

# CONTENTS IN DETAIL

## ACKNOWLEDGMENTS

xix

<b>1</b>		
<b>HELLO, WORLD OF ASSEMBLY LANGUAGE</b>		<b>1</b>
1.1	The Anatomy of an HLA Program.....	2
1.2	Running Your First HLA Program.....	4
1.3	Some Basic HLA Data Declarations.....	5
1.4	Boolean Values.....	7
1.5	Character Values.....	8
1.6	An Introduction to the Intel 80x86 CPU Family.....	8
1.7	The Memory Subsystem.....	11
1.8	Some Basic Machine Instructions.....	14
1.9	Some Basic HLA Control Structures.....	17
1.9.1	Boolean Expressions in HLA Statements.....	18
1.9.2	The HLA if..then..elseif..else..endif Statement.....	20
1.9.3	Conjunction, Disjunction, and Negation in Boolean Expressions.....	22
1.9.4	The while..endwhile Statement.....	24
1.9.5	The for..endfor Statement.....	25
1.9.6	The repeat..until Statement.....	26
1.9.7	The break and breakif Statements.....	27
1.9.8	The forever..endfor Statement.....	27
1.9.9	The try..exception..endtry Statement.....	28
1.10	Introduction to the HLA Standard Library.....	32
1.10.1	Predefined Constants in the stdio Module.....	33
1.10.2	Standard In and Standard Out.....	34
1.10.3	The stdout.newLine Routine.....	35
1.10.4	The stdout.putiX Routines.....	35
1.10.5	The stdout.putiXSize Routines.....	35
1.10.6	The stdout.put Routine.....	37
1.10.7	The stdin.getc Routine.....	38
1.10.8	The stdin.getiX Routines.....	39
1.10.9	The stdin.readLine and stdin.flushInput Routines.....	40
1.10.10	The stdin.get Routine.....	41
1.11	Additional Details About try..endtry.....	42
1.11.1	Nesting try..endtry Statements.....	43
1.11.2	The unprotected Clause in a try..endtry Statement.....	45
1.11.3	The anyexception Clause in a try..endtry Statement.....	48
1.11.4	Registers and the try..endtry Statement.....	48
1.12	High-Level Assembly Language vs. Low-Level Assembly Language.....	50
1.13	For More Information.....	51

<b>2</b>		
<b>DATA REPRESENTATION</b>		<b>53</b>
2.1	Numbering Systems .....	54
2.1.1	A Review of the Decimal System .....	54
2.1.2	The Binary Numbering System .....	54
2.1.3	Binary Formats .....	55
2.2	The Hexadecimal Numbering System .....	56
2.3	Data Organization.....	58
2.3.1	Bits .....	58
2.3.2	Nibbles .....	59
2.3.3	Bytes.....	60
2.3.4	Words .....	61
2.3.5	Double Words .....	62
2.3.6	Quad Words and Long Words .....	63
2.4	Arithmetic Operations on Binary and Hexadecimal Numbers .....	64
2.5	A Note About Numbers vs. Representation .....	65
2.6	Logical Operations on Bits .....	67
2.7	Logical Operations on Binary Numbers and Bit Strings.....	70
2.8	Signed and Unsigned Numbers.....	72
2.9	Sign Extension, Zero Extension, Contraction, and Saturation .....	76
2.10	Shifts and Rotates .....	80
2.11	Bit Fields and Packed Data .....	85
2.12	An Introduction to Floating-Point Arithmetic .....	89
2.12.1	IEEE Floating-Point Formats .....	93
2.12.2	HLA Support for Floating-Point Values.....	96
2.13	Binary-Coded Decimal Representation .....	100
2.14	Characters .....	101
2.14.1	The ASCII Character Encoding .....	101
2.14.2	HLA Support for ASCII Characters .....	105
2.15	The Unicode Character Set .....	109
2.16	For More Information .....	110

<b>3</b>		
<b>MEMORY ACCESS AND ORGANIZATION</b>		<b>111</b>
3.1	The 80x86 Addressing Modes .....	112
3.1.1	80x86 Register Addressing Modes .....	112
3.1.2	80x86 32-Bit Memory Addressing Modes .....	113
3.2	Runtime Memory Organization.....	119
3.2.1	The code Section .....	120
3.2.2	The static Section.....	122
3.2.3	The readonly Data Section.....	123
3.2.4	The storage Section .....	123
3.2.5	The @nostorage Attribute .....	124
3.2.6	The var Section .....	125
3.2.7	Organization of Declaration Sections Within Your Programs.....	126
3.3	How HLA Allocates Memory for Variables .....	127
3.4	HLA Support for Data Alignment.....	128

3.5	Address Expressions.....	131
3.6	Type Coercion.....	133
3.7	Register Type Coercion.....	136
3.8	The stack Segment and the push and pop Instructions.....	137
3.8.1	The Basic push Instruction .....	137
3.8.2	The Basic pop Instruction .....	138
3.8.3	Preserving Registers with the push and pop Instructions .....	140
3.9	The Stack Is a LIFO Data Structure.....	140
3.9.1	Other push and pop Instructions .....	143
3.9.2	Removing Data from the Stack Without Popping It .....	144
3.10	Accessing Data You've Pushed onto the Stack Without Popping It.....	146
3.11	Dynamic Memory Allocation and the Heap Segment.....	147
3.12	The inc and dec Instructions .....	152
3.13	Obtaining the Address of a Memory Object.....	152
3.14	For More Information .....	153

## **4**

### **CONSTANTS, VARIABLES, AND DATA TYPES** **155**

4.1	Some Additional Instructions: intmul, bound, into .....	156
4.2	HLA Constant and Value Declarations .....	160
4.2.1	Constant Types.....	164
4.2.2	String and Character Literal Constants.....	165
4.2.3	String and Text Constants in the const Section .....	167
4.2.4	Constant Expressions .....	169
4.2.5	Multiple const Sections and Their Order in an HLA Program .....	171
4.2.6	The HLA val Section .....	172
4.2.7	Modifying val Objects at Arbitrary Points in Your Programs .....	173
4.3	The HLA Type Section.....	173
4.4	enum and HLA Enumerated Data Types .....	174
4.5	Pointer Data Types .....	175
4.5.1	Using Pointers in Assembly Language.....	177
4.5.2	Declaring Pointers in HLA .....	178
4.5.3	Pointer Constants and Pointer Constant Expressions .....	179
4.5.4	Pointer Variables and Dynamic Memory Allocation.....	180
4.5.5	Common Pointer Problems .....	180
4.6	Composite Data Types.....	185
4.7	Character Strings.....	185
4.8	HLA Strings .....	188
4.9	Accessing the Characters Within a String .....	194
4.10	The HLA String Module and Other String-Related Routines.....	196
4.11	In-Memory Conversions .....	208
4.12	Character Sets.....	209
4.13	Character Set Implementation in HLA .....	210
4.14	HLA Character Set Constants and Character Set Expressions.....	212
4.15	Character Set Support in the HLA Standard Library .....	213
4.16	Using Character Sets in Your HLA Programs.....	217
4.17	Arrays .....	218
4.18	Declaring Arrays in Your HLA Programs .....	219

4.19	HLA Array Constants .....	220
4.20	Accessing Elements of a Single-Dimensional Array .....	221
4.21	Sorting an Array of Values .....	222
4.22	Multidimensional Arrays .....	224
	4.22.1 Row-Major Ordering .....	225
	4.22.2 Column-Major Ordering .....	228
4.23	Allocating Storage for Multidimensional Arrays .....	229
4.24	Accessing Multidimensional Array Elements in Assembly Language .....	231
4.25	Records .....	233
4.26	Record Constants .....	235
4.27	Arrays of Records .....	236
4.28	Arrays/Records as Record Fields .....	237
4.29	Aligning Fields Within a Record .....	241
4.30	Pointers to Records .....	242
4.31	Unions .....	243
4.32	Anonymous Unions .....	246
4.33	Variant Types .....	247
4.34	Namespaces .....	248
4.35	Dynamic Arrays in Assembly Language .....	251
4.36	For More Information .....	254

## **5 PROCEDURES AND UNITS 255**

5.1	Procedures .....	255
5.2	Saving the State of the Machine .....	258
5.3	Prematurely Returning from a Procedure .....	262
5.4	Local Variables .....	262
5.5	Other Local and Global Symbol Types .....	268
5.6	Parameters .....	268
	5.6.1 Pass by Value .....	269
	5.6.2 Pass by Reference .....	273
5.7	Functions and Function Results .....	275
	5.7.1 Returning Function Results .....	276
	5.7.2 Instruction Composition in HLA .....	277
	5.7.3 The HLA @returns Option in Procedures .....	280
5.8	Recursion .....	282
5.9	Forward Procedures .....	286
5.10	HLA v2.0 Procedure Declarations .....	287
5.11	Low-Level Procedures and the call Instruction .....	288
5.12	Procedures and the Stack .....	290
5.13	Activation Records .....	293
5.14	The Standard Entry Sequence .....	296
5.15	The Standard Exit Sequence .....	298
5.16	Low-Level Implementation of Automatic (Local) Variables .....	299
5.17	Low-Level Parameter Implementation .....	301
	5.17.1 Passing Parameters in Registers .....	301
	5.17.2 Passing Parameters in the Code Stream .....	304
	5.17.3 Passing Parameters on the Stack .....	307

5.18	Procedure Pointers .....	329
5.19	Procedural Parameters.....	333
5.20	Untyped Reference Parameters .....	334
5.21	Managing Large Programs.....	335
5.22	The #include Directive .....	336
5.23	Ignoring Duplicate #include Operations .....	337
5.24	Units and the external Directive .....	338
	5.24.1 Behavior of the external Directive .....	343
	5.24.2 Header Files in HLA .....	344
5.25	Namespace Pollution .....	345
5.26	For More Information .....	348

## 6

### ARITHMETIC

351

6.1	80x86 Integer Arithmetic Instructions.....	351
	6.1.1 The mul and imul Instructions.....	352
	6.1.2 The div and idiv Instructions .....	355
	6.1.3 The cmp Instruction .....	357
	6.1.4 The setcc Instructions .....	362
	6.1.5 The test Instruction.....	364
6.2	Arithmetic Expressions .....	365
	6.2.1 Simple Assignments .....	366
	6.2.2 Simple Expressions .....	366
	6.2.3 Complex Expressions .....	369
	6.2.4 Commutative Operators .....	374
6.3	Logical (Boolean) Expressions.....	375
6.4	Machine and Arithmetic Idioms .....	377
	6.4.1 Multiplying without mul, imul, or intmul.....	378
	6.4.2 Division Without div or idiv.....	379
	6.4.3 Implementing Modulo-N Counters with and .....	380
6.5	Floating-Point Arithmetic .....	380
	6.5.1 FPU Registers .....	380
	6.5.2 FPU Data Types .....	387
	6.5.3 The FPU Instruction Set .....	389
	6.5.4 FPU Data Movement Instructions .....	389
	6.5.5 Conversions.....	391
	6.5.6 Arithmetic Instructions.....	394
	6.5.7 Comparison Instructions.....	399
	6.5.8 Constant Instructions .....	402
	6.5.9 Transcendental Instructions.....	402
	6.5.10 Miscellaneous Instructions .....	404
	6.5.11 Integer Operations.....	405
6.6	Converting Floating-Point Expressions to Assembly Language .....	406
	6.6.1 Converting Arithmetic Expressions to Postfix Notation .....	407
	6.6.2 Converting Postfix Notation to Assembly Language.....	409
6.7	HLA Standard Library Support for Floating-Point Arithmetic .....	411
6.8	For More Information .....	411

<b>7</b>			
	<b>LOW-LEVEL CONTROL STRUCTURES</b>		<b>413</b>
7.1	Low-Level Control Structures.....		414
7.2	Statement Labels.....		414
7.3	Unconditional Transfer of Control (jmp) .....		416
7.4	The Conditional Jump Instructions.....		418
7.5	“Medium-Level” Control Structures: jt and jf.....		421
7.6	Implementing Common Control Structures in Assembly Language.....		422
7.7	Introduction to Decisions.....		422
	7.7.1 if..then..else Sequences .....		424
	7.7.2 Translating HLA if Statements into Pure Assembly Language .....		427
	7.7.3 Implementing Complex if Statements Using Complete Boolean Evaluation .....		432
	7.7.4 Short-Circuit Boolean Evaluation .....		433
	7.7.5 Short-Circuit vs. Complete Boolean Evaluation.....		435
	7.7.6 Efficient Implementation of if Statements in Assembly Language.....		437
	7.7.7 switch/case Statements .....		442
7.8	State Machines and Indirect Jumps.....		452
7.9	Spaghetti Code .....		455
7.10	Loops .....		456
	7.10.1 while Loops.....		457
	7.10.2 repeat..until Loops .....		458
	7.10.3 forever..endfor Loops .....		459
	7.10.4 for Loops .....		460
	7.10.5 The break and continue Statements .....		461
	7.10.6 Register Usage and Loops.....		465
7.11	Performance Improvements .....		466
	7.11.1 Moving the Termination Condition to the End of a Loop.....		466
	7.11.2 Executing the Loop Backwards .....		469
	7.11.3 Loop-Invariant Computations .....		470
	7.11.4 Unraveling Loops.....		471
	7.11.5 Induction Variables.....		472
7.12	Hybrid Control Structures in HLA.....		473
7.13	For More Information .....		476

<b>8</b>			
	<b>ADVANCED ARITHMETIC</b>		<b>477</b>
8.1	Multiprecision Operations.....		478
	8.1.1 HLA Standard Library Support for Extended-Precision Operations ...		478
	8.1.2 Multiprecision Addition Operations.....		480
	8.1.3 Multiprecision Subtraction Operations.....		483
	8.1.4 Extended-Precision Comparisons .....		485
	8.1.5 Extended-Precision Multiplication .....		488
	8.1.6 Extended-Precision Division.....		492
	8.1.7 Extended-Precision neg Operations.....		501
	8.1.8 Extended-Precision and Operations.....		503
	8.1.9 Extended-Precision or Operations .....		503

8.1.10	Extended-Precision xor Operations.....	504
8.1.11	Extended-Precision not Operations.....	504
8.1.12	Extended-Precision Shift Operations.....	504
8.1.13	Extended-Precision Rotate Operations.....	508
8.1.14	Extended-Precision I/O.....	509
8.2	Operating on Different-Size Operands.....	530
8.3	Decimal Arithmetic.....	532
8.3.1	Literal BCD Constants.....	533
8.3.2	The 80x86 daa and das Instructions.....	534
8.3.3	The 80x86 aaa, aas, aam, and aad Instructions.....	535
8.3.4	Packed Decimal Arithmetic Using the FPU.....	537
8.4	Tables.....	539
8.4.1	Function Computation via Table Lookup.....	539
8.4.2	Domain Conditioning.....	544
8.4.3	Generating Tables.....	545
8.4.4	Table Lookup Performance.....	548
8.5	For More Information.....	549

## **9 MACROS AND THE HLA COMPILE-TIME LANGUAGE 551**

9.1	Introduction to the Compile-Time Language (CTL).....	551
9.2	The #print and #error Statements.....	553
9.3	Compile-Time Constants and Variables.....	555
9.4	Compile-Time Expressions and Operators.....	555
9.5	Compile-Time Functions.....	558
9.5.1	Type-Conversion Compile-Time Functions.....	559
9.5.2	Numeric Compile-Time Functions.....	561
9.5.3	Character-Classification Compile-Time Functions.....	561
9.5.4	Compile-Time String Functions.....	561
9.5.5	Compile-Time Symbol Information.....	562
9.5.6	Miscellaneous Compile-Time Functions.....	563
9.5.7	Compile-Time Type Conversions of Text Objects.....	564
9.6	Conditional Compilation (Compile-Time Decisions).....	565
9.7	Repetitive Compilation (Compile-Time Loops).....	570
9.8	Macros (Compile-Time Procedures).....	573
9.8.1	Standard Macros.....	574
9.8.2	Macro Parameters.....	576
9.8.3	Local Symbols in a Macro.....	582
9.8.4	Macros as Compile-Time Procedures.....	585
9.8.5	Simulating Function Overloading with Macros.....	586
9.9	Writing Compile-Time "Programs".....	592
9.9.1	Constructing Data Tables at Compile Time.....	592
9.9.2	Unrolling Loops.....	596
9.10	Using Macros in Different Source Files.....	598
9.11	For More Information.....	598

<b>10</b>		
<b>BIT MANIPULATION</b>		<b>599</b>
10.1	What Is Bit Data, Anyway?	600
10.2	Instructions That Manipulate Bits	601
10.3	The Carry Flag as a Bit Accumulator	609
10.4	Packing and Unpacking Bit Strings	609
10.5	Coalescing Bit Sets and Distributing Bit Strings	612
10.6	Packed Arrays of Bit Strings	615
10.7	Searching for a Bit	617
10.8	Counting Bits	620
10.9	Reversing a Bit String	623
10.10	Merging Bit Strings	625
10.11	Extracting Bit Strings	626
10.12	Searching for a Bit Pattern	627
10.13	The HLA Standard Library Bits Module	628
10.14	For More Information	631

<b>11</b>		
<b>THE STRING INSTRUCTIONS</b>		<b>633</b>
11.1	The 80x86 String Instructions	634
11.1.1	How the String Instructions Operate	634
11.1.2	The rep/repe/repz and repnz/repne Prefixes	635
11.1.3	The Direction Flag	636
11.1.4	The movs Instruction	638
11.1.5	The cmps Instruction	644
11.1.6	The scas Instruction	647
11.1.7	The stos Instruction	648
11.1.8	The lods Instruction	648
11.1.9	Building Complex String Functions from lods and stos	649
11.2	Performance of the 80x86 String Instructions	650
11.3	For More Information	650



<b>12</b>		
<b>CLASSES AND OBJECTS</b>		<b>651</b>
12.1	General Principles.....	652
12.2	Classes in HLA .....	654
12.3	Objects .....	657
12.4	Inheritance.....	659
12.5	Overriding.....	660
12.6	Virtual Methods vs. Static Procedures .....	661
12.7	Writing Class Methods and Procedures .....	663
12.8	Object Implementation .....	668
	12.8.1 Virtual Method Tables .....	671
	12.8.2 Object Representation with Inheritance.....	673
12.9	Constructors and Object Initialization.....	677
	12.9.1 Dynamic Object Allocation Within the Constructor .....	679
	12.9.2 Constructors and Inheritance .....	681
	12.9.3 Constructor Parameters and Procedure Overloading .....	685
12.10	Destructors.....	686
12.11	HLA's <code>_initialize_</code> and <code>_finalize_</code> Strings.....	687
12.12	Abstract Methods.....	693
12.13	Runtime Type Information.....	696
12.14	Calling Base Class Methods.....	698
12.15	For More Information .....	699
<b>APPENDIX</b>		
<b>ASCII CHARACTER SET</b>		<b>701</b>
<b>INDEX</b>		<b>705</b>