



Contents

Introduction: “Why Would You Want to Do <i>That?</i>”	xxvii
Chapter 1 Another Pleasant Valley Saturday	1
It’s All in the Plan	1
Steps and Tests	2
More Than Two Ways?	3
Computers Think Like Us	4
Had This Been the Real Thing . . .	4
Do Not Pass Go	5
The Game of Big Bux	6
Playing Big Bux	8
Assembly Language Programming As a Board Game	9
Code and Data	10
Addresses	11
Metaphor Check!	12
Chapter 2 Alien Bases	15
The Return of the New Math Monster	15
Counting in Martian	16
Dissecting a Martian Number	18
The Essence of a Number Base	20
Octal: How the Grinch Stole Eight and Nine	20
Who Stole Eight and Nine?	21
Hexadecimal: Solving the Digit Shortage	24
From Hex to Decimal and from Decimal to Hex	28
From Hex to Decimal	28
From Decimal to Hex	29
Practice. Practice! PRACTICE!	31

Arithmetic in Hex	32
Columns and Carries	35
Subtraction and Borrows	35
Borrows across Multiple Columns	37
What's the Point?	38
Binary	38
Values in Binary	40
Why Binary?	42
Hexadecimal As Shorthand for Binary	43
Prepare to Compute	44
Chapter 3 Lifting the Hood	45
RAXie, We Hardly Knew Ye . . .	45
Gus to the Rescue	46
Switches, Transistors, and Memory	47
One If by Land . . .	48
Transistor Switches	48
The Incredible Shrinking Bit	50
Random Access	52
Memory Access Time	53
Bytes, Words, Double Words, and Quad Words	54
Pretty Chips All in a Row	55
The Shop Foreman and the Assembly Line	57
Talking to Memory	58
Riding the Data Bus	59
The Foreman's Pockets	60
The Assembly Line	61
The Box That Follows a Plan	61
Fetch and Execute	63
The Foreman's Innards	64
Changing Course	65
What vs. How: Architecture and Microarchitecture	66
Evolving Architectures	67
The Secret Machinery in the Basement	68
Enter the Plant Manager	70
Operating Systems: The Corner Office	70
BIOS: Software, Just Not as Soft	71
Multitasking Magic	71
Promotion to Kernel	73
The Core Explosion	73
The Plan	74

Chapter 4	Location, Location, Location	77
	The Joy of Memory Models	77
	16 Bits'll Buy You 64K	79
	The Nature of a Megabyte	82
	Backward Compatibility and Virtual 86 Mode	83
	16-Bit Blinders	83
	The Nature of Segments	85
	A Horizon, Not a Place	88
	Making 20-Bit Addresses out of 16-Bit Registers	88
	16-Bit and 32-Bit Registers	90
	General-Purpose Registers	91
	Register Halves	93
	The Instruction Pointer	95
	The Flags Register	96
	The Three Major Assembly Programming Models	96
	Real Mode Flat Model	97
	Real Mode Segmented Model	99
	Protected Mode Flat Model	101
	What Protected Mode Won't Let Us Do Anymore	104
	Memory-Mapped Video	104
	Direct Access to Port Hardware	105
	Direct Calls into the BIOS	106
	Looking Ahead: 64-Bit "Long Mode"	106
	64-Bit Memory: What May Be Possible Someday vs. What We Can Do Now	107
 Chapter 5	 The Right to Assemble	 109
	Files and What's Inside Them	110
	Binary Files vs. Text Files	111
	Looking at File Internals with the Bless Editor	112
	Interpreting Raw Data	116
	"Endianness"	117
	Text In, Code Out	121
	Assembly Language	121
	Comments	124
	Beware "Write-Only" Source Code!	124
	Object Code and Linkers	125
	Relocatability	128
	The Assembly Language Development Process	128
	The Discipline of Working Directories	129
	Editing the Source Code File	131

Assembling the Source Code File	131
Assembler Errors	132
Back to the Editor	133
Assembler Warnings	134
Linking the Object Code File	135
Linker Errors	136
Testing the .EXE File	136
Errors versus Bugs	137
Are We There Yet?	138
Debuggers and Debugging	138
Taking a Trip Down Assembly Lane	139
Installing the Software	139
Step 1: Edit the Program in an Editor	142
Step 2: Assemble the Program with NASM	143
Step 3: Link the Program with LD	146
Step 4: Test the Executable File	147
Step 5: Watch It Run in the Debugger	147
Ready to Get Serious?	153
Chapter 6 A Place to Stand, with Access to Tools	155
The Kate Editor	157
Installing Kate	157
Launching Kate	158
Configuration	160
Kate Sessions	162
Creating a New Session	162
Opening an Existing Session	163
Deleting or Renaming Sessions	163
Kate's File Management	164
Filesystem Browser Navigation	165
Adding a File to the Current Session	165
Dropping a File from the Current Session	166
Switching Between Session Files in the Editor	166
Creating a Brand-New File	166
Creating a Brand-New Folder on Disk	166
Deleting a File from Disk (Move File to Trash)	166
Reloading a File from Disk	167
Saving All Unsaved Changes in Session Files	167
Printing the File in the Editor Window	167
Exporting a File As HTML	167
Adding Items to the Toolbar	167
Kate's Editing Controls	168
Cursor Movement	169
Bookmarks	169
Selecting Text	170

Searching the Text	171
Using Search and Replace	172
Using Kate While Programming	172
Creating and Using Project Directories	173
Focus!	175
Linux and Terminals	176
The Linux Console	176
Character Encoding in Konsole	177
The Three Standard Unix Files	178
I/O Redirection	180
Simple Text Filters	182
Terminal Control with Escape Sequences	183
So Why Not GUI Apps?	185
Using Linux Make	186
Dependencies	187
When a File Is Up to Date	189
Chains of Dependencies	189
Invoking Make from Inside Kate	191
Using Touch to Force a Build	193
The Insight Debugger	194
Running Insight	195
Insight's Many Windows	195
A Quick Insight Run-Through	197
Pick Up Your Tools . . .	200
Chapter 7 Following Your Instructions	201
Build Yourself a Sandbox	201
A Minimal NASM Program	202
Instructions and Their Operands	204
Source and Destination Operands	204
Immediate Data	205
Register Data	207
Memory Data	209
Confusing Data and Its Address	210
The Size of Memory Data	211
The Bad Old Days	211
Rally Round the Flags, Boys!	212
Flag Etiquette	215
Adding and Subtracting One with INC and DEC	215
Watching Flags from Insight	216
How Flags Change Program Execution	218
Signed and Unsigned Values	221
Two's Complement and NEG	221
Sign Extension and MOVSX	224

Implicit Operands and MUL	225
MUL and the Carry Flag	227
Unsigned Division with DIV	228
The x86 Slowpokes	229
Reading and Using an Assembly Language Reference	230
Memory Joggers for Complex Memories	230
An Assembly Language Reference for Beginners	231
Flags	232
NEG: Negate (Two's Complement; i.e., Multiply by -1)	233
Flags affected	233
Legal forms	233
Examples	233
Notes	233
Legal Forms	234
Operand Symbols	234
Examples	235
Notes	235
What's Not Here . . .	235

Chapter 8 Our Object All Sublime 237

The Bones of an Assembly Language Program	237
The Initial Comment Block	239
The .data Section	240
The .bss Section	240
The .text Section	241
Labels	241
Variables for Initialized Data	242
String Variables	242
Deriving String Length with EQU and \$	244
Last In, First Out via the Stack	246
Five Hundred Plates per Hour	246
Stacking Things Upside Down	248
Push-y Instructions	249
POP Goes the Opcode	251
Storage for the Short Term	253
Using Linux Kernel Services Through INT80	254
An Interrupt That Doesn't Interrupt Anything	254
Getting Home Again	259
Exiting a Program via INT 80h	260
Software Interrupts versus Hardware Interrupts	261
INT 80h and the Portability Fetish	262
Designing a Non-Trivial Program	264
Defining the Problem	264
Starting with Pseudo-code	265

Successive Refinement	266
Those Inevitable “Whoops!” Moments	270
Scanning a Buffer	271
“Off By One” Errors	273
Going Further	277
Chapter 9 Bits, Flags, Branches, and Tables	279
Bits Is Bits (and Bytes Is Bits)	279
Bit Numbering	280
“It’s the Logical Thing to Do, Jim. . .”	280
The AND Instruction	281
Masking Out Bits	282
The OR Instruction	283
The XOR Instruction	284
The NOT Instruction	285
Segment Registers Don’t Respond to Logic!	285
Shifting Bits	286
Shift By What?	286
How Bit Shifting Works	287
Bumping Bits into the Carry Flag	287
The Rotate Instructions	288
Setting a Known Value into the Carry Flag	289
Bit-Bashing in Action	289
Splitting a Byte into Two Nybbles	292
Shifting the High Nybble into the Low Nybble	293
Using a Lookup Table	293
Multiplying by Shifting and Adding	295
Flags, Tests, and Branches	298
Unconditional Jumps	298
Conditional Jumps	299
Jumping on the Absence of a Condition	300
Flags	301
Comparisons with CMP	301
A Jungle of Jump Instructions	302
“Greater Than” Versus “Above”	303
Looking for 1-Bits with TEST	304
Looking for 0 Bits with BT	306
Protected Mode Memory Addressing in Detail	307
Effective Address Calculations	308
Displacements	309
Base + Displacement Addressing	310
Base + Index Addressing	310
Index × Scale + Displacement Addressing	312
Other Addressing Schemes	313

LEA: The Top-Secret Math Machine	315
The Burden of 16-Bit Registers	317
Character Table Translation	318
Translation Tables	318
Translating with MOV or XLAT	320
Tables Instead of Calculations	325
Chapter 10 Dividing and Conquering	327
Boxes within Boxes	328
Procedures As Boxes for Code	329
Calling and Returning	336
Calls within Calls	338
The Dangers of Accidental Recursion	340
A Flag Etiquette Bug to Beware Of	341
Procedures and the Data They Need	342
Saving the Caller's Registers	343
Local Data	346
More Table Tricks	347
Placing Constant Data in Procedure Definitions	349
Local Labels and the Lengths of Jumps	350
"Forcing" Local Label Access	353
Short, Near, and Far Jumps	354
Building External Procedure Libraries	355
Global and External Declarations	356
The Mechanics of Globals and Externals	357
Linking Libraries into Your Programs	365
The Dangers of Too Many Procedures and Too Many Libraries	366
The Art of Crafting Procedures	367
Maintainability and Reuse	367
Deciding What Should Be a Procedure	368
Use Comment Headers!	370
Simple Cursor Control in the Linux Console	371
Console Control Cautions	377
Creating and Using Macros	378
The Mechanics of Macro Definition	379
Defining Macros with Parameters	385
The Mechanics of Invoking Macros	386
Local Labels Within Macros	387
Macro Libraries As Include Files	388
Macros versus Procedures: Pros and Cons	389

Chapter 11	Strings and Things	393
	The Notion of an Assembly Language String	393
	Turning Your “String Sense” Inside-Out	394
	Source Strings and Destination Strings	395
	A Text Display Virtual Screen	395
	REP STOSB, the Software Machine Gun	402
	Machine-Gunning the Virtual Display	403
	Executing the STOSB Instruction	404
	STOSB and the Direction Flag (DF)	405
	Defining Lines in the Display Buffer	406
	Sending the Buffer to the Linux Console	406
	The Semiautomatic Weapon: STOSB without REP	407
	Who Decrements ECX?	407
	The LOOP Instructions	408
	Displaying a Ruler on the Screen	409
	MUL Is Not IMUL	410
	Adding ASCII Digits	411
	Adjusting AAA’s Adjustments	413
	Ruler’s Lessons	414
	16-bit and 32-bit Versions of STOS	414
	MOVSB: Fast Block Copies	414
	DF and Overlapping Block Moves	416
	Single-Stepping REP String Instructions with Insight	418
	Storing Data to Discontinuous Strings	419
	Displaying an ASCII Table	419
	Nested Instruction Loops	420
	Jumping When ECX Goes to 0	421
	Closing the Inner Loop	421
	Closing the Outer Loop	422
	Showchar Recap	423
	Command-Line Arguments and Examining the Stack	424
	Virtual Memory in Two Chunks	424
	Anatomy of the Linux Stack	427
	Why Stack Addresses Aren’t Predictable	429
	Setting Command-Line Arguments with Insight	429
	Examining the Stack with Insight’s Memory View	430
	String Searches with SCASB	432
	REPNE vs. REPE	435
	Pop the Stack or Address It?	436
	For Extra Credit . . .	438

Chapter 12 Heading Out to C	439
What's GNU?	440
The Swiss Army Compiler	441
Building Code the GNU Way	441
How to Use gcc in Assembly Work	443
Why Not gas?	444
Linking to the Standard C Library	445
C Calling Conventions	446
A Framework to Build On	447
Saving and Restoring Registers	447
Setting Up a Stack Frame	448
Destroying a Stack Frame	450
Characters Out via puts()	451
Formatted Text Output with printf()	452
Passing Parameters to printf()	454
Data In with fgets() and scanf()	456
Using scanf() for Entry of Numeric Values	458
Be a Time Lord	462
The C Library's Time Machine	462
Fetching time_t Values from the System Clock	464
Converting a time_t Value to a Formatted String	464
Generating Separate Local Time Values	465
Making a Copy of glibc's tm Struct with MOVSD	466
Understanding AT&T Instruction Mnemonics	470
AT&T Mnemonic Conventions	470
Examining gas Source Files Created by gcc	471
AT&T Memory Reference Syntax	474
Generating Random Numbers	475
Seeding the Generator with srand()	476
Generating Pseudorandom Numbers	477
Some Bits Are More Random Than Others	482
Calls to Addresses in Registers	483
How C Sees Command-Line Arguments	484
Simple File I/O	487
Converting Strings into Numbers with sscanf()	487
Creating and Opening Files	489
Reading Text from Files with fgets()	490
Writing Text to Files with fprintf()	493
Notes on Gathering Your Procedures into Libraries	494
Conclusion: Not the End, But Only the Beginning	503
Where to Now?	504
Stepping off Square One	506

Appendix A Partial x86 Instruction Set Reference	507
Notes on the Instruction Set Reference	510
AAA: Adjust AL after BCD Addition	512
ADC: Arithmetic Addition with Carry	513
ADD: Arithmetic Addition	515
AND: Logical AND	517
BT: Bit Test	519
CALL: Call Procedure	521
CLC: Clear Carry Flag (CF)	523
CLD: Clear Direction Flag (DF)	524
CMP: Arithmetic Comparison	525
DEC: Decrement Operand	527
DIV: Unsigned Integer Division	528
INC: Increment Operand	529
INT: Software Interrupt	530
IRET: Return from Interrupt	531
J?: Jump on Condition	532
JCZ: Jump If CX=0	534
JECZ: Jump If ECX=0	535
JMP: Unconditional Jump	536
LEA: Load Effective Address	537
LOOP: Loop until CX/ECX=0	538
LOOPNZ/LOOPNE: Loop While CX/ECX > 0 and ZF=0	540
LOOPZ/LOOPE: Loop While CX/ECX > 0 and ZF=1	541
MOV: Move (Copy) Right Operand into Left Operand	542
MOVS: Move String	544
MOVSX: Move (Copy) with Sign Extension	546
MUL: Unsigned Integer Multiplication	547
NEG: Negate (Two's Complement; i.e., Multiply by -1)	549
NOP: No Operation	550
NOT: Logical NOT (One's Complement)	551
OR: Logical OR	552
POP: Pop Top of Stack into Operand	554
POPА/POPAD: Pop All GP Registers	555
POPF: Pop Top of Stack into 16-Bit Flags	556
POPFD: Pop Top of Stack into EFlags	557
PUSH: Push Operand onto Top of Stack	558
PUSHA: Push All 16-Bit GP Registers	559
PUSHAD: Push All 32-Bit GP Registers	560
PUSHF: Push 16-Bit Flags onto Stack	561
PUSHFD: Push 32-Bit EFlags onto Stack	562
RET: Return from Procedure	563
ROL: Rotate Left	564

ROR: Rotate Right	566
SBB: Arithmetic Subtraction with Borrow	568
SHL: Shift Left	570
SHR: Shift Right	572
STC: Set Carry Flag (CF)	574
STD: Set Direction Flag (DF)	575
STOS: Store String	576
SUB: Arithmetic Subtraction	577
XCHG: Exchange Operands	579
XLAT: Translate Byte via Table	580
XOR: Exclusive Or	581
Appendix B Character Set Charts	583
Index	587