

CONTENTS IN DETAIL

ACKNOWLEDGMENTS

xvii

INTRODUCTION

xix

I.1	Content	xix
I.2	Expectations and Prerequisites.	xx
I.3	Source Code in This Book.	xxi
I.4	Stand-Alone Programs and C/C++-Based Programs.	xxiii
I.5	Typography and Pedantry	xxvi

1

ADVANCED MACROS

1

1.1	An Introduction to Domain-Specific Languages	2
1.2	A Review of the MASM Macro Declaration	2
1.2.1	MASM Text Literals	4
1.2.2	The Macro Text Expansion Operator (&).	5
1.2.3	The Text Escape Character (!)	6
1.2.4	The Value Expansion Operator (%)	8
1.2.5	Local Symbols in Macros	8
1.2.6	Macro Functions	11
1.3	Macros as Directives	13
1.4	Text Processing in MASM Macros	16
1.4.1	The <code>sizestr</code> Directive	16
1.4.2	The MASM <code>instr</code> Directive	17
1.4.3	The MASM <code>substr</code> Directive	18
1.4.4	The MASM <code>catstr</code> Directive	19
1.4.5	Functional String Directives	20
1.4.6	An In-Depth Look at Text Equates	22
1.5	Processing Escape Character Sequences in MASM Macros	27
1.6	Simulating Namespaces in MASM	32
1.7	Creating Macro Data Declaration Directives	39
1.8	Context-Free Macros	54
1.8.1	The <code>_push</code> and <code>_pop</code> Macros (Version 1).	59
1.8.2	The <code>_push</code> and <code>_pop</code> Macros (Version 2).	64
1.8.3	The <code>_if</code> and <code>_endif</code> Macros	68
1.8.4	More on Context-Free Macros.	72
1.9	Performance Impact of Macro Programming	73
1.10	Summary	73
1.11	For More Information	74

2

UNICODE STRINGS

75

2.1	The Unicode Character Set.	75
2.2	Unicode Code Points	76
2.3	Unicode Code Planes.	77

2.4	Surrogate Code Points	77
2.5	Glyphs, Characters, and Grapheme Clusters	78
2.6	Unicode Normals and Canonical Equivalence	79
2.7	Unicode Encodings	80
2.7.1	UTF-32: Simplicity at a Cost	80
2.7.2	UTF-16: Windows Internal Format	81
2.7.3	UTF-8: The Ubiquitous Web Standard	81
2.8	Unicode Combining Characters	82
2.9	Unicode and Windows	84
2.10	Printing UTF-8 Characters in a Windows Console Application.	88
2.11	Unicode in Assembly Language.	91
2.11.1	Creating UTF-16 Character Strings in MASM	92
2.11.2	Writing Console Applications That Use UTF-16	111
2.11.3	Creating Unicode Strings in a Resource File	124
2.12	Unicode String Functions	131
2.12.1	Windows ABI	131
2.12.2	The wcsat Function.	132
2.12.3	The wcschr Function.	132
2.12.4	The wcsncmp Function.	133
2.12.5	The wcsncpy Function	133
2.12.6	The wcsnspn Function.	134
2.12.7	The wcslen Function.	134
2.12.8	The wcsspn Function	134
2.12.9	The wcsstr Function	135
2.13	Example Wide String Functions Program	135
2.14	Summary	137
2.15	For More Information	138

3

TRANSCENDENTAL FUNCTIONS

139

3.1	Calling C Standard Library Functions.	140
3.2	FPU Advanced Arithmetic Instruction Review	141
3.2.1	The FPU Status Register	141
3.2.2	The fsqrt Instruction	142
3.2.3	The fscale Instruction	143
3.2.4	The fprem and fprem1 Instructions	144
3.2.5	The fextract Instruction	144
3.2.6	The Constant Instructions	145
3.2.7	The f2xm1 Instruction	145
3.2.8	The fsin, fcos, and fsincos Instructions	146
3.2.9	The fptan Instruction.	146
3.2.10	The fpatan Instruction.	146
3.2.11	The fyl2x and fyl2xp1 Instructions	147
3.3	The Transcendental Function Library.	147
3.3.1	The math.inc Header File	149
3.3.2	The Math Library (Building).	154
3.4	Implementation of the Transcendental Functions.	156
3.4.1	The math_acos Function	157
3.4.2	The math_acot Function	160
3.4.3	The math_acsc Function	163
3.4.4	The math_asec Function	167

3.4.5 The math_asin Function	171
3.4.6 The math_atan Function	174
3.4.7 The math_cos and math_sec Functions	177
3.4.8 The math_exp, math_twoTox, math_tenTox, and math_yTox Functions.	181
3.4.9 The math_ln and math_log Functions	186
3.4.10 The math_sin and math_csc Functions	192
3.4.11 The math_tan and math_cot Functions	196
3.5 Summary	200
3.6 For More Information	201

4 ADVANCED PROCEDURES 203

4.1 Lexical Nesting, Static Links, and Displays	204
4.1.1 Scope	204
4.1.2 Unit Activation, Address Binding, and Variable Lifetime	206
4.1.3 Static Links	209
4.1.4 Nonlocal Variable Access Using Static Links	216
4.1.5 Nonlocal Variable Access Using a Display	218
4.1.6 The enter and leave Instructions.	222
4.1.7 Global, Intermediate, and Local Variables	233
4.1.8 Intermediate Variable Justification	233
4.2 Nesting MASM proc Declarations, Part I	235
4.2.1 Building Activation Records Manually	236
4.3 Nesting MASM proc Declarations, Part II	260
4.4 Signatures and HLL-Like Calls	261
4.4.1 Overloading Procedures Based on Parameter Count	262
4.4.2 Creating Usable Types with MASM	264
4.4.3 Overloading Procedures Based on Parameter Type	276
4.5 Some Additional MASM proc Options	278
4.6 Biasing the Frame Pointer	289
4.7 Hot Patching	292
4.8 Summary	293
4.9 For More Information	294

5 CONCURRENT PROGRAMMING 295

5.1 Multitasking	296
5.2 Multiprocessing	297
5.3 Programs, Processes, Threads, and Fibers.	298
5.4 Windows Programming Using the MASM Library	298
5.5 Creating New Threads Under Windows.	299
5.6 Synchronization	303
5.6.1 Critical Sections	307
5.6.2 Events	314
5.6.3 Mutexes	323
5.6.4 Semaphores	324
5.6.5 Deadlocks	326
5.7 Waitable Timers	339
5.8 Thread-Local Storage	345
5.9 I/O-Bound and Compute-Bound Tasks	356

5.10 Time-Slice Quantum and Task Priorities	357
5.11 Processes	359
5.12 CPU Affinity	366
5.13 Queuing Tasks	370
5.14 Race Conditions	380
5.15 Concurrency Instructions	380
5.15.1 Atomic Instructions	380
5.15.2 Lock Prefix	383
5.15.3 Bit Test and Set Instructions and Spin Loops	385
5.15.4 Compare and Exchange Instruction	387
5.16 Memory Fences and Memory Ordering	393
5.17 Summary	394
5.18 For More Information	395

6

OBJECT-ORIENTED PROGRAMMING WITH MASM

397

6.1 Object-Oriented Programming in Assembly Language	398
6.2 Classes and Objects	398
6.3 Simple Class Declarations in C++	399
6.4 Virtual Method Tables	400
6.5 Classes in Assembly Language	404
6.6 Virtual Methods in Assembly Language	405
6.7 Method Overloading	405
6.8 Sharing VMTs	406
6.9 The this Pointer	407
6.10 Inheritance in Classes	409
6.11 Abstract Methods	413
6.12 Polymorphism in Classes	414
6.13 Multiple Inheritance	415
6.13.1 The Diamond Problem	416
6.13.2 Multiple Inheritance in MASM	416
6.13.3 Multiple Inheritance Method Dispatch	417
6.13.4 Multiple Inheritance and Constructors	418
6.13.5 Problems with Multiple Inheritance	418
6.14 A Set of Macros for Creating MASM Classes	419
6.15 Static (Class) Methods and Virtual Methods	425
6.16 The Class Constructor	426
6.17 The Class Destructor	431
6.18 Super Classes and Calling Base Functions	431
6.19 A Complete Class/Object Example	435
6.20 Protocols and Interfaces	444
6.21 Implementation of the Class Macros	456
6.22 Runtime Type Information	477
6.23 Summary	479
6.24 For More Information	480

7

EXCEPTION HANDLING

481

7.1 Exception Handling in MSVC and Other HLLs	482
7.2 Structured Exception Handling in Windows	487
7.2.1 Using Exception-Handling Strategies	487

7.2.2 Unwinding the Stack	492
7.2.3 Meeting Exception-Handling Prerequisites	499
7.3 MASM Structured Exception-Handling Directives	506
7.3.1 The .allocstack Directive	507
7.3.2 The .endprolog Directive	507
7.3.3 The .pushreg Directive	508
7.3.4 The .savereg Directive	508
7.3.5 The .savexmm128 Directive	509
7.3.6 The .setframe Directive	509
7.3.7 Instructions in a Prolog or Epilog	510
7.4 MASM's HLL-Like proc Features and SEH	512
7.5 Creating a Custom Prolog and Epilog Macro	513
7.6 Procedure Types for Windows SEH	519
7.7 Handling Exceptions	521
7.8 Exception-Handling Macros	541
7.8.1 The try Macro	545
7.8.2 The exception Macro	548
7.8.3 The buildScope "Helper" Macro	553
7.8.4 The finally Macro	554
7.8.5 The endtry Macro	556
7.9 The Exception Procedure Macros	558
7.9.1 The xproc Macro	561
7.9.2 The sehProlog Macro	562
7.9.3 The endxp and sehReturn Macros	566
7.10 A Full Demonstration of the Exception-Handling Macros	569
7.11 Summary	578
7.12 For More Information	579

8 THUNKS AND CLOSURES 581

8.1 First-Class Objects	582
8.2 Thunks	587
8.3 Initializing Thunks	593
8.4 Returning Thunks as Function Results	595
8.5 Activation Record Lifetimes and Thunks	597
8.6 Closures	599
8.6.1 Concurrency and Thread Safety in Closures	600
8.6.2 Closures and Object Duality	601
8.7 Thunks and AI Code	601
8.7.1 The Performance Trade-Off	602
8.7.2 Thunks for Enabling Key AI Features	602
8.7.3 Thunks for Broadcasting and Callbacks	602
8.7.4 Thunks vs. Traditional Function Calls	603
8.8 Jumping Out of a Thunk	603
8.9 Handling Exceptions with Thunks	605
8.10 Summary	607
8.11 For More Information	608

9	ADVANCED PARAMETER IMPLEMENTATION	609
9.1	Parameters	610
9.2	Review of the Windows ABI	610
9.2.1	Data Types and the Windows ABI	610
9.2.2	Parameter Locations in the Windows ABI	611
9.2.3	Volatile and Nonvolatile Registers	612
9.2.4	Stack Alignment	614
9.2.5	Parameter Setup and Cleanup.	615
9.2.6	Final Windows ABI Notes.	617
9.3	Where You Can Pass Parameters.	618
9.3.1	In General-Purpose Registers.	618
9.3.2	In FPU Registers.	620
9.3.3	In Global Variables	622
9.3.4	On the Stack.	623
9.3.5	In the Code Stream	624
9.3.6	In a Parameter Block	626
9.4	How You Can Pass Parameters	627
9.4.1	Pass by Value/Result	628
9.4.2	Caller-Side Value/Result Copying	630
9.4.3	Pass by Result	631
9.4.4	Pass by Name.	632
9.4.5	Pass by Lazy Evaluation	637
9.5	Passing Parameters as Parameters to Another Procedure	638
9.6	Passing Intermediate Variables as Parameters	640
9.7	Passing Procedures as Parameters	641
9.8	Variable (Variadic) Parameter Lists	644
9.9	Function Results	647
9.9.1	Returning Function Results in a Register	647
9.9.2	Returning Function Results on the Stack.	648
9.9.3	Returning Function Results in Memory Locations.	648
9.9.4	Returning Large Function Results	648
9.10	Summary	649
9.11	For More Information	650

10	ITERATORS	651
10.1	Defining an Iterator	652
10.2	Creating Iterators.	654
10.2.1	Implementing Iterators with Inline Expansion	654
10.2.2	Implementing Iterators with Resume Frames.	657
10.3	Macro Implementation of the foreach Loop	665
10.4	Breaking Out of a foreach Loop	669
10.5	Exception Handling and Iterators.	670
10.6	Iterators Within Classes	671
10.7	The Iterator Design Pattern Within Classes	672
10.8	Summary	673
10.9	For More Information	674

11	
COROUTINES, GENERATORS, AND FIBERS	675
11.1 Coroutines	676
11.1.1 Initializing Coroutines	677
11.1.2 Implementing Coroutines	677
11.2 The Windows Fiber API	677
11.2.1 Switching Between Fibers	679
11.2.2 Cleaning Up	679
11.3 Parameters, Register Values, and Shared Data in Coroutine Calls	683
11.3.1 Safer Alternatives for Data Sharing	684
11.3.2 An Example	684
11.4 Recursion and Reentrancy in Coroutines	688
11.4.1 Recursion	688
11.4.2 Reentrancy	688
11.5 Fiber-Local Storage	689
11.5.1 Why Use FLS?	689
11.5.2 FLS API Functions	689
11.5.3 An Example Use Case	690
11.6 Generators	690
11.6.1 Comparison to Iterators	691
11.6.2 Key Differences from Iterators	691
11.6.3 Example Usage Scenario	691
11.6.4 Management of Generator State	692
11.6.5 A Class-Iterator Implementation of a Generator	695
11.6.6 A Coroutine/Fiber Implementation of a Generator	703
11.6.7 More-Complex Generators	727
11.7 Exceptions and Coroutines	729
11.8 Summary	730
11.9 For More Information	730
A	
ASCII CHARACTER SET	731
B	
GLOSSARY	737
C	
INSTALLING AND USING VISUAL STUDIO	743
C.1 Installing Visual Studio Community	744
C.2 Creating a Command-Line Prompt for MASM	744
C.3 Editing a MASM Source File	747
INDEX	751