

# The Art of Assembly Language

## (Full Contents)

|  |    |
|--|----|
| 1.1 Foreword to the HLA Version of “The Art of Assembly...”            | 3  |
| 1.2 Intended Audience  | 6  |
| 1.3 Teaching From This Text  | 6  |
| 1.4 Copyright Notice   | 7  |
| 1.5 How to Get a Hard Copy of This Text                                | 8  |
| 1.6 Obtaining Program Source Listings and Other Materials in This Text | 8  |
| 1.7 Where to Get Help  | 8  |
| 1.8 Other Materials You Will Need (Windows Version)                    | 8  |
| 1.9 Other Materials You Will Need (Linux Version)                      | 9  |
| 2.1 Chapter Overview   | 11 |
| 2.2 Installing the HLA Distribution Package                            | 11 |
| 2.2.1 Installation Under Windows                                       | 12 |
| 2.2.2 Installation Under Linux   | 15 |
| 2.2.3 Installing “Art of Assembly” Related Files                       | 18 |
| 2.3 The Anatomy of an HLA Program                                      | 19 |
| 2.4 Some Basic HLA Data Declarations                                   | 21 |
| 2.5 Boolean Values   | 23 |
| 2.6 Character Values   | 23 |
| 2.7 An Introduction to the Intel 80x86 CPU Family                      | 23 |
| 2.8 Some Basic Machine Instructions                                    | 26 |
| 2.9 Some Basic HLA Control Structures                                  | 29 |
| 2.9.1 Boolean Expressions in HLA Statements                            | 30 |
| 2.9.2 The HLA IF..THEN..ELSEIF..ELSE..ENDIF Statement                  | 32 |
| 2.9.3 The WHILE..ENDWHILE Statement                                    | 33 |
| 2.9.4 The FOR..ENDFOR Statement  | 34 |
| 2.9.5 The REPEAT..UNTIL Statement                                      | 35 |
| 2.9.6 The BREAK and BREAKIF Statements                                 | 36 |
| 2.9.7 The FOREVER..ENDFOR Statement                                    | 36 |
| 2.9.8 The TRY..EXCEPTION..ENDTRY Statement                             | 37 |
| 2.10 Introduction to the HLA Standard Library                          | 38 |
| 2.10.1 Predefined Constants in the STDIO Module                        | 40 |
| 2.10.2 Standard In and Standard Out                                    | 40 |
| 2.10.3 The stdout.newLn Routine  | 41 |
| 2.10.4 The stdout.putiX Routines                                       | 41 |
| 2.10.5 The stdout.putiXSize Routines                                   | 41 |
| 2.10.6 The stdout.put Routine  | 42 |
| 2.10.7 The stdin.getc Routine  | 43 |
| 2.10.8 The stdin.getiX Routines  | 44 |
| 2.10.9 The stdin.readLn and stdin.flushInput Routines                  | 46 |

|  |     |
|--|-----|
| 2.10.10 The stdin.get Macro .....                                      | 46  |
| 2.11 Putting It All Together .....                                     | 47  |
| 2.12 Sample Programs .....   | 47  |
| 2.12.1 Powers of Two Table Generation .....                            | 47  |
| 2.12.2 Checkerboard Program .....                                      | 48  |
| 2.12.3 Fibonacci Number Generation .....                               | 50  |
| 3.1 Chapter Overview .....   | 53  |
| 3.2 Numbering Systems .....  | 53  |
| 3.2.1 A Review of the Decimal System .....                             | 53  |
| 3.2.2 The Binary Numbering System .....                                | 54  |
| 3.2.3 Binary Formats .....   | 55  |
| 3.3 Data Organization .....  | 56  |
| 3.3.1 Bits .....   | 56  |
| 3.3.2 Nibbles .....  | 56  |
| 3.3.3 Bytes .....  | 57  |
| 3.3.4 Words .....  | 58  |
| 3.3.5 Double Words .....   | 59  |
| 3.4 The Hexadecimal Numbering System .....                             | 60  |
| 3.5 Arithmetic Operations on Binary and Hexadecimal Numbers .....      | 62  |
| 3.6 A Note About Numbers vs. Representation .....                      | 63  |
| 3.7 Logical Operations on Bits .....                                   | 65  |
| 3.8 Logical Operations on Binary Numbers and Bit Strings .....         | 68  |
| 3.9 Signed and Unsigned Numbers .....                                  | 69  |
| 3.10 Sign Extension, Zero Extension, Contraction, and Saturation ..... | 73  |
| 3.11 Shifts and Rotates .....  | 76  |
| 3.12 Bit Fields and Packed Data .....                                  | 81  |
| 3.13 Putting It All Together .....                                     | 85  |
| 4.1 Chapter Overview .....   | 87  |
| 4.2 An Introduction to Floating Point Arithmetic .....                 | 87  |
| 4.2.1 IEEE Floating Point Formats .....                                | 90  |
| 4.2.2 HLA Support for Floating Point Values .....                      | 93  |
| 4.3 Binary Coded Decimal (BCD) Representation .....                    | 95  |
| 4.4 Characters .....   | 96  |
| 4.4.1 The ASCII Character Encoding .....                               | 97  |
| 4.4.2 HLA Support for ASCII Characters .....                           | 100 |
| 4.4.3 The ASCII Character Set .....                                    | 104 |
| 4.5 The UNICODE Character Set .....                                    | 108 |
| 4.6 Other Data Representations .....                                   | 109 |
| 4.6.1 Representing Colors on a Video Display .....                     | 109 |
| 4.6.2 Representing Audio Information .....                             | 111 |
| 4.6.3 Representing Musical Information .....                           | 114 |
| 4.6.4 Representing Video Information .....                             | 115 |
| 4.6.5 Where to Get More Information About Data Types .....             | 115 |
| 4.7 Putting It All Together .....                                      | 116 |

|   |     |
|---|-----|
| 5.1 Questions .....   | 119 |
| 5.2 Programming Projects for Chapter Two .....                    | 124 |
| 5.3 Programming Projects for Chapter Three .....                  | 124 |
| 5.4 Programming Projects for Chapter Four .....                   | 125 |
| 5.5 Laboratory Exercises for Chapter Two .....                    | 126 |
| 5.5.1 A Short Note on Laboratory Exercises and Lab Reports .....  | 126 |
| 5.5.2 Compiling Your First Program .....                          | 127 |
| 5.5.3 Compiling Other Programs Appearing in this Chapter .....    | 128 |
| 5.5.4 Creating and Modifying HLA Programs .....                   | 129 |
| 5.5.5 Writing a New Program .....                                 | 129 |
| 5.5.6 Correcting Errors in an HLA Program .....                   | 130 |
| 5.5.7 Write Your Own Sample Program .....                         | 131 |
| 5.6 Laboratory Exercises for Chapter Three and Chapter Four ..... | 132 |
| 5.6.1 Data Conversion Exercises .....                             | 132 |
| 5.6.2 Logical Operations Exercises .....                          | 133 |
| 5.6.3 Sign and Zero Extension Exercises .....                     | 133 |
| 5.6.4 Packed Data Exercises .....                                 | 134 |
| 5.6.5 Running this Chapter's Sample Programs .....                | 134 |
| 5.6.6 Write Your Own Sample Program .....                         | 134 |
| 1.1 Chapter Overview .....  | 137 |
| 1.2 The Basic System Components .....                             | 137 |
| 1.2.1 The System Bus .....  | 138 |
| 1.2.1.1 The Data Bus .....  | 138 |
| 1.2.1.2 The Address Bus .....                                     | 139 |
| 1.2.1.3 The Control Bus .....                                     | 139 |
| 1.2.2 The Memory Subsystem .....                                  | 140 |
| 1.2.3 The I/O Subsystem .....                                     | 146 |
| 1.3 HLA Support for Data Alignment .....                          | 146 |
| 1.4 System Timing .....   | 149 |
| 1.4.1 The System Clock .....                                      | 149 |
| 1.4.2 Memory Access and the System Clock .....                    | 150 |
| 1.4.3 Wait States .....   | 151 |
| 1.4.4 Cache Memory .....  | 153 |
| 1.5 Putting It All Together .....                                 | 156 |
| 2.1 Chapter Overview .....  | 157 |
| 2.2 The 80x86 Addressing Modes .....                              | 157 |
| 2.2.1 80x86 Register Addressing Modes .....                       | 157 |
| 2.2.2 80x86 32-bit Memory Addressing Modes .....                  | 158 |
| 2.2.2.1 The Displacement Only Addressing Mode .....               | 158 |
| 2.2.2.2 The Register Indirect Addressing Modes .....              | 159 |
| 2.2.2.3 Indexed Addressing Modes .....                            | 160 |
| 2.2.2.4 Variations on the Indexed Addressing Mode .....           | 161 |
| 2.2.2.5 Scaled Indexed Addressing Modes .....                     | 163 |
| 2.2.2.6 Addressing Mode Wrap-up .....                             | 164 |
| 2.3 Run-Time Memory Organization .....                            | 164 |
| 2.3.1 The Code Section .....                                      | 165 |
| 2.3.2 The Static Sections .....                                   | 167 |

|  |     |
|--|-----|
| 2.3.3 The Read-Only Data Section .....                                       | 167 |
| 2.3.4 The Storage Section .....  | 168 |
| 2.3.5 The @NOSTORAGE Attribute .....   | 169 |
| 2.3.6 The Var Section .....  | 169 |
| 2.3.7 Organization of Declaration Sections Within Your Programs .....        | 170 |
| 2.4 Address Expressions .....  | 171 |
| 2.5 Type Coercion .....  | 173 |
| 2.6 Register Type Coercion .....   | 175 |
| 2.7 The Stack Segment and the Push and Pop Instructions .....                | 176 |
| 2.7.1 The Basic PUSH Instruction .....                                       | 176 |
| 2.7.2 The Basic POP Instruction .....  | 177 |
| 2.7.3 Preserving Registers With the PUSH and POP Instructions .....          | 179 |
| 2.7.4 The Stack is a LIFO Data Structure .....                               | 180 |
| 2.7.5 Other PUSH and POP Instructions .....                                  | 183 |
| 2.7.6 Removing Data From the Stack Without Popping It .....                  | 184 |
| 2.7.7 Accessing Data You've Pushed on the Stack Without Popping It .....     | 186 |
| 2.8 Dynamic Memory Allocation and the Heap Segment .....                     | 187 |
| 2.9 The INC and DEC Instructions .....                                       | 190 |
| 2.10 Obtaining the Address of a Memory Object .....                          | 191 |
| 2.11 Bonus Section: The HLA Standard Library CONSOLE Module .....            | 192 |
| 2.11.1 Clearing the Screen .....   | 192 |
| 2.11.2 Positioning the Cursor .....  | 193 |
| 2.11.3 Locating the Cursor .....   | 194 |
| 2.11.4 Text Attributes .....   | 195 |
| 2.11.5 Filling a Rectangular Section of the Screen .....                     | 197 |
| 2.11.6 Console Direct String Output .....                                    | 199 |
| 2.11.7 Other Console Module Routines .....                                   | 200 |
| 2.12 Putting It All Together .....   | 201 |
| 3.1 Boolean Algebra .....  | 203 |
| 3.2 Boolean Functions and Truth Tables .....                                 | 205 |
| 3.3 Algebraic Manipulation of Boolean Expressions .....                      | 208 |
| 3.4 Canonical Forms .....  | 209 |
| 3.5 Simplification of Boolean Functions .....                                | 214 |
| 3.6 What Does This Have To Do With Computers, Anyway? .....                  | 221 |
| 3.6.1 Correspondence Between Electronic Circuits and Boolean Functions ..... | 221 |
| 3.6.2 Combinatorial Circuits .....   | 223 |
| 3.6.3 Sequential and Clocked Logic .....                                     | 228 |
| 3.7 Okay, What Does It Have To Do With Programming, Then? .....              | 232 |
| 3.8 Putting It All Together .....  | 233 |
| 4.1 Chapter Overview .....   | 234 |
| 4.2 The History of the 80x86 CPU Family .....                                | 234 |
| 4.3 A History of Software Development for the x86 .....                      | 241 |
| 4.4 Basic CPU Design .....   | 245 |
| 4.5 Decoding and Executing Instructions: Random Logic Versus Microcode ..... | 247 |

|  |     |
|--|-----|
| 4.6 RISC vs. CISC vs. VLIW .....   | 248 |
| 4.7 Instruction Execution, Step-By-Step .....  | 250 |
| 4.8 Parallelism – the Key to Faster Processors .....                                     | 253 |
| 4.8.1 The Prefetch Queue – Using Unused Bus Cycles .....                                 | 255 |
| 4.8.2 Pipelining – Overlapping the Execution of Multiple Instructions .....              | 259 |
| 4.8.2.1 A Typical Pipeline .....   | 259 |
| 4.8.2.2 Stalls in a Pipeline .....   | 261 |
| 4.8.3 Instruction Caches – Providing Multiple Paths to Memory .....                      | 262 |
| 4.8.4 Hazards .....  | 263 |
| 4.8.5 Superscalar Operation– Executing Instructions in Parallel .....                    | 265 |
| 4.8.6 Out of Order Execution .....   | 266 |
| 4.8.7 Register Renaming .....  | 266 |
| 4.8.8 Very Long Instruction Word Architecture (VLIW) .....                               | 267 |
| 4.8.9 Parallel Processing .....  | 268 |
| 4.8.10 Multiprocessing .....   | 268 |
| 4.9 Putting It All Together .....  | 269 |
| 5.1 Chapter Overview .....   | 270 |
| 5.2 The Importance of the Design of the Instruction Set .....                            | 270 |
| 5.3 Basic Instruction Design Goals .....   | 271 |
| 5.3.1 Addressing Modes on the Y86 .....  | 278 |
| 5.3.2 Encoding Y86 Instructions .....  | 279 |
| 5.3.3 Hand Encoding Instructions .....   | 282 |
| 5.3.4 Using an Assembler to Encode Instructions .....                                    | 286 |
| 5.3.5 Extending the Y86 Instruction Set .....  | 287 |
| 5.4 Encoding 80x86 Instructions .....  | 288 |
| 5.4.1 Encoding Instruction Operands .....  | 290 |
| 5.4.2 Encoding the ADD Instruction: Some Examples .....                                  | 296 |
| 5.4.3 Encoding Immediate Operands .....  | 300 |
| 5.4.4 Encoding Eight, Sixteen, and Thirty-Two Bit Operands .....                         | 301 |
| 5.4.5 Alternate Encodings for Instructions .....   | 301 |
| 5.5 Putting It All Together .....  | 302 |
| 6.1 Chapter Overview .....   | 303 |
| 6.2 The Memory Hierarchy .....   | 303 |
| 6.3 How the Memory Hierarchy Operates .....  | 305 |
| 6.4 Relative Performance of Memory Subsystems .....                                      | 306 |
| 6.5 Cache Architecture .....   | 308 |
| 6.6 Virtual Memory, Protection, and Paging .....   | 312 |
| 6.7 Thrashing .....  | 314 |
| 6.8 NUMA and Peripheral Devices .....  | 315 |
| 6.9 Segmentation .....   | 316 |
| 6.10 Segments and HLA .....  | 316 |
| 6.10.1 Renaming Segments Under Windows .....   | 317 |
| 6.11 User Defined Segments in HLA (Windows Only) .....                                   | 319 |
| 6.12 Controlling the Placement and Attributes of Segments in Memory (Windows Only) ..... | 321 |
| 6.13 Putting it All Together .....   | 325 |

|   |     |
|---|-----|
| 7.1 Chapter Overview .....  | 327 |
| 7.2 Connecting a CPU to the Outside World .....                           | 327 |
| 7.3 Read-Only, Write-Only, Read/Write, and Dual I/O Ports .....           | 329 |
| 7.4 I/O (Input/Output) Mechanisms .....                                   | 331 |
| 7.4.1 Memory Mapped Input/Output .....                                    | 331 |
| 7.4.2 I/O Mapped Input/Output .....                                       | 332 |
| 7.4.3 Direct Memory Access .....  | 333 |
| 7.5 I/O Speed Hierarchy .....   | 333 |
| 7.6 System Busses and Data Transfer Rates .....                           | 334 |
| 7.7 The AGP Bus .....   | 336 |
| 7.8 Handshaking .....   | 337 |
| 7.9 Time-outs on an I/O Port .....  | 340 |
| 7.10 Interrupts and Polled I/O .....                                      | 342 |
| 7.11 Using a Circular Queue to Buffer Input Data from an ISR .....        | 343 |
| 7.12 Using a Circular Queue to Buffer Output Data for an ISR .....        | 349 |
| 7.13 I/O and the Cache .....  | 352 |
| 7.14 Protected Mode Operation .....                                       | 352 |
| 7.15 Device Drivers .....   | 353 |
| 7.16 Putting It All Together .....  | 354 |
| 8.1 Questions .....   | 355 |
| 8.2 Programming Projects .....  | 361 |
| 8.3 Chapters One and Two Laboratory Exercises .....                       | 363 |
| 8.3.1 Memory Organization Exercises .....                                 | 363 |
| 8.3.2 Data Alignment Exercises .....                                      | 364 |
| 8.3.3 Readonly Segment Exercises .....                                    | 367 |
| 8.3.4 Type Coercion Exercises .....                                       | 367 |
| 8.3.5 Dynamic Memory Allocation Exercises .....                           | 368 |
| 8.4 Chapter Three Laboratory Exercises .....                              | 369 |
| 8.4.1 Truth Tables and Logic Equations Exercises .....                    | 370 |
| 8.4.2 Canonical Logic Equations Exercises .....                           | 371 |
| 8.4.3 Optimization Exercises .....  | 372 |
| 8.4.4 Logic Evaluation Exercises .....                                    | 372 |
| 8.5 Laboratory Exercises for Chapters Four, Five, Six, and Seven .....    | 377 |
| 8.5.1 The SIMY86 Program – Some Simple Y86 Programs .....                 | 377 |
| 8.5.2 Simple I/O-Mapped Input/Output Operations .....                     | 380 |
| 8.5.3 Memory Mapped I/O .....   | 381 |
| 8.5.4 DMA Exercises .....   | 382 |
| 8.5.5 Interrupt Driven I/O Exercises .....                                | 383 |
| 8.5.6 Machine Language Programming & Instruction Encoding Exercises ..... | 384 |
| 8.5.7 Self Modifying Code Exercises .....                                 | 386 |
| 8.5.8 Virtual Memory Exercise .....                                       | 388 |
| 1.1 Chapter Overview .....  | 393 |
| 1.2 Some Additional Instructions: INTMUL, BOUND, INTO .....               | 393 |
| 1.3 The QWORD and TBYTE Data Types .....                                  | 397 |

|       |   |     |
|-------|---|-----|
| 1.4   | HLA Constant and Value Declarations .....                       | 397 |
| 1.4.1 | Constant Types .....  | 400 |
| 1.4.2 | String and Character Literal Constants .....                    | 401 |
| 1.4.3 | String and Text Constants in the CONST Section .....            | 402 |
| 1.4.4 | Constant Expressions .....                                      | 403 |
| 1.4.5 | Multiple CONST Sections and Their Order in an HLA Program ..    | 405 |
| 1.4.6 | The HLA VAL Section .....                                       | 406 |
| 1.4.7 | Modifying VAL Objects at Arbitrary Points in Your Programs .... | 406 |
| 1.5   | The HLA TYPE Section .....                                      | 407 |
| 1.6   | ENUM and HLA Enumerated Data Types .....                        | 408 |
| 1.7   | Pointer Data Types .....  | 409 |
| 1.7.1 | Using Pointers in Assembly Language .....                       | 410 |
| 1.7.2 | Declaring Pointers in HLA .....                                 | 411 |
| 1.7.3 | Pointer Constants and Pointer Constant Expressions .....        | 411 |
| 1.7.4 | Pointer Variables and Dynamic Memory Allocation .....           | 412 |
| 1.7.5 | Common Pointer Problems .....                                   | 413 |
| 1.8   | Putting It All Together .....                                   | 417 |
| 2.1   | Chapter Overview .....  | 419 |
| 2.2   | Composite Data Types .....                                      | 419 |
| 2.3   | Character Strings .....   | 419 |
| 2.4   | HLA Strings .....   | 421 |
| 2.5   | Accessing the Characters Within a String .....                  | 426 |
| 2.6   | The HLA String Module and Other String-Related Routines .....   | 428 |
| 2.7   | In-Memory Conversions .....                                     | 437 |
| 2.8   | Putting It All Together .....                                   | 438 |
| 3.1   | Chapter Overview .....  | 439 |
| 3.2   | The HLA Standard Library CHARS.HHF Module .....                 | 439 |
| 3.3   | Character Sets .....  | 441 |
| 3.4   | Character Set Implementation in HLA .....                       | 442 |
| 3.5   | HLA Character Set Constants and Character Set Expressions ..... | 443 |
| 3.6   | The IN Operator in HLA HLL Boolean Expressions .....            | 444 |
| 3.7   | Character Set Support in the HLA Standard Library .....         | 445 |
| 3.8   | Using Character Sets in Your HLA Programs .....                 | 447 |
| 3.9   | Low-level Implementation of Set Operations .....                | 449 |
| 3.9.1 | Character Set Functions That Build Sets .....                   | 449 |
| 3.9.2 | Traditional Set Operations .....                                | 455 |
| 3.9.3 | Testing Character Sets .....                                    | 458 |
| 3.10  | Putting It All Together .....                                   | 461 |
| 4.1   | Chapter Overview .....  | 463 |
| 4.2   | Arrays .....  | 463 |
| 4.3   | Declaring Arrays in Your HLA Programs .....                     | 464 |
| 4.4   | HLA Array Constants .....                                       | 464 |
| 4.5   | Accessing Elements of a Single Dimension Array .....            | 465 |

|  |     |
|--|-----|
| 4.5.1 Sorting an Array of Values .....                                 | 467 |
| 4.6 Multidimensional Arrays .....                                      | 468 |
| 4.6.1 Row Major Ordering .....   | 469 |
| 4.6.2 Column Major Ordering .....                                      | 473 |
| 4.7 Allocating Storage for Multidimensional Arrays .....               | 474 |
| 4.8 Accessing Multidimensional Array Elements in Assembly Language ... | 475 |
| 4.9 Large Arrays and MASM .....  | 476 |
| 4.10 Dynamic Arrays in Assembly Language .....                         | 477 |
| 4.11 HLA Standard Library Array Support .....                          | 479 |
| 4.12 Putting It All Together .....                                     | 481 |
| 5.1 Chapter Overview .....   | 483 |
| 5.2 Records .....  | 483 |
| 5.3 Record Constants .....   | 485 |
| 5.4 Arrays of Records .....  | 486 |
| 5.5 Arrays/Records as Record Fields .....                              | 487 |
| 5.6 Controlling Field Offsets Within a Record .....                    | 489 |
| 5.7 Aligning Fields Within a Record .....                              | 490 |
| 5.8 Pointers to Records .....  | 491 |
| 5.9 Unions .....   | 492 |
| 5.10 Anonymous Unions .....  | 494 |
| 5.11 Variant Types .....   | 495 |
| 5.12 Namespaces .....  | 496 |
| 5.13 Putting It All Together .....                                     | 498 |
| 6.1 Chapter Overview .....   | 501 |
| 6.2 Dates .....  | 501 |
| 6.3 A Brief History of the Calendar .....                              | 502 |
| 6.4 HLA Date Functions .....   | 505 |
| 6.4.1 date.IsValid and date.validate .....                             | 505 |
| 6.4.2 Checking for Leap Years .....                                    | 507 |
| 6.4.3 Obtaining the System Date .....                                  | 509 |
| 6.4.4 Date to String Conversions and Date Output .....                 | 510 |
| 6.4.5 date.unpack and data.pack .....                                  | 511 |
| 6.4.6 date.Julian, date.fromJulian .....                               | 512 |
| 6.4.7 date.datePlusDays, date.datePlusMonths, and date.daysBetween ..  | 512 |
| 6.4.8 date.dayNumber, date.daysLeft, and date.dayOfWeek .....          | 513 |
| 6.5 Times .....  | 514 |
| 6.5.1 time.curTime .....   | 514 |
| 6.5.2 time.hmsToSecs and time.sectoHMS .....                           | 515 |
| 6.5.3 Time Input/Output .....  | 515 |
| 6.6 Putting It All Together .....                                      | 516 |
| 7.1 Chapter Overview .....   | 517 |
| 7.2 File Organization .....  | 517 |



|   |     |
|---|-----|
| 7.2.1 Files as Lists of Records .....                   | 517 |
| 7.2.2 Binary vs. Text Files .....                       | 518 |
| 7.3 Sequential Files .....                              | 520 |
| 7.4 Random Access Files .....                           | 527 |
| 7.5 ISAM (Indexed Sequential Access Method) Files ..... | 530 |
| 7.6 Truncating a File .....                             | 533 |
| 7.7 File Utility Routines .....                         | 534 |
| 7.7.1 Copying, Moving, and Renaming Files .....         | 534 |
| 7.7.2 Computing the File Size .....                     | 536 |
| 7.7.3 Deleting Files .....                              | 538 |
| 7.8 Directory Operations .....                          | 538 |
| 7.9 Putting It All Together .....                       | 539 |
| 8.1 Chapter Overview .....                              | 541 |
| 8.2 Procedures .....                                    | 541 |
| 8.3 Saving the State of the Machine .....               | 543 |
| 8.4 Prematurely Returning from a Procedure .....        | 546 |
| 8.5 Local Variables .....                               | 547 |
| 8.6 Other Local and Global Symbol Types .....           | 551 |
| 8.7 Parameters .....                                    | 552 |
| 8.7.1 Pass by Value .....                               | 552 |
| 8.7.2 Pass by Reference .....                           | 555 |
| 8.8 Functions and Function Results .....                | 557 |
| 8.8.1 Returning Function Results .....                  | 558 |
| 8.8.2 Instruction Composition in HLA .....              | 558 |
| 8.8.3 The HLA RETURNS Option in Procedures .....        | 560 |
| 8.9 Side Effects .....                                  | 562 |
| 8.10 Recursion .....                                    | 563 |
| 8.11 Forward Procedures .....                           | 567 |
| 8.12 Putting It All Together .....                      | 567 |
| 9.1 Chapter Overview .....                              | 569 |
| 9.2 Managing Large Programs .....                       | 569 |
| 9.3 The #INCLUDE Directive .....                        | 570 |
| 9.4 Ignoring Duplicate Include Operations .....         | 571 |
| 9.5 UNITS and the EXTERNAL Directive .....              | 572 |
| 9.5.1 Behavior of the EXTERNAL Directive .....          | 575 |
| 9.5.2 Header Files in HLA .....                         | 576 |
| 9.6 Make Files .....                                    | 578 |
| 9.7 Code Reuse .....                                    | 580 |
| 9.8 Creating and Managing Libraries .....               | 581 |
| 9.9 Name Space Pollution .....                          | 583 |
| 9.10 Putting It All Together .....                      | 585 |
| 10.1 Chapter Overview .....                             | 587 |

|          |   |     |
|----------|---|-----|
| 10.2     | 80x86 Integer Arithmetic Instructions .....           | 587 |
| 10.2.1   | The MUL and IMUL Instructions .....                   | 587 |
| 10.2.2   | The DIV and IDIV Instructions .....                   | 589 |
| 10.2.3   | The CMP Instruction .....                             | 592 |
| 10.2.4   | The SETcc Instructions .....                          | 593 |
| 10.2.5   | The TEST Instruction .....                            | 596 |
| 10.3     | Arithmetic Expressions .....                          | 597 |
| 10.3.1   | Simple Assignments .....                              | 597 |
| 10.3.2   | Simple Expressions .....                              | 598 |
| 10.3.3   | Complex Expressions .....                             | 600 |
| 10.3.4   | Commutative Operators .....                           | 603 |
| 10.4     | Logical (Boolean) Expressions .....                   | 604 |
| 10.5     | Machine and Arithmetic Idioms .....                   | 606 |
| 10.5.1   | Multiplying without MUL, IMUL, or INTMUL .....        | 606 |
| 10.5.2   | Division Without DIV or IDIV .....                    | 607 |
| 10.5.3   | Implementing Modulo-N Counters with AND .....         | 608 |
| 10.5.4   | Careless Use of Machine Idioms .....                  | 608 |
| 10.6     | The HLA (Pseudo) Random Number Unit .....             | 608 |
| 10.7     | Putting It All Together .....                         | 610 |
| 11.1     | Chapter Overview .....                                | 611 |
| 11.2     | Floating Point Arithmetic .....                       | 611 |
| 11.2.1   | FPU Registers .....                                   | 611 |
| 11.2.1.1 | FPU Data Registers .....                              | 612 |
| 11.2.1.2 | The FPU Control Register .....                        | 612 |
| 11.2.1.3 | The FPU Status Register .....                         | 615 |
| 11.2.2   | FPU Data Types .....                                  | 619 |
| 11.2.3   | The FPU Instruction Set .....                         | 621 |
| 11.2.4   | FPU Data Movement Instructions .....                  | 621 |
| 11.2.4.1 | The FLD Instruction .....                             | 621 |
| 11.2.4.2 | The FST and FSTP Instructions .....                   | 622 |
| 11.2.4.3 | The FXCH Instruction .....                            | 622 |
| 11.2.5   | Conversions .....                                     | 623 |
| 11.2.5.1 | The FILD Instruction .....                            | 623 |
| 11.2.5.2 | The FIST and FISTP Instructions .....                 | 623 |
| 11.2.5.3 | The FBLD and FBSTP Instructions .....                 | 624 |
| 11.2.6   | Arithmetic Instructions .....                         | 624 |
| 11.2.6.1 | The FADD and FADDP Instructions .....                 | 625 |
| 11.2.6.2 | The FSUB, FSUBP, FSUBR, and FSUBRP Instructions ....  | 625 |
| 11.2.6.3 | The FMUL and FMULP Instructions .....                 | 626 |
| 11.2.6.4 | The FDIV, FDIVP, FDIVR, and FDIVRP Instructions ..... | 626 |
| 11.2.6.5 | The FSQRT Instruction .....                           | 627 |
| 11.2.6.6 | The FPREM and FPREM1 Instructions .....               | 628 |
| 11.2.6.7 | The FRNDINT Instruction .....                         | 628 |
| 11.2.6.8 | The FABS Instruction .....                            | 628 |
| 11.2.6.9 | The FCHS Instruction .....                            | 629 |
| 11.2.7   | Comparison Instructions .....                         | 629 |
| 11.2.7.1 | The FCOM, FCOMP, and FCOMPP Instructions .....        | 629 |
| 11.2.7.2 | The FTST Instruction .....                            | 630 |
| 11.2.8   | Constant Instructions .....                           | 631 |

|           |   |     |
|-----------|---|-----|
| 11.2.9    | Transcendental Instructions                                   | 631 |
| 11.2.9.1  | The F2XM1 Instruction   | 631 |
| 11.2.9.2  | The FSIN, FCOS, and FSINCOS Instructions                      | 631 |
| 11.2.9.3  | The FPTAN Instruction   | 632 |
| 11.2.9.4  | The FPATAN Instruction  | 632 |
| 11.2.9.5  | The FYL2X Instruction   | 632 |
| 11.2.9.6  | The FYL2XP1 Instruction                                       | 632 |
| 11.2.10   | Miscellaneous instructions                                    | 633 |
| 11.2.10.1 | The FINIT and FNINIT Instructions                             | 633 |
| 11.2.10.2 | The FLDCW and FSTCW Instructions                              | 633 |
| 11.2.10.3 | The FCLEX and FNCLEX Instructions                             | 633 |
| 11.2.10.4 | The FSTSW and FNSTSW Instructions                             | 633 |
| 11.2.11   | Integer Operations  | 634 |
| 11.3      | Converting Floating Point Expressions to Assembly Language    | 634 |
| 11.3.1    | Converting Arithmetic Expressions to Postfix Notation         | 635 |
| 11.3.2    | Converting Postfix Notation to Assembly Language              | 637 |
| 11.3.3    | Mixed Integer and Floating Point Arithmetic                   | 638 |
| 11.4      | HLA Standard Library Support for Floating Point Arithmetic    | 638 |
| 11.4.1    | The stdin.getf and fileio.getf Functions                      | 639 |
| 11.4.2    | Trigonometric Functions in the HLA Math Library               | 639 |
| 11.4.3    | Exponential and Logarithmic Functions in the HLA Math Library | 640 |
| 11.5      | Sample Program  | 640 |
| 11.6      | Putting It All Together                                       | 646 |
| 12.1      | Chapter Overview  | 647 |
| 12.2      | Tables  | 647 |
| 12.2.1    | Function Computation via Table Look-up                        | 647 |
| 12.2.2    | Domain Conditioning   | 650 |
| 12.2.3    | Generating Tables   | 651 |
| 12.3      | High Performance Implementation of cs.rangeChar               | 655 |
| 13.1      | Questions   | 663 |
| 13.2      | Programming Projects  | 670 |
| 13.3      | Laboratory Exercises  | 677 |
| 13.3.1    | Using the BOUND Instruction to Check Array Indices            | 677 |
| 13.3.2    | Using TEXT Constants in Your Programs                         | 680 |
| 13.3.3    | Constant Expressions Lab Exercise                             | 682 |
| 13.3.4    | Pointers and Pointer Constants Exercises                      | 684 |
| 13.3.5    | String Exercises  | 685 |
| 13.3.6    | String and Character Set Exercises                            | 687 |
| 13.3.7    | Console Array Exercise  | 691 |
| 13.3.8    | Multidimensional Array Exercises                              | 693 |
| 13.3.9    | Console Attributes Laboratory Exercise                        | 696 |
| 13.3.10   | Records, Arrays, and Pointers Laboratory Exercise             | 698 |
| 13.3.11   | Separate Compilation Exercises                                | 704 |
| 13.3.12   | The HLA (Pseudo) Random Number Unit                           | 710 |
| 13.3.13   | File I/O in HLA   | 711 |
| 13.3.14   | Timing Various Arithmetic Instructions                        | 712 |
| 13.3.15   | Using the RDTSC Instruction to Time a Code Sequence           | 715 |
| 13.3.16   | Timing Floating Point Instructions                            | 719 |

|  |     |
|--|-----|
| 13.3.17 Table Lookup Exercise .....  | 722 |
| 1.1 Chapter Overview .....   | 727 |
| 1.2 Conjunction, Disjunction, and Negation in Boolean Expressions .....          | 727 |
| 1.3 TRY..ENDTRY .....  | 729 |
| 1.3.1 Nesting TRY..ENDTRY Statements .....                                       | 730 |
| 1.3.2 The UNPROTECTED Clause in a TRY..ENDTRY Statement .....                    | 732 |
| 1.3.3 The ANYEXCEPTION Clause in a TRY..ENDTRY Statement .....                   | 735 |
| 1.3.4 Raising User-Defined Exceptions .....                                      | 735 |
| 1.3.5 Reraising Exceptions in a TRY..ENDTRY Statement .....                      | 737 |
| 1.3.6 A List of the Predefined HLA Exceptions .....                              | 737 |
| 1.3.7 How to Handle Exceptions in Your Programs .....                            | 737 |
| 1.3.8 Registers and the TRY..ENDTRY Statement .....                              | 739 |
| 1.4 BEGIN..EXIT..EXITIF..END .....   | 740 |
| 1.5 CONTINUE..CONTINUEIF .....   | 745 |
| 1.6 SWITCH..CASE..DEFAULT..ENDSWITCH .....                                       | 747 |
| 1.7 Putting It All Together .....  | 749 |
| 2.1 Chapter Overview .....   | 751 |
| 2.2 Low Level Control Structures .....   | 751 |
| 2.3 Statement Labels .....   | 751 |
| 2.4 Unconditional Transfer of Control (JMP) .....                                | 753 |
| 2.5 The Conditional Jump Instructions .....                                      | 755 |
| 2.6 “Medium-Level” Control Structures: JT and JF .....                           | 759 |
| 2.7 Implementing Common Control Structures in Assembly Language .....            | 759 |
| 2.8 Introduction to Decisions .....  | 760 |
| 2.8.1 IF..THEN..ELSE Sequences .....   | 761 |
| 2.8.2 Translating HLA IF Statements into Pure Assembly Language .....            | 764 |
| 2.8.3 Implementing Complex IF Statements Using Complete Boolean Evaluation ..... | 768 |
| 2.8.4 Short Circuit Boolean Evaluation .....                                     | 769 |
| 2.8.5 Short Circuit vs. Complete Boolean Evaluation .....                        | 770 |
| 2.8.6 Efficient Implementation of IF Statements in Assembly Language .....       | 772 |
| 2.8.7 SWITCH/CASE Statements .....   | 776 |
| 2.9 State Machines and Indirect Jumps .....                                      | 784 |
| 2.10 Spaghetti Code .....  | 786 |
| 2.11 Loops .....   | 787 |
| 2.11.1 While Loops .....   | 787 |
| 2.11.2 Repeat..Until Loops .....   | 788 |
| 2.11.3 FOREVER..ENDFOR Loops .....   | 789 |
| 2.11.4 FOR Loops .....   | 790 |
| 2.11.5 The BREAK and CONTINUE Statements .....                                   | 791 |
| 2.11.6 Register Usage and Loops .....  | 795 |
| 2.12 Performance Improvements .....  | 796 |
| 2.12.1 Moving the Termination Condition to the End of a Loop .....               | 796 |
| 2.12.2 Executing the Loop Backwards .....  | 798 |
| 2.12.3 Loop Invariant Computations .....   | 799 |
| 2.12.4 Unraveling Loops .....  | 800 |

|   |     |
|---|-----|
| 2.12.5 Induction Variables .....                                  | 801 |
| 2.13 Hybrid Control Structures in HLA .....                       | 802 |
| 2.14 Putting It All Together .....                                | 804 |
| 3.1 Chapter Overview .....  | 805 |
| 3.2 Procedures and the CALL Instruction .....                     | 805 |
| 3.3 Procedures and the Stack .....                                | 807 |
| 3.4 Activation Records .....                                      | 810 |
| 3.5 The Standard Entry Sequence .....                             | 813 |
| 3.6 The Standard Exit Sequence .....                              | 814 |
| 3.7 HLA Local Variables .....                                     | 815 |
| 3.8 Parameters .....  | 816 |
| 3.8.1 Pass by Value .....   | 817 |
| 3.8.2 Pass by Reference .....                                     | 817 |
| 3.8.3 Passing Parameters in Registers .....                       | 818 |
| 3.8.4 Passing Parameters in the Code Stream .....                 | 820 |
| 3.8.5 Passing Parameters on the Stack .....                       | 822 |
| 3.8.5.1 Accessing Value Parameters on the Stack .....             | 824 |
| 3.8.5.2 Passing Value Parameters on the Stack .....               | 825 |
| 3.8.5.3 Accessing Reference Parameters on the Stack .....         | 831 |
| 3.8.5.4 Passing Reference Parameters on the Stack .....           | 834 |
| 3.8.5.5 Passing Formal Parameters as Actual Parameters .....      | 836 |
| 3.8.5.6 HLA Hybrid Parameter Passing Facilities .....             | 838 |
| 3.8.5.7 Mixing Register and Stack Based Parameters .....          | 839 |
| 3.9 Procedure Pointers .....                                      | 839 |
| 3.10 Procedural Parameters .....                                  | 842 |
| 3.11 Untyped Reference Parameters .....                           | 843 |
| 3.12 Iterators and the FOREACH Loop .....                         | 843 |
| 3.13 Sample Programs .....  | 846 |
| 3.13.1 Generating the Fibonacci Sequence Using an Iterator .....  | 846 |
| 3.13.2 Outer Product Computation with Procedural Parameters ..... | 848 |
| 3.14 Putting It All Together .....                                | 851 |
| 4.1 Chapter Overview .....  | 853 |
| 4.2 Multiprecision Operations .....                               | 853 |
| 4.2.1 Multiprecision Addition Operations .....                    | 853 |
| 4.2.2 Multiprecision Subtraction Operations .....                 | 856 |
| 4.2.3 Extended Precision Comparisons .....                        | 857 |
| 4.2.4 Extended Precision Multiplication .....                     | 860 |
| 4.2.5 Extended Precision Division .....                           | 864 |
| 4.2.6 Extended Precision NEG Operations .....                     | 872 |
| 4.2.7 Extended Precision AND Operations .....                     | 873 |
| 4.2.8 Extended Precision OR Operations .....                      | 874 |
| 4.2.9 Extended Precision XOR Operations .....                     | 874 |
| 4.2.10 Extended Precision NOT Operations .....                    | 874 |
| 4.2.11 Extended Precision Shift Operations .....                  | 875 |
| 4.2.12 Extended Precision Rotate Operations .....                 | 878 |
| 4.2.13 Extended Precision I/O .....                               | 878 |

|          |  |     |
|----------|--|-----|
| 4.2.13.1 | Extended Precision Hexadecimal Output .....                | 879 |
| 4.2.13.2 | Extended Precision Unsigned Decimal Output .....           | 879 |
| 4.2.13.3 | Extended Precision Signed Decimal Output .....             | 882 |
| 4.2.13.4 | Extended Precision Formatted I/O .....                     | 883 |
| 4.2.13.5 | Extended Precision Input Routines .....                    | 884 |
| 4.2.13.6 | Extended Precision Hexadecimal Input .....                 | 887 |
| 4.2.13.7 | Extended Precision Unsigned Decimal Input .....            | 891 |
| 4.2.13.8 | Extended Precision Signed Decimal Input .....              | 895 |
| 4.3      | Operating on Different Sized Operands .....                | 895 |
| 4.4      | Decimal Arithmetic .....                                   | 897 |
| 4.4.1    | Literal BCD Constants .....                                | 898 |
| 4.4.2    | The 80x86 DAA and DAS Instructions .....                   | 898 |
| 4.4.3    | The 80x86 AAA, AAS, AAM, and AAD Instructions .....        | 900 |
| 4.4.4    | Packed Decimal Arithmetic Using the FPU .....              | 901 |
| 4.5      | Sample Program .....                                       | 903 |
| 4.6      | Putting It All Together .....                              | 906 |
| 5.1      | Chapter Overview .....                                     | 909 |
| 5.2      | What is Bit Data, Anyway? .....                            | 909 |
| 5.3      | Instructions That Manipulate Bits .....                    | 910 |
| 5.4      | The Carry Flag as a Bit Accumulator .....                  | 916 |
| 5.5      | Packing and Unpacking Bit Strings .....                    | 917 |
| 5.6      | Coalescing Bit Sets and Distributing Bit Strings .....     | 920 |
| 5.7      | Packed Arrays of Bit Strings .....                         | 922 |
| 5.8      | Searching for a Bit .....                                  | 923 |
| 5.9      | Counting Bits .....  | 925 |
| 5.10     | Reversing a Bit String .....                               | 927 |
| 5.11     | Merging Bit Strings .....                                  | 929 |
| 5.12     | Extracting Bit Strings .....                               | 930 |
| 5.13     | Searching for a Bit Pattern .....                          | 931 |
| 5.14     | The HLA Standard Library Bits Module .....                 | 932 |
| 5.15     | Putting It All Together .....                              | 933 |
| 6.1      | Chapter Overview .....                                     | 935 |
| 6.2      | The 80x86 String Instructions .....                        | 935 |
| 6.2.1    | How the String Instructions Operate .....                  | 936 |
| 6.2.2    | The REP/REPE/REPZ and REPNZ/REPNE Prefixes .....           | 936 |
| 6.2.3    | The Direction Flag .....                                   | 937 |
| 6.2.4    | The MOVS Instruction .....                                 | 938 |
| 6.2.5    | The CMPS Instruction .....                                 | 943 |
| 6.2.6    | The SCAS Instruction .....                                 | 946 |
| 6.2.7    | The STOS Instruction .....                                 | 946 |
| 6.2.8    | The LODS Instruction .....                                 | 947 |
| 6.2.9    | Building Complex String Functions from LODS and STOS ..... | 947 |
| 6.3      | Putting It All Together .....                              | 948 |
| 7.1      | Chapter Overview .....                                     | 949 |

|         |  |      |
|---------|--|------|
| 7.2     | Introduction to the Compile-Time Language (CTL)  | 949  |
| 7.3     | The #PRINT and #ERROR Statements                 | 951  |
| 7.4     | Compile-Time Constants and Variables             | 952  |
| 7.5     | Compile-Time Expressions and Operators           | 953  |
| 7.6     | Compile-Time Functions                           | 956  |
| 7.6.1   | Type Conversion Compile-time Functions           | 957  |
| 7.6.2   | Numeric Compile-Time Functions                   | 957  |
| 7.6.3   | Character Classification Compile-Time Functions  | 958  |
| 7.6.4   | Compile-Time String Functions                    | 958  |
| 7.6.5   | Compile-Time Pattern Matching Functions          | 958  |
| 7.6.6   | Compile-Time Symbol Information                  | 959  |
| 7.6.7   | Compile-Time Expression Classification Functions | 960  |
| 7.6.8   | Miscellaneous Compile-Time Functions             | 961  |
| 7.6.9   | Predefined Compile-Time Variables                | 961  |
| 7.6.10  | Compile-Time Type Conversions of TEXT Objects    | 961  |
| 7.7     | Conditional Compilation (Compile-Time Decisions) | 962  |
| 7.8     | Repetitive Compilation (Compile-Time Loops)      | 966  |
| 7.9     | Putting It All Together                          | 968  |
| 8.1     | Chapter Overview                                 | 969  |
| 8.2     | Macros (Compile-Time Procedures)                 | 969  |
| 8.2.1   | Standard Macros                                  | 969  |
| 8.2.2   | Macro Parameters                                 | 971  |
| 8.2.2.1 | Standard Macro Parameter Expansion               | 971  |
| 8.2.2.2 | Macros with a Variable Number of Parameters      | 974  |
| 8.2.2.3 | Required Versus Optional Macro Parameters        | 975  |
| 8.2.2.4 | The "#(" and ")#" Macro Parameter Brackets       | 976  |
| 8.2.2.5 | Eager vs. Deferred Macro Parameter Evaluation    | 977  |
| 8.2.3   | Local Symbols in a Macro                         | 981  |
| 8.2.4   | Macros as Compile-Time Procedures                | 985  |
| 8.2.5   | Multi-part (Context-Free) Macros                 | 985  |
| 8.2.6   | Simulating Function Overloading with Macros      | 990  |
| 8.3     | Writing Compile-Time "Programs"                  | 995  |
| 8.3.1   | Constructing Data Tables at Compile Time         | 996  |
| 8.3.2   | Unrolling Loops                                  | 999  |
| 8.4     | Using Macros in Different Source Files           | 1001 |
| 8.5     | Putting It All Together                          | 1001 |
| 9.1     | Chapter Overview                                 | 1003 |
| 9.2     | Introduction to DSELs in HLA                     | 1003 |
| 9.2.1   | Implementing the Standard HLA Control Structures | 1003 |
| 9.2.1.1 | The FOREVER Loop                                 | 1004 |
| 9.2.1.2 | The WHILE Loop                                   | 1007 |
| 9.2.1.3 | The IF Statement                                 | 1009 |
| 9.2.2   | The HLA SWITCH/CASE Statement                    | 1015 |
| 9.2.3   | A Modified WHILE Loop                            | 1026 |
| 9.2.4   | A Modified IF..ELSE..ENDIF Statement             | 1030 |
| 9.3     | Sample Program: A Simple Expression Compiler     | 1036 |



|  |      |
|--|------|
| 9.4 Putting It All Together .....                                | 1057 |
| 10.1 Chapter Overview .....                                      | 1059 |
| 10.2 General Principles .....                                    | 1059 |
| 10.3 Classes in HLA .....  | 1061 |
| 10.4 Objects .....   | 1063 |
| 10.5 Inheritance .....   | 1064 |
| 10.6 Overriding .....  | 1065 |
| 10.7 Virtual Methods vs. Static Procedures .....                 | 1066 |
| 10.8 Writing Class Methods, Iterators, and Procedures .....      | 1067 |
| 10.9 Object Implementation .....                                 | 1071 |
| 10.9.1 Virtual Method Tables .....                               | 1073 |
| 10.9.2 Object Representation with Inheritance .....              | 1075 |
| 10.10 Constructors and Object Initialization .....               | 1079 |
| 10.10.1 Dynamic Object Allocation Within the Constructor .....   | 1081 |
| 10.10.2 Constructors and Inheritance .....                       | 1082 |
| 10.10.3 Constructor Parameters and Procedure Overloading .....   | 1085 |
| 10.11 Destructors .....  | 1086 |
| 10.12 HLA's “_initialize_” and “_finalize_” Strings .....        | 1087 |
| 10.13 Abstract Methods .....                                     | 1091 |
| 10.14 Run-time Type Information (RTTI) .....                     | 1094 |
| 10.15 Calling Base Class Methods .....                           | 1095 |
| 10.16 Sample Program .....                                       | 1096 |
| 10.17 Putting It All Together .....                              | 1112 |
| 11.1 Chapter Overview .....                                      | 1113 |
| 11.2 Determining if a CPU Supports the MMX Instruction Set ..... | 1113 |
| 11.3 The MMX Programming Environment .....                       | 1114 |
| 11.3.1 The MMX Registers .....                                   | 1114 |
| 11.3.2 The MMX Data Types .....                                  | 1116 |
| 11.4 The Purpose of the MMX Instruction Set .....                | 1117 |
| 11.5 Saturation Arithmetic and Wraparound Mode .....             | 1118 |
| 11.6 MMX Instruction Operands .....                              | 1118 |
| 11.7 MMX Technology Instructions .....                           | 1123 |
| 11.7.1 MMX Data Transfer Instructions .....                      | 1123 |
| 11.7.2 MMX Conversion Instructions .....                         | 1123 |
| 11.7.3 MMX Packed Arithmetic Instructions .....                  | 1131 |
| 11.7.4 MMX Logic Instructions .....                              | 1133 |
| 11.7.5 MMX Comparison Instructions .....                         | 1134 |
| 11.7.6 MMX Shift Instructions .....                              | 1138 |
| 11.8 The EMMS Instruction .....                                  | 1139 |
| 11.9 The MMX Programming Paradigm .....                          | 1140 |
| 11.10 Putting It All Together .....                              | 1148 |
| 12.1 Chapter Overview .....                                      | 1151 |



|          |  |      |
|----------|--|------|
| 12.2     | Mixing HLA and MASM/Gas Code in the Same Program .....                     | 1151 |
| 12.2.1   | In-Line (MASM/Gas) Assembly Code in Your HLA Programs ..                   | 1151 |
| 12.2.2   | Linking MASM/Gas-Assembled Modules with HLA Modules ..                     | 1154 |
| 12.3     | Programming in Delphi/Kylix and HLA .....                                  | 1157 |
| 12.3.1   | Linking HLA Modules With Delphi Programs .....                             | 1158 |
| 12.3.2   | Register Preservation .....  | 1161 |
| 12.3.3   | Function Results .....   | 1161 |
| 12.3.4   | Calling Conventions .....  | 1167 |
| 12.3.5   | Pass by Value, Reference, CONST, and OUT in Delphi .....                   | 1172 |
| 12.3.6   | Scalar Data Type Correspondence Between Delphi and HLA ....                | 1173 |
| 12.3.7   | Passing String Data Between Delphi and HLA Code .....                      | 1175 |
| 12.3.8   | Passing Record Data Between HLA and Delphi .....                           | 1177 |
| 12.3.9   | Passing Set Data Between Delphi and HLA .....                              | 1181 |
| 12.3.10  | Passing Array Data Between HLA and Delphi .....                            | 1181 |
| 12.3.11  | Delphi Limitations When Linking with (Non-TASM) Assembly Code .....        | 1182 |
| 12.3.12  | Referencing Delphi Objects from HLA Code .....                             | 1182 |
| 12.4     | Programming in C/C++ and HLA .....   | 1182 |
| 12.4.1   | Linking HLA Modules With C/C++ Programs .....                              | 1183 |
| 12.4.2   | Register Preservation .....  | 1186 |
| 12.4.3   | Function Results .....   | 1186 |
| 12.4.4   | Calling Conventions .....  | 1186 |
| 12.4.5   | Pass by Value and Reference in C/C++ .....                                 | 1189 |
| 12.4.6   | Scalar Data Type Correspondence Between C/C++ and HLA ....                 | 1189 |
| 12.4.7   | Passing String Data Between C/C++ and HLA Code .....                       | 1191 |
| 12.4.8   | Passing Record/Structure Data Between HLA and C/C++ .....                  | 1191 |
| 12.4.9   | Passing Array Data Between HLA and C/C++ .....                             | 1192 |
| 12.5     | Putting It All Together .....  | 1193 |
| 13.1     | Questions .....  | 1195 |
| 13.2     | Programming Problems .....   | 1203 |
| 13.3     | Laboratory Exercises .....   | 1212 |
| 13.3.1   | Dynamically Nested TRY..ENDTRY Statements .....                            | 1213 |
| 13.3.2   | The TRY..ENDTRY Unprotected Section .....                                  | 1214 |
| 13.3.3   | Performance of SWITCH Statement .....                                      | 1215 |
| 13.3.4   | Complete Versus Short Circuit Boolean Evaluation .....                     | 1219 |
| 13.3.5   | Conversion of High Level Language Statements to Pure Assembly .....        | 1222 |
| 13.3.6   | Activation Record Exercises .....  | 1222 |
| 13.3.6.1 | Automatic Activation Record Generation and Access .....                    | 1222 |
| 13.3.6.2 | The _vars_ and _parms_ Constants .....                                     | 1224 |
| 13.3.6.3 | Manually Constructing an Activation Record .....                           | 1226 |
| 13.3.7   | Reference Parameter Exercise .....   | 1228 |
| 13.3.8   | Procedural Parameter Exercise .....  | 1231 |
| 13.3.9   | Iterator Exercises .....   | 1234 |
| 13.3.10  | Performance of Multiprecision Multiplication and Division Operations ..... | 1237 |
| 13.3.11  | Performance of the Extended Precision NEG Operation .....                  | 1237 |
| 13.3.12  | Testing the Extended Precision Input Routines .....                        | 1238 |
| 13.3.13  | Illegal Decimal Operations .....   | 1238 |
| 13.3.14  | MOVS Performance Exercise #1 .....   | 1238 |
| 13.3.15  | MOVS Performance Exercise #2 .....   | 1240 |
| 13.3.16  | Memory Performance Exercise .....  | 1242 |

|         |  |      |
|---------|--|------|
| 13.3.17 | The Performance of Length-Prefixed vs. Zero-Terminated Strings | 1243 |
| 13.3.18 | Introduction to Compile-Time Programs                          | 1249 |
| 13.3.19 | Conditional Compilation and Debug Code                         | 1250 |
| 13.3.20 | The Assert Macro   | 1252 |
| 13.3.21 | Demonstration of Compile-Time Loops (#while)                   | 1254 |
| 13.3.22 | Writing a Trace Macro  | 1256 |
| 13.3.23 | Overloading  | 1258 |
| 13.3.24 | Multi-part Macros and RatASM (Rational Assembly)               | 1261 |
| 13.3.25 | Virtual Methods vs. Static Procedures in a Class               | 1264 |
| 13.3.26 | Using the _initialize_ and _finalize_ Strings in a Program     | 1267 |
| 13.3.27 | Using RTTI in a Program  | 1270 |
| 1.1     | Chapter Overview   | 1279 |
| 1.2     | First Class Objects  | 1279 |
| 1.3     | Thunks   | 1281 |
| 1.4     | Initializing Thunks  | 1282 |
| 1.5     | Manipulating Thunks  | 1283 |
| 1.5.1   | Assigning Thunks   | 1283 |
| 1.5.2   | Comparing Thunks   | 1284 |
| 1.5.3   | Passing Thunks as Parameters                                   | 1285 |
| 1.5.4   | Returning Thunks as Function Results                           | 1286 |
| 1.6     | Activation Record Lifetimes and Thunks                         | 1288 |
| 1.7     | Comparing Thunks and Objects                                   | 1289 |
| 1.8     | An Example of a Thunk Using the Fibonacci Function             | 1289 |
| 1.9     | Thunks and Artificial Intelligence Code                        | 1294 |
| 1.10    | Thunks as Triggers   | 1296 |
| 1.11    | Jumping Out of a Thunk   | 1299 |
| 1.12    | Handling Exceptions with Thunks                                | 1302 |
| 1.13    | Using Thunks in an Appropriate Manner                          | 1302 |
| 1.14    | Putting It All Together  | 1303 |
| 2.1     | Chapter Overview   | 1305 |
| 2.2     | Review of Iterators  | 1305 |
| 2.2.1   | Implementing Iterators Using In-Line Expansion                 | 1307 |
| 2.2.2   | Implementing Iterators with Resume Frames                      | 1308 |
| 2.3     | Other Possible Iterator Implementations                        | 1314 |
| 2.4     | Breaking Out of a FOREACH Loop                                 | 1316 |
| 2.5     | An Iterator Implementation of the Fibonacci Number Generator   | 1317 |
| 2.6     | Iterators and Recursion  | 1323 |
| 2.7     | Calling Other Procedures Within an Iterator                    | 1327 |
| 2.8     | Iterators Within Classes                                       | 1327 |
| 2.9     | Putting It Altogether  | 1327 |
| 3.1     | Chapter Overview   | 1329 |
| 3.2     | Coroutines   | 1329 |
| 3.3     | Parameters and Register Values in Coroutine Calls              | 1334 |

|   |      |
|---|------|
| 3.4 Recursion, Reentrancy, and Variables .....                                      | 1335 |
| 3.5 Generators .....  | 1337 |
| 3.6 Exceptions and Coroutines .....   | 1340 |
| 3.7 Putting It All Together .....   | 1340 |
| 4.1 Chapter Overview .....  | 1341 |
| 4.2 Parameters .....  | 1341 |
| 4.3 Where You Can Pass Parameters .....   | 1341 |
| 4.3.1 Passing Parameters in (Integer) Registers .....                               | 1342 |
| 4.3.2 Passing Parameters in FPU and MMX Registers .....                             | 1345 |
| 4.3.3 Passing Parameters in Global Variables .....                                  | 1346 |
| 4.3.4 Passing Parameters on the Stack .....   | 1347 |
| 4.3.5 Passing Parameters in the Code Stream .....                                   | 1351 |
| 4.3.6 Passing Parameters via a Parameter Block .....                                | 1353 |
| 4.4 How You Can Pass Parameters .....   | 1354 |
| 4.4.1 Pass by Value-Result .....  | 1354 |
| 4.4.2 Pass by Result .....  | 1359 |
| 4.4.3 Pass by Name .....  | 1360 |
| 4.4.4 Pass by Lazy-Evaluation .....   | 1362 |
| 4.5 Passing Parameters as Parameters to Another Procedure .....                     | 1363 |
| 4.5.1 Passing Reference Parameters to Other Procedures .....                        | 1363 |
| 4.5.2 Passing Value-Result and Result Parameters as Parameters .....                | 1365 |
| 4.5.3 Passing Name Parameters to Other Procedures .....                             | 1365 |
| 4.5.4 Passing Lazy Evaluation Parameters as Parameters .....                        | 1366 |
| 4.5.5 Parameter Passing Summary .....   | 1366 |
| 4.6 Variable Parameter Lists .....  | 1368 |
| 4.7 Function Results .....  | 1370 |
| 4.7.1 Returning Function Results in a Register .....                                | 1370 |
| 4.7.2 Returning Function Results on the Stack .....                                 | 1371 |
| 4.7.3 Returning Function Results in Memory Locations .....                          | 1371 |
| 4.7.4 Returning Large Function Results .....  | 1372 |
| 4.8 Putting It All Together .....   | 1372 |
| 5.1 Chapter Overview .....  | 1375 |
| 5.2 Lexical Nesting, Static Links, and Displays .....                               | 1375 |
| 5.2.1 Scope .....   | 1375 |
| 5.2.2 Unit Activation, Address Binding, and Variable Lifetime .....                 | 1376 |
| 5.2.3 Static Links .....  | 1377 |
| 5.2.4 Accessing Non-Local Variables Using Static Links .....                        | 1382 |
| 5.2.5 Nesting Procedures in HLA .....   | 1384 |
| 5.2.6 The Display .....   | 1387 |
| 5.2.7 The 80x86 ENTER and LEAVE Instructions .....                                  | 1391 |
| 5.3 Passing Variables at Different Lex Levels as Parameters. ....                   | 1394 |
| 5.3.1 Passing Parameters by Value .....   | 1394 |
| 5.3.2 Passing Parameters by Reference, Result, and Value-Result ...                 | 1395 |
| 5.3.3 Passing Parameters by Name and Lazy-Evaluation in a Block Structured Language | 1395 |
| 5.4 Passing Procedures as Parameters .....  | 1396 |

|   |      |
|---|------|
| 5.5 Faking Intermediate Variable Access .....                 | 1396 |
| 5.6 Putting It All Together .....                             | 1397 |
| 6.1 Questions .....   | 1399 |
| 6.2 Programming Problems .....                                | 1402 |
| C.1 Introduction .....  | 1411 |
| C.1.1 Intended Audience .....                                 | 1411 |
| C.1.2 Readability Metrics .....                               | 1411 |
| C.1.3 How to Achieve Readability .....                        | 1412 |
| C.1.4 How This Document is Organized .....                    | 1413 |
| C.1.5 Guidelines, Rules, Enforced Rules, and Exceptions ..... | 1413 |
| C.1.6 Source Language Concerns .....                          | 1414 |
| C.2 Program Organization .....                                | 1414 |
| C.2.1 Library Functions .....                                 | 1414 |
| C.2.2 Common Object Modules .....                             | 1415 |
| C.2.3 Local Modules .....                                     | 1415 |
| C.2.4 Program Make Files .....                                | 1416 |
| C.3 Module Organization .....                                 | 1417 |
| C.3.1 Module Attributes .....                                 | 1417 |
| C.3.1.1 Module Cohesion .....                                 | 1417 |
| C.3.1.2 Module Coupling .....                                 | 1418 |
| C.3.1.3 Physical Organization of Modules .....                | 1418 |
| C.3.1.4 Module Interface .....                                | 1419 |
| C.4 Program Unit Organization .....                           | 1420 |
| C.4.1 Routine Cohesion .....                                  | 1420 |
| C.4.2 Routine Coupling .....                                  | 1421 |
| C.4.3 Routine Size .....                                      | 1421 |
| C.5 Statement Organization .....                              | 1422 |
| C.5.1 Writing “Pure” Assembly Code .....                      | 1422 |
| C.5.2 Using HLA’s High Level Control Statements .....         | 1424 |
| C.6 Comments .....  | 1430 |
| C.6.1 What is a Bad Comment? .....                            | 1430 |
| C.6.2 What is a Good Comment? .....                           | 1431 |
| C.6.3 Endline vs. Standalone Comments .....                   | 1432 |
| C.6.4 Unfinished Code .....                                   | 1433 |
| C.6.5 Cross References in Code to Other Documents .....       | 1434 |
| C.7 Names, Instructions, Operators, and Operands .....        | 1435 |
| C.7.1 Names .....   | 1435 |
| C.7.1.1 Naming Conventions .....                              | 1437 |
| C.7.1.2 Alphabetic Case Considerations .....                  | 1437 |
| C.7.1.3 Abbreviations .....                                   | 1438 |
| C.7.1.4 The Position of Components Within an Identifier ..... | 1439 |
| C.7.1.5 Names to Avoid .....                                  | 1440 |
| C.7.1.6 Special Identifiers .....                             | 1441 |
| C.7.2 Instructions, Directives, and Pseudo-Opcodes .....      | 1442 |
| C.7.2.1 Choosing the Best Instruction Sequence .....          | 1442 |
| C.7.2.2 Control Structures .....                              | 1443 |
| C.7.2.3 Instruction Synonyms .....                            | 1446 |

|  |      |
|--|------|
| C.8 Data Types .....   | 1447 |
| C.8.1 Declaring Structures in Assembly Language .....                    | 1447 |
| H.1 Conversion Functions .....   | 1493 |
| H.2 Numeric Functions .....  | 1495 |
| H.3 Date/Time Functions .....  | 1498 |
| H.4 Classification Functions .....                                       | 1498 |
| H.5 String and Character Set Functions .....                             | 1500 |
| H.6 Pattern Matching Functions .....                                     | 1504 |
| H.6.1 String/Cset Pattern Matching Functions .....                       | 1505 |
| H.6.2 String/Character Pattern Matching Functions .....                  | 1510 |
| H.6.3 String/Case Insensitive Character Pattern Matching Functions ..... | 1514 |
| H.6.4 String/String Pattern Matching Functions .....                     | 1516 |
| H.6.5 String/Misc Pattern Matching Functions .....                       | 1517 |
| H.7 HLA Information and Symbol Table Functions .....                     | 1521 |
| H.8 Compile-Time Variables .....   | 1527 |
| H.9 Miscellaneous Compile-Time Functions .....                           | 1528 |
| J.1 The @TRACE Pseudo-Variable .....                                     | 1533 |
| J.2 The Assert Macro .....   | 1536 |
| L.1 The HLA Standard Library .....                                       | 1541 |
| L.2 Compiling to MASM Code -- The Final Word .....                       | 1542 |
| L.3 The HLA if..then..endif Statement, Part I .....                      | 1547 |
| L.4 Boolean Expressions in HLA Control Structures .....                  | 1548 |
| L.5 The JT/JF Pseudo-Instructions .....                                  | 1554 |
| L.6 The HLA if..then..elseif..else..endif Statement, Part II .....       | 1554 |
| L.7 The While Statement .....  | 1558 |
| L.8 repeat..until .....  | 1559 |
| L.9 for..endfor .....  | 1559 |
| L.10 forever..endfor .....   | 1559 |
| L.11 break, breakif .....  | 1559 |
| L.12 continue, continueif .....  | 1559 |
| L.13 begin..end, exit, exitif .....                                      | 1559 |
| L.14 foreach..endfor .....   | 1559 |
| L.15 try..unprotect..exception..anyexception..endtry, raise .....        | 1559 |

