



Professional

Assembly Language

Richard Blum



Updates, source code, and Wrox technical support at www.wrox.com

Contents

Acknowledgments	xi
Contents	xiii
Introduction	xxiii
Chapter 1: What Is Assembly Language?	1
Processor Instructions	1
Instruction code handling	2
Instruction code format	3
High-Level Languages	6
Types of high-level languages	7
High-level language features	9
Assembly Language	10
Opcode mnemonics	11
Defining data	12
Directives	14
Summary	15
Chapter 2: The IA-32 Platform	17
Core Parts of an IA-32 Processor	17
Control unit	19
Execution unit	24
Registers	25
Flags	29
Advanced IA-32 Features	32
The x87 floating-point unit	32
Multimedia extensions (MMX)	33
Streaming SIMD extensions (SSE)	33
Hyperthreading	34
The IA-32 Processor Family	34
Intel processors	35
Non-Intel processors	36
Summary	37

Chapter 3: The Tools of the Trade	39
The Development Tools	39
The Assembler	40
The Linker	42
The Debugger	43
The Compiler	44
The object code disassembler	44
The Profiler	44
The GNU Assembler	45
Installing the assembler	45
Using the assembler	47
A word about opcode syntax	49
The GNU Linker	50
The GNU Compiler	53
Downloading and installing gcc	53
Using gcc	54
The GNU Debugger Program	56
Downloading and installing gdb	56
Using gdb	57
The KDE Debugger	60
Downloading and installing kdbg	60
Using kdbg	60
The GNU Objdump Program	62
Using objdump	63
An objdump example	64
The GNU Profiler Program	65
Using the profiler	65
A profile example	68
A Complete Assembly Development System	69
The basics of Linux	69
Downloading and running MEPIS	70
Your new development system	71
Summary	72
Chapter 4: A Sample Assembly Language Program	73
The Parts of a Program	73
Defining sections	74
Defining the starting point	74
Creating a Simple Program	75
The CPUID instruction	76
The sample program	77

Building the executable	80
Running the executable	80
Assembling using a compiler	80
Debugging the Program	81
Using gdb	81
Using C Library Functions in Assembly	86
Using printf	87
Linking with C library functions	88
Summary	90
 Chapter 5: Moving Data	 91
Defining Data Elements	91
The data section	91
Defining static symbols	94
The bss section	95
Moving Data Elements	97
The MOV instruction formats	97
Moving immediate data to registers and memory	98
Moving data between registers	99
Moving data between memory and registers	99
Conditional Move Instructions	106
The CMOV instructions	107
Using CMOV instructions	109
Exchanging Data	110
The data exchange instructions	111
Using the data exchange instruction	116
The Stack	119
How the stack works	119
PUSHing and POPing data	120
PUSHing and POPing all the registers	123
Manually using the ESP and EBP registers	123
Optimizing Memory Access	123
Summary	124
 Chapter 6: Controlling Execution Flow	 127
The Instruction Pointer	127
Unconditional Branches	129
Jumps	129
Calls	132
Interrupts	135

Contents

Conditional Branches	136
Conditional jump instructions	136
The compare instruction	138
Examples of using the flag bits	140
Loops	144
The loop instructions	144
A loop example	145
Preventing LOOP catastrophes	145
Duplicating High-Level Conditional Branches	146
if statements	147
for loops	150
Optimizing Branch Instructions	153
Branch prediction	153
Optimizing tips	155
Summary	158
 Chapter 7: Using Numbers	 161
 Numeric Data Types	 161
Integers	162
Standard integer sizes	162
Unsigned integers	164
Signed integers	166
Using signed integers	168
Extending integers	169
Defining integers in GAS	172
SIMD Integers	173
MMX integers	173
Moving MMX integers	174
SSE integers	176
Moving SSE integers	177
Binary Coded Decimal	178
What is BCD?	178
FPU BCD values	179
Moving BCD values	180
Floating-Point Numbers	182
What are floating-point numbers?	182
Standard floating-point data types	184
IA-32 floating-point values	186
Defining floating-point values in GAS	187
Moving floating-point values	187
Using preset floating-point values	189

SSE floating-point data types	190
Moving SSE floating-point values	191
Conversions	196
Conversion instructions	196
A conversion example	198
Summary	199
 Chapter 8: Basic Math Functions	 201
Integer Arithmetic	201
Addition	201
Subtraction	210
Incrementing and decrementing	215
Multiplication	216
Division	221
Shift Instructions	223
Multiply by shifting	224
Dividing by shifting	225
Rotating bits	226
Decimal Arithmetic	227
Unpacked BCD arithmetic	227
Packed BCD arithmetic	229
Logical Operations	231
Boolean logic	231
Bit testing	232
Summary	233
 Chapter 9: Advanced Math Functions	 235
The FPU Environment	235
The FPU register stack	236
The FPU status, control, and tag registers	237
Using the FPU stack	242
Basic Floating-Point Math	245
Advanced Floating-Point Math	249
Floating-point functions	249
Partial remainders	252
Trigonometric functions	254
Logarithmic functions	257
Floating-Point Conditional Branches	259
The FCOM instruction family	260
The FCOMI instruction family	262
The FCMOV instruction family	263

Contents

Saving and Restoring the FPU State	265
Saving and restoring the FPU environment	265
Saving and restoring the FPU state	266
Waiting versus Nonwaiting Instructions	269
Optimizing Floating-Point Calculations	270
Summary	270
Chapter 10: Working with Strings	273
Moving Strings	273
The MOVS instruction	274
The REP prefix	278
Other REP instructions	283
Storing and Loading Strings	283
The LODS instruction	283
The STOS instruction	284
Building your own string functions	285
Comparing Strings	286
The CMPS instruction	286
Using REP with CMPS	288
String inequality	289
Scanning Strings	291
The SCAS instruction	292
Scanning for multiple characters	293
Finding a string length	295
Summary	296
Chapter 11: Using Functions	297
Defining Functions	297
Assembly Functions	299
Writing functions	299
Accessing functions	302
Function placement	304
Using registers	304
Using global data	304
Passing Data Values in C Style	306
Revisiting the stack	306
Passing function parameters on the stack	306
Function prologue and epilogue	308
Defining local function data	309

Cleaning out the stack	312
An example	312
Watching the stack in action	314
Using Separate Function Files	317
Creating a separate function file	317
Creating the executable file	318
Debugging separate function files	319
Using Command-Line Parameters	320
The anatomy of a program	320
Analyzing the stack	321
Viewing command-line parameters	323
Viewing environment variables	325
An example using command-line parameters	326
Summary	328
 Chapter 12: Using Linux System Calls	 329
 The Linux Kernel	 329
Parts of the kernel	330
Linux kernel version	336
System Calls	337
Finding system calls	337
Finding system call definitions	338
Common system calls	339
Using System Calls	341
The system call format	341
Advanced System Call Return Values	346
The sysinfo system call	346
Using the return structure	347
Viewing the results	348
Tracing System Calls	349
The strace program	349
Advanced strace parameters	350
Watching program system calls	351
Attaching to a running program	353
System Calls versus C Libraries	355
The C libraries	356
Tracing C functions	357
Comparing system calls and C libraries	358
Summary	359

Contents

Chapter 13: Using Inline Assembly	361
What Is Inline Assembly?	361
Basic Inline Assembly Code	365
The asm format	365
Using global C variables	367
Using the volatile modifier	369
Using an alternate keyword	369
Extended ASM	370
Extended ASM format	370
Specifying input and output values	370
Using registers	372
Using placeholders	373
Referencing placeholders	376
Alternative placeholders	377
Changed registers list	377
Using memory locations	379
Using floating-point values	380
Handling jumps	382
Using Inline Assembly Code	384
What are macros?	384
C macro functions	384
Creating inline assembly macro functions	386
Summary	387
Chapter 14: Calling Assembly Libraries	389
Creating Assembly Functions	389
Compiling the C and Assembly Programs	391
Compiling assembly source code files	392
Using assembly object code files	392
The executable file	393
Using Assembly Functions in C Programs	395
Using integer return values	396
Using string return values	397
Using floating-point return values	400
Using multiple input values	401
Using mixed data type input values	403
Using Assembly Functions in C++ Programs	407
Creating Static Libraries	408
What is a static library?	408
The ar command	409

Creating a static library file	410
Compiling with static libraries	412
Using Shared Libraries	412
What are shared libraries?	412
Creating a shared library	414
Compiling with a shared library	414
Running programs that use shared libraries	415
Debugging Assembly Functions	417
Debugging C programs	417
Debugging assembly functions	418
Summary	420
 Chapter 15: Optimizing Routines	 421
 Optimized Compiler Code	 421
Compiler optimization level 1	422
Compiler optimization level 2	423
Compiler optimization level 3	425
Creating Optimized Code	425
Generating the assembly language code	425
Viewing optimized code	429
Recompiling the optimized code	429
Optimization Tricks	430
Optimizing calculations	430
Optimizing variables	433
Optimizing loops	437
Optimizing conditional branches	442
Common subexpression elimination	447
Summary	450
 Chapter 16: Using Files	 453
 The File-Handling Sequence	 453
Opening and Closing Files	454
Access types	455
UNIX permissions	456
Open file code	458
Open error return codes	459
Closing files	460
Writing to Files	460
A simple write example	460
Changing file access modes	462
Handling file errors	462

Contents

Reading Files	463
A simple read example	464
A more complicated read example	465
Reading, Processing, and Writing Data	467
Memory-Mapped Files	470
What are memory-mapped files?	470
The mmap system call	471
mmap assembly language format	473
An mmap example	475
Summary	479
 Chapter 17: Using Advanced IA-32 Features	 481
A Brief Review of SIMD	481
MMX	482
SSE	483
SSE2	483
Detecting Supported SIMD Operations	483
Detecting support	484
SIMD feature program	485
Using MMX Instructions	487
Loading and retrieving packed integer values	487
Performing MMX operations	488
Using SSE Instructions	497
Moving data	498
Processing data	499
Using SSE2 Instructions	504
Moving data	505
Processing data	505
SSE3 Instructions	508
Summary	508
 Index	 511