Operator	770
13.10	Local Symbols in a Macro	770
13.11	The exitm Directive	772
13.12	MASM Macro Function Syntax	773
13.13	Macros as Compile-Time Procedures and Functions	775
13.14	Writing Compile-Time “Programs”	776
13.14.1	Constructing Data Tables at Compile Time	776
13.14.2	Unrolling Loops	779
13.15	Simulating HLL Procedure Calls	781
13.15.1	HLL-Like Calls with No Parameters	781
13.15.2	HLL-Like Calls with One Parameter	782
13.15.3	Using opattr to Determine Argument Types	784
13.15.4	HLL-Like Calls with a Fixed Number of Parameters	786
13.15.5	HLL-Like Calls with a Varying Parameter List	791
13.16	The invoke Macro	794
13.17	Advanced Macro Parameter Parsing	795
13.17.1	Checking for String Literal Constants	797
13.17.2	Checking for Real Constants	798
13.17.3	Checking for Registers	808
13.17.4	Compile-Time Arrays	813
13.18	Using Macros to Write Macros	818
13.19	Compile-Time Program Performance	822
13.20	For More Information	822
13.21	Test Yourself	823

14

THE STRING INSTRUCTIONS

825

14.1	The x86-64 String Instructions	826
14.1.1	The rep, repe, repz, and the repnz and repne Prefixes	826
14.1.2	The Direction Flag	827
14.1.3	The movs Instruction	827
14.1.4	The cmps Instruction	832
14.1.5	The scas Instruction	835
14.1.6	The stos Instruction	835
14.1.7	The lods Instruction	836
14.1.8	Building Complex String Functions from lods and stos	837
14.2	Performance of the x86-64 String Instructions	837

14.3 SIMD String Instructions	838
14.3.1 Packed Compare Operand Sizes	839
14.3.2 Type of Comparison.	839
14.3.3 Result Polarity	840
14.3.4 Output Processing	841
14.3.5 Packed String Compare Lengths	841
14.3.6 Packed String Comparison Results	843
14.4 Alignment and Memory Management Unit Pages	844
14.5 For More Information	845
14.6 Test Yourself	845

15
MANAGING COMPLEX PROJECTS **847**

15.1 The include Directive	848
15.2 Ignoring Duplicate Include Operations	849
15.3 Assembly Units and External Directives	849
15.4 Header Files in MASM.	852
15.5 The externdef Directive.	852
15.6 Separate Compilation	854
15.7 An Introduction to Makefiles	862
15.7.1 Basic Makefile Syntax	863
15.7.2 Make Dependencies	864
15.7.3 Make Clean and Touch	867
15.8 The Microsoft Linker and Library Code.	869
15.9 Object File and Library Impact on Program Size	870
15.10 For More Information	871
15.11 Test Yourself	871

16
STAND-ALONE ASSEMBLY LANGUAGE PROGRAMS **873**

16.1 Hello World, by Itself.	874
16.2 Header Files and the Windows Interface	876
16.3 The Win32 API and the Windows ABI	878
16.4 Building a Stand-Alone Console Application	878
16.5 Building a Stand-Alone GUI Application.	879
16.6 A Brief Look at the MessageBox Windows API Function	880
16.7 Windows File I/O	881
16.8 Windows Applications.	897
16.9 For More Information	897
16.10 Test Yourself	898

PART III: REFERENCE MATERIAL **899**

A
ASCII CHARACTER SET **901**

B
GLOSSARY **905**

C	
INSTALLING AND USING VISUAL STUDIO	919
C.1 Installing Visual Studio Community	919
C.2 Creating a Command Line Prompt for MASM	920
C.3 Editing, Assembling, and Running a MASM Source File	922
D	
THE WINDOWS COMMAND LINE INTERPRETER	925
D.1 Command Line Syntax	925
D.2 Directory Names and Drive Letters	928
D.3 Some Useful Built-in Commands	930
D.3.1 The cd and chdir Commands	930
D.3.2 The cls Command	931
D.3.3 The copy Command	931
D.3.4 The date Command	931
D.3.5 The del (erase) Command	932
D.3.6 The dir Command	932
D.3.7 The more Command	932
D.3.8 The move Command	933
D.3.9 The ren and rename Commands	933
D.3.10 The rd and rmdir Commands	933
D.3.11 The time Command	933
D.4 For More Information	934
E	
ANSWERS TO QUESTIONS	935
INDEX	967