To Jack and Candy Irvine
CONTENTS

Preface  xxiii

1  Basic Concepts  1
1.1  Welcome to Assembly Language  1
   1.1.1  Good Questions to Ask  2
   1.1.2  Assembly Language Applications  5
   1.1.3  Section Review  6
1.2  Virtual Machine Concept  7
   1.2.1  History of PC Assemblers  9
   1.2.2  Section Review  9
1.3  Data Representation  9
   1.3.1  Binary Numbers  10
   1.3.2  Binary Addition  11
   1.3.3  Integer Storage Sizes  12
   1.3.4  Hexadecimal Integers  13
   1.3.5  Signed Integers  14
   1.3.6  Character Storage  16
   1.3.7  Section Review  18
1.4  Boolean Operations  20
   1.4.1  Truth Tables for Boolean Functions  22
   1.4.2  Section Review  23
1.5  Chapter Summary  23

2  IA-32 Processor Architecture  25
2.1  General Concepts  25
   2.1.1  Basic Microcomputer Design  26
   2.1.2  Instruction Execution Cycle  27
   2.1.3  Reading from Memory  30
   2.1.4  How Programs Run  31
   2.1.5  Section Review  32
2.2  IA-32 Processor Architecture  33
   2.2.1  Modes of Operation  33
   2.2.2  Basic Execution Environment  34
2.2.3 Floating-Point Unit 36
2.2.4 Intel Microprocessor History 37
2.2.5 Section Review 38

2.3 IA-32 Memory Management 39
2.3.1 Real-Address Mode 39
2.3.2 Protected Mode 41
2.3.3 Section Review 43

2.4 Components of an IA-32 Microcomputer 43
2.4.1 Motherboard 43
2.4.2 Video Output 44
2.4.3 Memory 45
2.4.4 Input-Output Ports and Device Interfaces 45
2.4.5 Section Review 46

2.5 Input-Output System 46
2.5.1 How It All Works 46
2.5.2 Section Review 48

2.6 Chapter Summary 49

3 Assembly Language Fundamentals 51

3.1 Basic Elements of Assembly Language 51
3.1.1 Integer Constants 52
3.1.2 Integer Expressions 52
3.1.3 Real Number Constants 53
3.1.4 Character Constants 54
3.1.5 String Constants 54
3.1.6 Reserved Words 54
3.1.7 Identifiers 54
3.1.8 Directives 55
3.1.9 Instructions 55
3.1.10 The NOP (No Operation) Instruction 57
3.1.11 Section Review 58

3.2 Example: Adding Three Integers 58
3.2.1 Alternative Version of AddSub 60
3.2.2 Program Template 61
3.2.3 Section Review 61

3.3 Assembling, Linking, and Running Programs 62
3.3.1 The Assemble-Link-Execute Cycle 62
3.3.2 Section Review 64

3.4 Defining Data 64
3.4.1 Intrinsic Data Types 64
3.4.2 Data Definition Statement 64
3.4.3 Defining BYTE and SBYTE Data 66
6.6.2 Validating a Signed Integer 180
6.6.3 Section Review 183

6.7 Decision Directives 184
6.7.1 Signed and Unsigned Comparisons 185
6.7.2 Compound Expressions 186
6.7.3 .REPEAT and .WHILE Directives 188

6.8 Chapter Summary 189
6.9 Programming Exercises 190

7 Integer Arithmetic 193

7.1 Introduction 193

7.2 Shift and Rotate Instructions 194
7.2.1 Logical Shifts and Arithmetic Shifts 194
7.2.2 SHL Instruction 195
7.2.3 SHR Instruction 196
7.2.4 SAL and SAR Instructions 196
7.2.5 ROL Instruction 197
7.2.6 ROR Instruction 198
7.2.7 RCL and RCR Instructions 198
7.2.8 Signed Overflow 199
7.2.9 SHLD/SHRD Instructions 199
7.2.10 Section Review 200

7.3 Shift and Rotate Applications 201
7.3.1 Shifting Multiple Doublewords 201
7.3.2 Binary Multiplication 202
7.3.3 Displaying Binary Bits 202
7.3.4 Isolating MS-DOS File Date Fields 203
7.3.5 Section Review 203

7.4 Multiplication and Division Operations 204
7.4.1 MUL Instruction 204
7.4.2 IMUL Instruction 205
7.4.3 Benchmarking Multiplication Operations 207
7.4.4 DIV Instruction 208
7.4.5 Signed Integer Division 209
7.4.6 Implementing Arithmetic Expressions 211
7.4.7 Section Review 212

7.5 Extended Addition and Subtraction 213
7.5.1 ADC Instruction 213
7.5.2 Extended Addition Example 213
7.5.3 SBB Instruction 214
7.5.4 Section Review 215
<table>
<thead>
<tr>
<th>7.6</th>
<th>ASCII and Unpacked Decimal Arithmetic</th>
<th>215</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.6.1</td>
<td>AAA Instruction</td>
<td>216</td>
</tr>
<tr>
<td>7.6.2</td>
<td>AAS Instruction</td>
<td>218</td>
</tr>
<tr>
<td>7.6.3</td>
<td>AAM Instruction</td>
<td>218</td>
</tr>
<tr>
<td>7.6.4</td>
<td>AAD Instruction</td>
<td>218</td>
</tr>
<tr>
<td>7.6.5</td>
<td>Section Review</td>
<td>219</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7.7</th>
<th>Packed Decimal Arithmetic</th>
<th>219</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.7.1</td>
<td>DAA Instruction</td>
<td>219</td>
</tr>
<tr>
<td>7.7.2</td>
<td>DAS Instruction</td>
<td>220</td>
</tr>
<tr>
<td>7.7.3</td>
<td>Section Review</td>
<td>220</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7.8</th>
<th>Chapter Summary</th>
<th>221</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.9</td>
<td>Programming Exercises</td>
<td>222</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8.0</th>
<th>Advanced Procedures</th>
<th>224</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Introduction</td>
<td>224</td>
</tr>
<tr>
<td>8.2</td>
<td>Stack Frames</td>
<td>225</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Stack Parameters</td>
<td>225</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Local Variables</td>
<td>233</td>
</tr>
<tr>
<td>8.2.3</td>
<td>ENTER and LEAVE Instructions</td>
<td>236</td>
</tr>
<tr>
<td>8.2.4</td>
<td>LOCAL Directive</td>
<td>237</td>
</tr>
<tr>
<td>8.2.5</td>
<td>WriteStackFrame Procedure</td>
<td>240</td>
</tr>
<tr>
<td>8.2.6</td>
<td>Section Review</td>
<td>241</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8.3</th>
<th>Recursion</th>
<th>242</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.1</td>
<td>Recursively Calculating a Sum</td>
<td>243</td>
</tr>
<tr>
<td>8.3.2</td>
<td>Calculating a Factorial</td>
<td>243</td>
</tr>
<tr>
<td>8.3.3</td>
<td>Section Review</td>
<td>245</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8.4</th>
<th>.MODEL Directive</th>
<th>246</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.1</td>
<td>Language Specifiers</td>
<td>247</td>
</tr>
<tr>
<td>8.4.2</td>
<td>Section Review</td>
<td>248</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8.5</th>
<th>INVOKE, ADDR, PROC, and PROTO (Optional)</th>
<th>248</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.5.1</td>
<td>INVOKE Directive</td>
<td>248</td>
</tr>
<tr>
<td>8.5.2</td>
<td>ADDR Operator</td>
<td>249</td>
</tr>
<tr>
<td>8.5.3</td>
<td>PROC Directive</td>
<td>250</td>
</tr>
<tr>
<td>8.5.4</td>
<td>PROTO Directive</td>
<td>253</td>
</tr>
<tr>
<td>8.5.5</td>
<td>Parameter Classifications</td>
<td>255</td>
</tr>
<tr>
<td>8.5.6</td>
<td>Example: Exchanging Two Integers</td>
<td>256</td>
</tr>
<tr>
<td>8.5.7</td>
<td>Debugging Tips</td>
<td>256</td>
</tr>
<tr>
<td>8.5.8</td>
<td>Section Review</td>
<td>257</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8.6</th>
<th>Creating Multimodule Programs</th>
<th>258</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.6.1</td>
<td>Hiding and Exporting Procedure Names</td>
<td>258</td>
</tr>
<tr>
<td>8.6.2</td>
<td>Calling External Procedures</td>
<td>258</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>8.6.3</td>
<td>Using Variables and Symbols Across Module Boundaries</td>
<td>259</td>
</tr>
<tr>
<td>8.6.4</td>
<td>Example: ArraySum Program</td>
<td>260</td>
</tr>
<tr>
<td>8.6.5</td>
<td>Creating the Modules Using Extern</td>
<td>261</td>
</tr>
<tr>
<td>8.6.6</td>
<td>Creating the Modules Using INVOKE and PROTO</td>
<td>264</td>
</tr>
<tr>
<td>8.6.7</td>
<td>Section Review</td>
<td>266</td>
</tr>
<tr>
<td>8.7</td>
<td>Chapter Summary</td>
<td>267</td>
</tr>
<tr>
<td>8.8</td>
<td>Programming Exercises</td>
<td>268</td>
</tr>
<tr>
<td>9</td>
<td>Strings and Arrays</td>
<td>269</td>
</tr>
<tr>
<td>9.1</td>
<td>Introduction</td>
<td>269</td>
</tr>
<tr>
<td>9.2</td>
<td>String Primitive Instructions</td>
<td>270</td>
</tr>
<tr>
<td>9.2.1</td>
<td>MOVSB, MOVSW, and MOVSD</td>
<td>271</td>
</tr>
<tr>
<td>9.2.2</td>
<td>CMPSB, CMPSW, and CMPSD</td>
<td>272</td>
</tr>
<tr>
<td>9.2.3</td>
<td>SCASB, SCASW, and SCASD</td>
<td>274</td>
</tr>
<tr>
<td>9.2.4</td>
<td>STOSB, STOSW, and STOSD</td>
<td>274</td>
</tr>
<tr>
<td>9.2.5</td>
<td>LODSB, LODSW, and LODSD</td>
<td>275</td>
</tr>
<tr>
<td>9.2.6</td>
<td>Section Review</td>
<td>275</td>
</tr>
<tr>
<td>9.3</td>
<td>Selected String Procedures</td>
<td>276</td>
</tr>
<tr>
<td>9.3.1</td>
<td>Str_compare Procedure</td>
<td>276</td>
</tr>
<tr>
<td>9.3.2</td>
<td>Str_length Procedure</td>
<td>277</td>
</tr>
<tr>
<td>9.3.3</td>
<td>Str_copy Procedure</td>
<td>278</td>
</tr>
<tr>
<td>9.3.4</td>
<td>Str_trim Procedure</td>
<td>278</td>
</tr>
<tr>
<td>9.3.5</td>
<td>Str_ucase Procedure</td>
<td>279</td>
</tr>
<tr>
<td>9.3.6</td>
<td>String Library Demo Program</td>
<td>280</td>
</tr>
<tr>
<td>9.3.7</td>
<td>Section Review</td>
<td>282</td>
</tr>
<tr>
<td>9.4</td>
<td>Two-Dimensional Arrays</td>
<td>282</td>
</tr>
<tr>
<td>9.4.1</td>
<td>Ordering of Rows and Columns</td>
<td>282</td>
</tr>
<tr>
<td>9.4.2</td>
<td>Base-Index Operands</td>
<td>283</td>
</tr>
<tr>
<td>9.4.3</td>
<td>Base-Index-Displacement Operands</td>
<td>285</td>
</tr>
<tr>
<td>9.4.4</td>
<td>Section Review</td>
<td>285</td>
</tr>
<tr>
<td>9.5</td>
<td>Searching and Sorting Integer Arrays</td>
<td>285</td>
</tr>
<tr>
<td>9.5.1</td>
<td>Bubble Sort</td>
<td>286</td>
</tr>
<tr>
<td>9.5.2</td>
<td>Binary Search</td>
<td>287</td>
</tr>
<tr>
<td>9.5.3</td>
<td>Section Review</td>
<td>293</td>
</tr>
<tr>
<td>9.6</td>
<td>Chapter Summary</td>
<td>294</td>
</tr>
<tr>
<td>9.7</td>
<td>Programming Exercises</td>
<td>295</td>
</tr>
<tr>
<td>10</td>
<td>Structures and Macros</td>
<td>299</td>
</tr>
<tr>
<td>10.1</td>
<td>Structures</td>
<td>299</td>
</tr>
<tr>
<td>10.1.1</td>
<td>Defining Structures</td>
<td>300</td>
</tr>
</tbody>
</table>
10.1.2 Declaring Structure Variables 301
10.1.3 Referencing Structure Variables 302
10.1.4 Example: Displaying the System Time 305
10.1.5 Structures Containing Structures 307
10.1.6 Example: Drunkard’s Walk 307
10.1.7 Declaring and Using Unions 310
10.1.8 Section Review 312

10.2 Macros 313
10.2.1 Overview 313
10.2.2 Defining Macros 313
10.2.3 Invoking Macros 314
10.2.4 Additional Macro Features 315
10.2.5 Using the Book’s Macro Library 318
10.2.6 Example Program: Wrappers 324
10.2.7 Section Review 325

10.3 Conditional-Assembly Directives 326
10.3.1 Checking for Missing Arguments 326
10.3.2 Default Argument Initializers 328
10.3.3 Boolean Expressions 328
10.3.4 IF, ELSE, and ENDIF Directives 328
10.3.5 The IFIDN and IFIDNI Directives 329
10.3.6 Example: Summing a Matrix Row 330
10.3.7 Special Operators 333
10.3.8 Macro Functions 336
10.3.9 Section Review 337

10.4 Defining Repeat Blocks 338
10.4.1 WHILE Directive 338
10.4.2 REPEAT Directive 338
10.4.3 FOR Directive 339
10.4.4 FORC Directive 340
10.4.5 Example: Linked List 340
10.4.6 Section Review 342

10.5 Chapter Summary 342

10.6 Programming Exercises 343

11 MS-Windows Programming 346

11.1 Win32 Console Programming 346
11.1.1 Background Information 347
11.1.2 Win32 Console Functions 350
11.1.3 Displaying a Message Box 352
11.1.4 Console Input 354
11.1.5 Console Output 360
11.1.6 Reading and Writing Files 361
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1.7 File I/O in the Irvine32 Library</td>
<td>365</td>
</tr>
<tr>
<td>11.1.8 Testing the File I/O Procedures</td>
<td>367</td>
</tr>
<tr>
<td>11.1.9 Console Window Manipulation</td>
<td>370</td>
</tr>
<tr>
<td>11.1.10 Controlling the Cursor</td>
<td>373</td>
</tr>
<tr>
<td>11.1.11 Controlling the Text Color</td>
<td>373</td>
</tr>
<tr>
<td>11.1.12 Time and Date Functions</td>
<td>375</td>
</tr>
<tr>
<td>11.1.13 Section Review</td>
<td>379</td>
</tr>
<tr>
<td>11.2 Writing a Graphical Windows Application</td>
<td>379</td>
</tr>
<tr>
<td>11.2.1 Necessary Structures</td>
<td>380</td>
</tr>
<tr>
<td>11.2.2 The MessageBox Function</td>
<td>381</td>
</tr>
<tr>
<td>11.2.3 The WinMain Procedure</td>
<td>382</td>
</tr>
<tr>
<td>11.2.4 The WinProc Procedure</td>
<td>382</td>
</tr>
<tr>
<td>11.2.5 The ErrorHandler Procedure</td>
<td>383</td>
</tr>
<tr>
<td>11.2.6 Program Listing</td>
<td>383</td>
</tr>
<tr>
<td>11.2.7 Section Review</td>
<td>386</td>
</tr>
<tr>
<td>11.3 Dynamic Memory Allocation</td>
<td>387</td>
</tr>
<tr>
<td>11.3.1 HeapTest Programs</td>
<td>390</td>
</tr>
<tr>
<td>11.3.2 Section Review</td>
<td>393</td>
</tr>
<tr>
<td>11.4 IA-32 Memory Management</td>
<td>393</td>
</tr>
<tr>
<td>11.4.1 Linear Addresses</td>
<td>394</td>
</tr>
<tr>
<td>11.4.2 Page Translation</td>
<td>397</td>
</tr>
<tr>
<td>11.4.3 Section Review</td>
<td>398</td>
</tr>
<tr>
<td>11.5 Chapter Summary</td>
<td>399</td>
</tr>
<tr>
<td>11.6 Programming Exercises</td>
<td>400</td>
</tr>
<tr>
<td>12 High-Level Language Interface</td>
<td>402</td>
</tr>
<tr>
<td>12.1 Introduction</td>
<td>402</td>
</tr>
<tr>
<td>12.1.1 General Conventions</td>
<td>402</td>
</tr>
<tr>
<td>12.1.2 Section Review</td>
<td>403</td>
</tr>
<tr>
<td>12.2 Inline Assembly Code</td>
<td>404</td>
</tr>
<tr>
<td>12.2.1 __asm Directive in Microsoft Visual C++</td>
<td>404</td>
</tr>
<tr>
<td>12.2.2 File Encryption Example</td>
<td>406</td>
</tr>
<tr>
<td>12.2.3 Section Review</td>
<td>409</td>
</tr>
<tr>
<td>12.3 Linking to C/C++ in Protected Mode</td>
<td>409</td>
</tr>
<tr>
<td>12.3.1 Using Assembly Language to Optimize C++ Code</td>
<td>410</td>
</tr>
<tr>
<td>12.3.2 Calling C and C++ Functions</td>
<td>415</td>
</tr>
<tr>
<td>12.3.3 Multiplication Table Example</td>
<td>416</td>
</tr>
<tr>
<td>12.3.4 Calling C Library Functions</td>
<td>419</td>
</tr>
<tr>
<td>12.3.5 Directory Listing Program</td>
<td>422</td>
</tr>
<tr>
<td>12.3.6 Section Review</td>
<td>423</td>
</tr>
<tr>
<td>12.4 Linking to C/C++ in Real-Address Mode</td>
<td>423</td>
</tr>
<tr>
<td>12.4.1 Linking to Borland C++</td>
<td>424</td>
</tr>
</tbody>
</table>
14.2.2 FAT16 469
14.2.3 FAT32 469
14.2.4 NTFS 470
14.2.5 Primary Disk Areas 470
14.2.6 Section Review 471

14.3 Disk Directory 472
14.3.1 MS-DOS Directory Structure 473
14.3.2 Long Filenames in MS-Windows 475
14.3.3 File Allocation Table (FAT) 476
14.3.4 Section Review 477

14.4 Reading and Writing Disk Sectors (7305h) 477
14.4.1 Sector Display Program 478
14.4.2 Section Review 482

14.5 System-Level File Functions 482
14.5.1 Get Disk Free Space (7303h) 483
14.5.2 Create Subdirectory (39h) 485
14.5.3 Remove Subdirectory (3Ah) 486
14.5.4 Set Current Directory (3Bh) 486
14.5.5 Get Current Directory (47h) 486
14.5.6 Get and Set File Attributes (7143h) 486
14.5.7 Section Review 487

14.6 Chapter Summary 487
14.7 Programming Exercises 488

15 BIOS-Level Programming 490

15.1 Introduction 490
15.1.1 BIOS Data Area 491

15.2 Keyboard Input with INT 16h 492
15.2.1 How the Keyboard Works 492
15.2.2 INT 16h Functions 493
15.2.3 Section Review 497

15.3 VIDEO Programming with INT 10h 498
15.3.1 Basic Background 498
15.3.2 Controlling the Color 499
15.3.3 INT 10h Video Functions 501
15.3.4 Library Procedure Examples 511
15.3.5 Section Review 512

15.4 Drawing Graphics Using INT 10h 512
15.4.1 INT 10h Pixel-Related Functions 513
15.4.2 DrawLine Program 514
15.4.3 Cartesian Coordinates Program  515
15.4.4 Converting Cartesian Coordinates to Screen Coordinates  517
15.4.5 Section Review  518

15.5 Memory-Mapped Graphics  519
15.5.1 Mode 13h: 320 × 200, 256 Colors  519
15.5.2 Memory-Mapped Graphics Program  520
15.5.3 Section Review  523

15.6 Mouse Programming  523
15.6.1 Mouse INT 33h Functions  523
15.6.2 Mouse Tracking Program  528
15.6.3 Section Review  532

15.7 Chapter Summary  533
15.8 Chapter Exercises  534

16 Expert MS-DOS Programming  536

16.1 Introduction  536

16.2 Defining Segments  537
16.2.1 Simplified Segment Directives  537
16.2.2 Explicit Segment Definitions  538
16.2.3 Segment Overrides  542
16.2.4 Combining Segments  542
16.2.5 Section Review  543

16.3 Runtime Program Structure  544
16.3.1 Program Segment Prefix  544
16.3.2 COM Programs  545
16.3.3 EXE Programs  546
16.3.4 Section Review  547

16.4 Interrupt Handling  548
16.4.1 Hardware Interrupts  549
16.4.2 Interrupt Control Instructions  550
16.4.3 Writing a Custom Interrupt Handler  551
16.4.4 Terminate and Stay Resident Programs  553
16.4.5 Application: The No_Reset Program  554
16.4.6 Section Review  557

16.5 Hardware Control Using I/O Ports  558
16.5.1 Input-Output Ports  558
16.5.2 PC Sound Program  558

16.6 Chapter Summary  560
Assembly Language for Intel-Based Computers, Fifth Edition, teaches assembly language programming and architecture for Intel IA-32 processors. It is an appropriate text for the following types of college courses:

- Assembly Language Programming
- Fundamentals of Computer Systems
- Fundamentals of Computer Architecture

Students use Intel or AMD processors and program with Microsoft Macro Assembler (MASM) 8.0, running on any of the following MS-Windows platforms: Windows 95, 98, Millenium, NT, 2000, and XP.

Although this book was originally designed as a programming textbook for college students, it has evolved over the last 15 years into much more. Many universities use the book for their introductory computer architecture courses. As a testament to its popularity, the fourth edition was translated into Korean, Chinese, French, Russian, and Polish.

Emphasis of Topics  This edition includes topics that lead naturally into subsequent courses in computer architecture, operating systems, and compiler writing:

- Virtual machine concept
- Elementary boolean operations
- Instruction execution cycle
- Memory access and handshaking
- Interrupts and polling
- Pipelining and superscalar concepts
- Hardware-based I/O
- Floating-point binary representation

Other topics relate specifically to Intel IA-32 architecture:

- IA-32 protected memory and paging
- Memory segmentation in real-address mode
- 16-bit interrupt handling
- MS-DOS and BIOS system calls (interrupts)
- IA-32 Floating-Point Unit architecture and programming
- IA-32 Instruction encoding

Certain examples presented in the book lend themselves to courses that occur later in a computer science curriculum:

- Searching and sorting algorithms
- High-level language structures
- Finite-state machines
- Code optimization examples
Improvements in the Fifth Edition  A number of improvements and new information have been added in this edition, listed in the following table by chapter number:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Improved explanation of the instruction execution cycle.</td>
</tr>
<tr>
<td>5</td>
<td>An expanded link library with additional subroutines to write rich user interfaces, calculate program timings, generate pseudorandom integers, and parse integer strings. The documentation of the library has greatly improved.</td>
</tr>
<tr>
<td>6</td>
<td>Improved explanation of conditional jump encoding and relative jump ranges.</td>
</tr>
<tr>
<td>7</td>
<td>Two-operand and three-operand IMUL instructions are added. Performance comparisons are shown for differing approaches to integer multiplication.</td>
</tr>
<tr>
<td>8</td>
<td>Completely redesigned so that low-level details of stack frames (activation records) are explained first before introducing MASM’s high-level INVOKE and PROC directives.</td>
</tr>
<tr>
<td>10</td>
<td>Improved documentation of the book’s macro library.</td>
</tr>
<tr>
<td>11</td>
<td>New topic: Dynamic memory allocation in MS-Windows applications. Improved coverage of file handling and error reporting in MS-Windows applications.</td>
</tr>
<tr>
<td>12</td>
<td>Improved coverage of calling C and C++ functions from assembly language.</td>
</tr>
<tr>
<td>17</td>
<td>Introduction to the IA-32 floating-point instruction set. Floating-point data types. IA-32 Instruction encoding and decoding.</td>
</tr>
</tbody>
</table>

Still a Programming Book  This book is still focused on its original mission: to teach students how to write and debug programs at the machine level. It will never replace a complete book on computer architecture, but it does give students the first-hand experience of writing software in an environment that teaches them how a computer works. Our premise is that students retain knowledge better when theory is combined with experience. In an engineering course, students construct prototypes; in a computer architecture course, students should write machine-level programs. In both cases, they have a memorable experience that gives them the confidence to work in any OS/machine-oriented environment.

Real Mode and Protected Mode  This edition emphasizes 32-bit protected mode, but it still has three chapters devoted to real-mode programming. For example, there is an entire chapter on BIOS programming for the keyboard, video display (including graphics), and mouse. Another chapter covers MS-DOS programming using interrupts (system calls). Students can benefit from programming directly to hardware and the BIOS.

The examples in the first half of the book are nearly all presented as 32-bit text-oriented applications running in protected mode using the flat memory model. This approach is wonderfully simple because it avoids the complications of segment-offset addressing. Specially marked paragraphs and popup boxes point out occasional differences between protected mode and real mode programming. Most differences are abstracted by the book’s parallel link libraries for real mode and protected mode programming.

Link Libraries  We supply two versions of the link library that students use for basic input-output, simulations, timing, and other useful stuff. The 32-bit version (Irvine32.lib) runs in protected mode, sending its output to the Win32 console. The 16-bit version (Irvine16.lib) runs in real-address mode. Full source code for the libraries is supplied on the book’s Web site. The link libraries are available only for convenience, not to prevent students from learning how to program input-output themselves. Students are encouraged to create their own libraries.
Included Software and Examples  All the example programs were tested with Microsoft Macro Assembler Version 8.0. The 32-bit C++ applications in Chapter 12 were tested with Microsoft Visual C++.NET. The real-address mode programs in Chapter 12 (linking to C++) were assembled with Borland Turbo Assembler (TASM).

Web Site Information  Updates and corrections to this book may be found at the book’s Web site, http://www.asmirvine.com, including additional programming projects for instructors to assign at the ends of chapters. If for some reason you cannot access this site, information about the book and a link to its current Web site can be found at www.prenhall.com by searching for the book title or for the author name “Kip Irvine.”

Overall Goals
The following goals of this book are designed to broaden the student’s interest and knowledge in topics related to assembly language:

• Intel IA-32 processor architecture and programming
• Real-address mode and protected mode programming
• Assembly language directives, macros, operators, and program structure
• Programming methodology, showing how to use assembly language to create system-level software tools and application programs
• Computer hardware manipulation
• Interaction between assembly language programs, the operating system, and other application programs

One of our goals is to help students approach programming problems with a machine-level mindset. It is important to think of the CPU as an interactive tool, and to learn to monitor its operation as directly as possible. A debugger is a programmer’s best friend, not only for catching errors, but as an educational tool that teaches about the CPU and operating system. We encourage students to look beneath the surface of high-level languages and to realize that most programming languages are designed to be portable and, therefore, independent of their host machines.

In addition to the short examples, this book contains hundreds of ready-to-run programs that demonstrate instructions or ideas as they are presented in the text. Reference materials, such as guides to MS-DOS interrupts and instruction mnemonics, are available at the end of the book.

Required Background  The reader should already be able to program confidently in at least one other programming language, preferably Java, C, or C++. One chapter covers C++ interfacing, so it is very helpful to have a compiler on hand. I have used this book in the classroom with majors in both computer science and management information systems, and it has been used elsewhere in engineering courses.

Features
Complete Program Listings  A companion CD-ROM contains all the source code from the examples in this book. Additional listings are available on the book’s Web page. An extensive link library is supplied with the book, containing more than 30 procedures that simplify user input-output, numeric processing, disk and file handling, and string handling. In the beginning stages of the course, students can use this library to enhance their programs. Later, they can create their own procedures and add them to the library.

Programming Logic  Two chapters emphasize boolean logic and bit-level manipulation. A conscious attempt is made to relate high-level programming logic to the low-level details of the machine. This approach helps students to create more efficient implementations and to better understand how compilers generate object code.
Hardware and Operating System Concepts  The first two chapters introduce basic hardware and data representation concepts, including binary numbers, CPU architecture, status flags, and memory mapping. A survey of the computer’s hardware and a historical perspective of the Intel processor family helps students to better understand their target computer system.

Structured Programming Approach  Beginning with Chapter 5, procedures and functional decomposition are emphasized. Students are given more complex programming exercises, requiring them to focus on design before starting to write code.

Disk Storage Concepts  Students learn the fundamental principles behind the disk storage system on MS-Windows–based systems from hardware and software points of view.

Creating Link Libraries  Students are free to add their own procedures to the book’s link library and create new libraries. They learn to use a toolbox approach to programming and to write code that is useful in more than one program.

Macros and Structures  A chapter is devoted to creating structures, unions, and macros, which are essential in assembly language and systems programming. Conditional macros with advanced operators serve to make the macros more professional.

Interfacing to High-Level Languages  A chapter is devoted to interfacing assembly language to C and C++. This is an important job skill for students who are likely to find jobs programming in high-level languages. They can learn to optimize their code and see examples of how C++ compilers optimize code.

Instructional Aids  All the program listings are available on disk and on the Web. Instructors are provided a test bank, answers to review questions, solutions to programming exercises, and a Microsoft PowerPoint slide presentation for each chapter.

Summary of Chapters

Chapters 1 to 8 contain a basic foundation of assembly language and should be covered in sequence. After that, you have a fair amount of freedom. The following chapter dependency graph shows how later chapters depend on knowledge gained from other chapters. Chapter 10 was split into two parts for this graph because no other chapter depends on one’s knowing how to create macros:

1. Basic Concepts: Applications of assembly language, basic concepts, machine language, and data representation.
2. IA-32 Processor Architecture: Basic microcomputer design, instruction execution cycle, IA-32 processor architecture, IA-32 memory management, components of a microcomputer, and the input-output system.
3. Assembly Language Fundamentals: Introduction to assembly language, linking and debugging, and defining constants and variables.
4. Data Transfers, Addressing, and Arithmetic: Simple data transfer and arithmetic instructions, assemble-link-execute cycle, operators, directives, expressions, JMP and LOOP instructions, and indirect addressing.

5. Procedures: Linking to an external library, description of the book’s link library, stack operations, defining and using procedures, flowcharts, and top-down structured design.

6. Conditional Processing: Boolean and comparison instructions, conditional jumps and loops, high-level logic structures, and finite state machines.

7. Integer Arithmetic: Shift and rotate instructions with useful applications, multiplication and division, extended addition and subtraction, and ASCII and packed decimal arithmetic.

8. Advanced Procedures: Stack parameters, local variables, advanced PROC and INVOKE directives, and recursion.

9. Strings and Arrays: String primitives, manipulating arrays of characters and integers, two-dimensional arrays, sorting, and searching.

10. Structures and Macros: Structures, macros, conditional assembly directives, and defining repeat blocks.

11. MS-Windows Programming: Protected mode memory management concepts, using the Microsoft Windows API to display text and colors, and dynamic memory allocation.

12. High-Level Language Interface: Parameter passing conventions, inline assembly code, and linking assembly language modules to C and C++ programs.

13. 16-Bit MS-DOS Programming: Calling MS-DOS interrupts for console and file input-output.

14. Disk Fundamentals: Disk storage systems, sectors, clusters, directories, file allocation tables, handling MS-DOS error codes, and drive and directory manipulation.

15. BIOS-Level Programming: Keyboard input, video text, graphics, and mouse programming.


17. Floating-Point Processing and Instruction Encoding: Floating-point binary representation and floating-point arithmetic. Learning to program the IA-32 Floating-Point Unit. Understanding the encoding of IA-32 machine instructions.

Appendix A: MASM Reference
Appendix B: The IA-32 Instruction Set
Appendix C: BIOS and MS-DOS Interrupts
Appendix D: Answers to Review Questions

Reference Materials


Help File  Help file (in Windows Help Format) by Gerald Cahill of Antelope Valley College. Documents the book’s link libraries, as well as Win32 data structures.

Assembly Language Workbook  An interactive workbook is included on the book’s Web site, covering such important topics as number conversions, addressing modes, register usage, Debug programming, and floating-point binary numbers. The content pages are HTML documents, making it easy for students and instructors to add their own customized content. This workbook is also available on the book’s Web site.


BIOS and MS-DOS Interrupts  Appendix C contains a brief listing of the most-often-used INT 10h (video), INT 16h (keyboard), and INT 21h (MS-DOS) functions.

Instruction Set  Appendix B lists most nonprivileged instructions for the IA-32 processor family.
For each instruction, we describe its effect, show its syntax, and show which flags are affected.

**PowerPoint Presentations**  
A complete set of Microsoft PowerPoint presentations written by the author.

**Acknowledgments**

Special thanks are due to Tracy Dunkleberger, Executive Editor for Computer Science at Prentice Hall, who provided friendly, helpful guidance during the writing of the fifth edition. Karen Ettinger did a terrific job as production editor, constantly keeping track of numerous minute details. Camille Trentacoste was the book’s managing editor.

**Fifth Edition**

I offer my special thanks and gratitude to the following professors who boosted my morale, gave me great pedagogical tips, and tirelessly examined the entire book. They have been a huge influence on the development of this book, in some cases across multiple editions:

- **Gerald Cahill**, Antelope Valley College
- **James Brink**, Pacific Lutheran University
- **William Barrett**, San Jose State University

Many thanks to Scott Blackledge and John Taylor, both professional programmers, who proofread most of the manuscript and flagged numerous errors. Several people reviewed individual chapters:

- Jerry Joyce, Keene State College
- Tianzheng Wu, Mount Mercy College
- Ron Davis, Kennedy-King College
- David Topham, Ohlone College
- Harvey Nice, DePaul University

**Fourth Edition**

The following people were tremendously helpful in creating the fourth edition:

- Gerald Cahill, Antelope Valley College
- James Brink, Pacific Lutheran University
- Maria Kolatis, County College of Morris
- Tom Joyce, Chief Engineer at Premier Heart, LLC
- Jeff Wothke, Purdue Calumet University
- Tim Downey, Florida International University

The following individuals provided valuable proofreading help in the fourth edition:

- Andres Altamirano, Miami
- Courtney Amor, Los Angeles
- Scott Blackledge, Platform Solutions, Inc.
- Ronald Davis, Kennedy-King College
- Ata Elahi, Southern Connecticut State University
- Jose Gonzalez, Miami
- Leroy Highsmith, Southern Connecticut State University
- Sajid Iqbal, Faran Institute of Technology
- Charles Jones, Maryville College
- Vincent Kayes, Mount St. Mary College
- Eric Kobrin, Miami
- Pablo Maurin, Miami
- Barry Meaker, Design Engineer, Boeing Corporation
- Ian Merkel, Miami
- Sylvia Miner, Miami
• M. Nawaz, OPSTEC College of Computer Science
• Kam Ng, Chinese University of Hong Kong
• Hien Nguyen, Miami
• Ernie Philipp, Northern Virginia Community College
• Boyd Stephens, UGMO Research, LLC
• John Taylor, England
• Zachary Taylor, Columbia College
• Virginia Welsh, Community College of Baltimore County
• Robert Workman, Southern Connecticut State University
• Tianzheng Wu, Mount Mercy College
• Matthew Zukoski, Lehigh University