

# Index

## Symbols

- . (period), in directive syntax, 14
- \$ (dollar sign), opcode syntax, 49
- % (percent sign), opcode syntax, 49
- 64-bit packed... data types, 162, 174
- 128-bit packed... data types, 162, 176, 191

## A

- AAA instruction, 227–229
- AAD instruction, 227–229
- AAM instruction, 227–229
- AAS instruction, 227–229
- absolute values, floating-point, 249–252
- AC (alignment check flag), 31
- access types, files, 455–456
- ADC instruction, 206–210
- ADD instruction, 202–204
- ADDB instruction, 204
- addition
  - carry detection, 204–206
  - floating-point, 245–248
  - integer values, 202–204
  - large integers, 206–210
  - MMX (Multimedia Extension), 488–491
  - overflow detection, 204–206
  - SSE2, 505–508
- ADDPD instruction, 506–508
- ADDPB instruction, 499–501
- address displacement byte, 5–6
- address size, 4–5
- addressing-form specifier (ModR/M) byte, 5
- ADDSD instruction, 506–508
- ADDSUBPD instruction, 508
- ADDSUBPS instruction, 508
- ADDW instruction, 204
- adjust flag (AF), 30
- AF (adjust flag), 30
- alignment check flag (AC), 31
- allocator, 22–24
- ALUs (Arithmetic Logical Units), 24–25
- AMD processors, 36
- AND operator, 231–232
- ANDNPS instruction, 499–501
- ANDPS instruction, 499–501
- ar command, 409–410
- arithmetic. *See also specific mathematical operations*
  - BCD
    - carry and borrow operations, 30
    - description, 178–179
    - FPU BCD values, 179–180
    - moving BCD values, 180–182
  - decimal
    - packed BCD, 229–231
    - unpacked BCD, 227–229
  - floating-point
    - absolute values, 249–252
    - addition, 245–248
    - binary fractions, 183–184
    - binary scientific notation, 183–184
    - conditional branches, 259–265
    - conditional data moves, 263–265

## arithmetic (continued)

- cosines, 254–257
- data types, 33, 184–186
- denormalized coefficients, 185
- description, 182–183, 185–186
- division, 245–248
- double floating-point values in `gas`, 187
- double-extended-precision, 186–187
- double-precision, 194–196
- exceptions, 269
- format, 183
- IA-32 floating-point values, 186–187
- IEEE Standard 754 floating-point formats, 184
- log base 2 calculations, 257–259
- logarithmic functions, 257–259
- moving floating-point values, 187–189
- multiplication, 245–248
- nonwaiting instructions, 269
- normalized coefficients, 185
- optimizing, 270
- partial remainders, 252–254
- preset floating-point values, 189–190
- real numbers, 182–183
- return values, 400–401
- rounding, 249–252
- sign changes, 249–252
- sines, 254–256
- single-precision, 185–186
- square roots, 249–252
- SSE data types, 190–193
- SSE2 data types, 191–193
- SSE3 instructions, 196
- subtraction, 245–248
- trigonometric functions, 254–257
- values, `asm` format, 380–382
- waiting instructions, 269
- logical operations
  - bit testing, 232–233
  - Boolean logic, 231–232
  - MMX, 493–494
- mathematical operations
  - BCD (Binary Coded Decimal), 30
  - borrow indicator, 30
  - carry indicator, 30
  - data corruption indicator, 30
  - overflow indicator, 30
  - parity indicator, 30
  - sign indicator, 30, 141–142

## arithmetic, integer

- addition
  - carry detection, 204–206
  - integer values, 202–204
  - large integers, 206–210
  - overflow detection, 204–206
- byte integers, 162
- decrementing, 215–216
- defining integers in `gas` (GNU assembler), 172–173
- division *See also* rotating bits; shift instructions
  - error detection, 223
  - signed integers, 222–223
  - unsigned integers, 221–222
- doubleword integers, 162
- extending integers, 169–172
- incrementing, 215–216
- little-endian *versus* big-endian, 162
- multiplication *See also* rotating bits; shift instructions
  - overflow detection, 220
  - signed integers, 218–220
  - unsigned integers, 216–218
- one's complement, 167
- quadword integers, 162
- return values, 306–397
- shift instructions *See also* division; multiplication
  - description, 223
  - division, 225–226, 226–227
  - MMX (Multimedia Extension), 493–494
  - multiplication, 224–225, 226–227
  - rotating bits, 226–227
- signed integers
  - description, 166–168
  - extending, 170–172
- signed magnitude, 166
- SSE instructions, 504
- standard sizes, 162
- subtraction
  - carry detection, 212–214
  - integer values, 210–212
  - large integers, 214–216
  - overflow detection, 212–214
- two's complement, 167
- unsigned integers
  - description, 164–165
  - extending, 169–170
- word integers, 162

**arithmetic, SIMD integers**

- description, 173
- MMX (Multimedia Extension)
  - addition and subtraction, 488–491
  - Boolean instructions, 493–494
  - comparison instructions, 494–496
  - description, 173–174, 482
  - instructions, 490
  - logical instructions, 493–494
  - moving, 174–176
  - multiplication, 491–493
  - overflow, 489–491
  - packed values, loading and retrieving, 487–488
  - shift instructions, 493–494
- SSE (Streaming SIMD Extension) instructions
  - arithmetic, 499–501
  - comparisons, 501–504
  - description, 497
  - integers, 504
  - moving data, 498–499
  - processing data, 499–504
- SSE (Streaming SIMD Extension) integers
  - description, 176, 483
  - instructions, 504
  - moving, 177–178
- SSE2 data types
  - description, 483
  - moving, 191–193
- SSE2 instructions
  - addition, 505–508
  - description, 504–508
  - moving data, 505
  - processing data, 505–508
- SSE3 instructions, 196, 508

**Arithmetic Logical Units (ALUs), 24–25**as **command**, 47**.ascii directive**, 92**.asciz directive**, 92asm **format**

- altering, 369
- description, 365–367
- extending
  - changed registers list, 377–379
  - constraints, 371
  - extended format, 370
  - floating-point values, 380–382
  - input/output values, specifying, 370–372

- jumps, 382–383
- memory locations, 379–380
- output modifiers, 371
- place holders, alternative, 377
- place holders, description, 373–376
- place holders, referencing, 376
- registers, 372–373

**assemblers**

- choosing, 41–42
- description, 40
- differences between, 40
- gas (GNU assembler)
  - defining integers, 172–173
  - description, 41–42
  - installing, 45–47
  - invoking, 47
  - opcode syntax, 49–50
  - platforms supporting, 45
  - using, 47–49
- HLA (High Level Assembler), 42
- MASM (Microsoft Assembler), 41
- NASM (Netwide Assembler), 41

**assembling, sample program, 80–81****assembly development system**

- Linux basics, 69–70
- MEPIS Linux distribution, 70–71
- using, 71–72

**assembly functions (assembly language programs)**

- calling, 302–303
- command-line parameters
  - analyzing the stack, 321–323
  - anatomy of a program, 320–321
  - environment variables, viewing, 325–326
  - example, 326–327
  - viewing, 323–325
- creating for
  - assembly language programs, 301–302
  - C or C++ programs, 389–391
- description, 297–299
- global variables, 304–305
- input values, defining, 300
- output values, defining, 301
- passing data, with the stack
  - cleaning the stack, 312
  - debugging the stack, 314–317
  - example, 312–314
  - function parameters, 306–308

# assembly functions (assembly language programs)

---

## assembly functions (assembly language programs)

### (continued)

- function prologue and epilogue, 308–309
  - local function data, 309–310
  - monitoring the stack, 314–317
  - placement, 304
  - processes, defining, 300
  - registers, 304
  - separate function files
    - creating, 317–318
    - debugging, 319–320
    - executable files, creating, 318–319
    - writing, 299–302
- assembly functions (C or C++ programs). See also assembly libraries; inline assembly**
- creating, 389–391
  - debugging
    - assembly functions, 418–419
    - C programs, 417–418
    - listing source code, 417–418
  - input values
    - mixed data type, 403–406
    - multiple, 401–403
    - ordering, 403–406
    - reading in order, 405–406
  - return values
    - floating-point, 400–401
    - integer, 306–397
    - string, 397–399
  - shared libraries
    - compiling with, 414–415
    - creating, 414
    - definition, 412–414
    - running programs, 415–417
  - static libraries
    - ar command, 409–410
    - compiling with, 412
    - creating, 410–411
    - definition, 408–409
  - using, 407–408
- assembly language**
- benefits of, 10
  - sample program
    - assembling, 80–81
    - basic template, 75
    - bss section, 73–74
    - code sample, 77–80
    - CPUID instruction, 76–77
    - creating, 76–81

- data section, 73–74
- debugging, 81–86
- executable, building and running, 80
- .global directive, 75
- linking, 88–89
- printf function, 87–88
- program parts, 73–74
- sections, 73–75
- \_start label, 75
- starting point, defining, 74–75
- text section, 73–74

## assembly libraries. See also inline assembly

- assembly object code files, 392–393
  - compiling source code files, 392
  - executable files, 393–395
  - sample program, C library functions, 86–89
- atof() function, 326–327**
- atoi() function, 326–327**
- atol() function, 326–327**

## B

### BCD (Binary Coded Decimal) arithmetic

- See also decimal arithmetic
- See also floating-point arithmetic
- See also integer arithmetic
- See also logical operations
- See also mathematical operations
- See also SIMD integers
- carry and borrow operations, 30
- description, 178–179
- FPU BCD values, 179–180
- moving BCD values, 180–182

### big-endian values

- definition, 6
- integer arithmetic, 162
- versus little-endian, 162
- swapping to little-endian, 112–113

### Binary Coded Decimal (BCD) arithmetic

- See also decimal arithmetic
- See also floating-point arithmetic
- See also integer arithmetic
- See also logical operations
- See also mathematical operations
- See also SIMD integers
- carry and borrow operations, 30
- description, 178–179
- FPU BCD values, 179–180
- moving BCD values, 180–182

**binary fractions, 183–184**

**binary scientific notation, 183–184**

`binutils` package, 45–47

**bit testing, 232–233**

**block device files, 332**

**Boolean instructions, MMX, 493–494**

**Boolean logic, 231–232**

**branch hint prefixes, 4**

**branch prediction**

deep branch prediction, 22

description, 22

dynamic data flow analysis, 22

speculative execution, 22

**branch prediction unit, 22**

**branches. See also execution flow**

conditional

comparing flag bits, 140–143

comparing values, 138–139

jumps, 136–138

predicting, 154–155

eliminating, 155–156

high-level, 146–152

optimizing

branch prediction, 153–155

coding predictable branches first, 156–157

conditional, 442–447

conditional branches, 154–155

eliminating branches, 155–156

functions for, 422–424

`if-then` code, 442–447

prediction, 423

unconditional branches, 153–154

unrolling loops, 157–158

predictable, coding first, 156–157

predicting, 153–155

unconditional

calls, 132–135

interrupts, 135–136

jumps, 129–132

predicting, 153–154

**breakpoints, 82–83**

**bss section**

defining, 74, 95–97

definition, 73

**BSWAP instruction, 112–113**

**bubble sort, 116–119**

**buffers, allocating, 15, 23**

**byte code, 9**

`.byte` directive, 92

**byte integers, 162**

**byte orientation, swapping, 112–113**

## C

**C library functions**

conversion utilities, 326–327

versus Linux kernel system calls, 355–359

sample program, 86–89

tracing, 357–358

**C macro functions, 384–385**

**C or C++ programs, assembly functions. See also assembly libraries; inline assembly**

creating, 389–391

debugging

assembly functions, 418–419

C programs, 417–418

listing source code, 417–418

input values

mixed data type, 403–406

multiple, 401–403

ordering, 403–406

reading in order, 405–406

return values

floating-point, 400–401

integer, 306–397

string, 397–399

shared libraries

compiling with, 414–415

creating, 414

definition, 412–414

running programs, 415–417

static libraries

`ar` command, 409–410

compiling with, 412

creating, 410–411

definition, 408–409

using, 407–408

**calculations, optimizing, 430–433**

**CALL instruction, 132–135**

**calls**

assembly language functions, 302–303, 424

system (Linux kernel)

versus C libraries, 355–359

call values, 342

common, 339–341

file system, 340

finding, 337–338

format, 341–346



- fcse-follow-jumps, 424
- fdefer-pop, 422
- fdelayed-branch, 423
- fdelete-null-pointer-checks, 424
- fexpensive-optimizations, 424
- fforce-mem, 423
- fgcse, 424
- fgcse-after-reload, 425
- fguess-branch-probability, 423
- fif-conversion, 423
- fif-conversion2, 423
- finline-functions, 425
- floop-optimize, 422
- fmerge-constants, 422
- foptimize-sibling-calls, 423
- fpeephole2, 424
- fregmove, 424
- freorder-blocks, 424
- frerun-cse-after-loop, 424
- fsched-interblock, 424
- fschedule-insns, 424
- fstrength-reduce, 423
- fstrict-aliasing, 424
- fthread-jumps, 422
- funit-at-a-time, 424
- fweb, 425

**code segment (CS) registers, 28**

COMISS instruction, 501–502

.comm directive, 95–97

**command-line parameters, functions**

- analyzing the stack, 321–323
- anatomy of a program, 320–321
- environment variables, viewing, 325–326
- example, 326–327
- viewing, 323–325

**commands**

- as, 47
- ar, 409–410
- l, 418
- next, 419
- print
  - debugging C programs, 418
  - register values, displaying, 85–86
- ps, 334–336
- x, 86

**Common Subexpression Elimination (cse)**

- example, 447–450
- fcse-follow-jumps function, 424
- fgcse function, 424

- fgcse-after-reload function, 425
- frerun-cse-after-loop function, 424

**comparisons**

- flag bits, 140–143
- MMX (Multimedia Extension), 494–496
- SSE (Streaming SIMD Extension) instructions, 501–504
- strings, 286–291
- values, 138–139

**compiled languages, 7–8****compilers**

- description, 44
- gcc (GNU compiler)
  - description, 53
  - downloading and installing, 53–54
  - example, 55–56, 80–81
  - invoking, 54
  - parameters, 54–56

**compiling**

- assembly functions with
  - shared libraries, 414–415
  - static libraries, 412
- assembly library source code files, 392
- programs, 7

**complex-integer operations, 24–25****conditional branches**

- comparing flag bits, 140–143
- comparing values, 138–139
- floating-point, 259–265
- jump, 136–138
- predicting, 154–155

**conditional data moves, floating-point, 263–265****conditional moves**

- CMOV . . . instructions, 107–110
- description, 106–107
- EFLAGS register, 107

**conditional processing**

- branches
  - eliminating, 155–156
  - high-level, 146–152
  - optimizing, 153–158
  - predictable, coding first, 156–157
  - predicting, 153–155
- branches, conditional
  - comparing flag bits, 140–143
  - comparing values, 138–139
  - jump, 136–138
  - predicting, 154–155

## conditional processing (continued)

- branches, unconditional
  - calls, 132–135
  - interrupts, 135–136
  - jumps, 129–132
  - predicting, 153–154
- conditional statements
  - high-level, duplicating, 146–152
  - if statements, 147–150
  - for statements, 150–152
- instruction pointers, 127–129
- loops
  - description, 144
  - error prevention, 145–146
  - example, 144–145
  - high-level, duplicating, 146–152
  - if statements, 147–150
  - instructions for, 144–145
  - for statements, 150–152
  - unrolling, 157–158
- next instruction, determining, 127–129
- conditional statements. See also execution flow**
  - high-level, duplicating, 146–152
  - if statements, 147–150
  - for statements, 150–152
- conditional statements, predicting, 4**
- constants, merging identical, 422**
- constraints, asm format, 371**
- control flags, 31**
- control register, 239–241**
- control unit, 19–24**
- controlling program execution. See execution flow**
- cosines, floating-point, 254–257**
- CPUID instruction, 76–77**
- creating**
  - assembly functions for
    - assembly language programs, 301–302
    - C or C++ programs, 389–391
  - sample program, 76–81
  - separate function files, 317–318
  - shared libraries, 414
  - static libraries, 410–411
- CRn registers, 29**
- cross-jumping, 425**
- CS (code segment) registers, 28**
- cse (Common Subexpression Elimination)**
  - example, 447–450
  - fcse-follow-jumps function, 424
  - fgcse function, 424

- fgcse-after-reload function, 425
- frerun-cse-after-loop function, 424
- CVTDQ2PD instruction, 197**
- CVTDQ2PS instruction, 197**
- CVTPD2DQ instruction, 197**
- CVTPD2PI instruction, 197**
- CVTPD2PS instruction, 197**
- CVTPI2PD instruction, 197**
- CVTPI2PS instruction, 197**
- CVTPS2DQ instruction, 197**
- CVTPS2PD instruction, 197**
- CVTPS2PI instruction, 197**
- CVTTPD2DQ instruction, 197**
- CVTTPD2PI instruction, 197**
- CVTTPS2DQ instruction, 197**
- CVTTPS2PI instruction, 197**
- Cyrix processors, 37**

## D

- DAA instruction, 229–231**
- DAS instruction, 229–231**
- data. See also directives; instructions**
  - comparisons
    - flag bits, 140–143
    - MMX (Multimedia Extension), 494–496
    - SSE (Streaming SIMD Extension) instructions, 501–504
    - strings, 286–291
    - values, 138–139
  - elements, defining
    - bss section, 95–97
    - data section, 91–94
    - static symbols, 94–95
  - storing and retrieving
    - with memory locations, 12–13
    - from the stack, 14
- data, moving**
  - conditional moves
    - CMOV... instructions, 107–110
    - description, 106–107
    - EFLAGS register, 107
  - exchanging data
    - data exchange instructions, 111–118
    - description, 110–111
  - immediate data, to registers and memory, 98–99
  - memory access, optimizing, 123
  - between memory and registers
    - indexed memory locations, 101–103
    - indirect addressing, 103–106

- memory to registers, 99–100
- pointers, 103
- registers to memory, 100–101
- MOV . . . instructions, 97–98
- between registers, 99
- SSE (Streaming SIMD Extension) instructions, 498–499
- SSE2, 505
- the stack
  - description, 119–120
  - ESP and EPB registers, 123
  - pushing and popping data, 120–122
  - pushing and popping registers, 123
- data codes, distinguishing from instruction codes, 2–3**
- .data **directive, 91–94**
- data elements**
  - defining
    - bss section, 95–97
    - data section, 91–94
    - static symbols, 94–95
  - definition, 6
- data exchange, 111–118**
- data pointers, 2**
- data section. See also variables**
  - defining, 74, 91–94
  - definition, 73
  - description, 91–94
  - read-only, 92
- data segment (DS) registers, 28**
- data segments, register pointers, 28**
- data types**
  - 64-bit packed..., 33
  - 128-bit packed..., 34
  - conversion example, 198–199
  - conversion instructions, 196–197
  - floating-point, 33
  - floating-point arithmetic, 184–186
  - numeric, 161–162
    - See also BCD (Binary Coded Decimal) arithmetic
    - See also floating-point arithmetic
    - See also integer arithmetic
  - reserving memory for, 92
  - SSE2, 483
- debuggers**
  - description, 43–44
  - gdb (GNU debugger)
    - breakpoints, 82–83
    - command-line parameters, 57–58
    - commands, 58–59
    - description, 56
    - downloading and installing, 56–57
    - example, 59, 81–86
    - invoking, 57
    - memory contents, displaying, 86
    - register values, displaying, 85–86
    - stepping through code, 82–84
    - variables, displaying, 87–88
    - viewing data, 84–86
  - kdbg (KDE debugger), 60–62
  - strace program
    - attaching to a running program, 353–355
    - description, 349–350
    - parameters, 350–351
    - watching system calls, 351–353
- debugging**
  - assembly functions, in C or C++ programs
    - assembly functions, 418–419
    - C programs, 417–418
    - listing source code, 417–418
  - function files, 319–320
  - sample program, 81–86
  - the stack, 314–317
- DEC instruction, 215–216**
- decimal arithmetic**
  - See also BCD (Binary Coded Decimal)
  - See also floating-point arithmetic
  - See also integer arithmetic
  - See also logical operations
  - See also mathematical operations
  - See also SIMD integers
  - packed BCD, 229–231
  - unpacked BCD, 227–229
- decrementing, 215–216**
- deep branch prediction, 22**
- denormalized coefficients, 185**
- device files, Linux kernel, 332**
- device management, Linux kernel, 331–333**
- DF (direction flag), 31**
- dictionary ordering, strings, 289**
- direction flag (DF), 31**
- directives**
  - See also data
  - See also instructions
  - See also *specific directives*
  - .ascii, 92
  - .asciz, 92
  - .byte, 92
  - .comm, 95–97

## directives (continued)

- .data, 91–94
- description, 14–15
- .double, 92
- .equ, 94–95
- .float, 92
- .globl, 75
- .int, 92
- .lcomm, 95–97
- .long, 92
- .octa, 92
- .quad, 92
- .short, 92
- .single, 92
- syntax, 14

## DIV instruction, 221–222

### division

- error detection, 223
- floating-point, 245–248
- shift instructions, 225–226, 226–227
- signed integers, 222–223
- unsigned integers, 221–222

## DIVPS instruction, 499–501

### dollar sign (\$), opcode syntax, 49

- .double directive, 92

### double floating-point values in `gas`, 187

### double-extended-precision, 186–187

### double-precision

- description, 185–186
- moving, 194–196

### doubleword integers, 162

### DS (data segment) registers, 28

### dump program, 62–65

### dynamic data flow analysis, 22

### dynamic libraries, 8

### dynamic linking, 88–89

## E

### EFLAGS register

- bit testing, 232–233
- carry/overflow, addition, 204
- carry/overflow, subtraction, 212
- CMOV instruction, 107–109
- description, 29–30
- setting comparison results, 138–139

### epilogue, passing data, 308–309

- .equ directive, 94–95

### ES (extra segment pointer) registers, 28

- `/etc/ld.so.conf` file, 416–417

- example program. See assembly language, sample program

### exceptions, floating-point, 269

### exchanging data

- data exchange instructions, 111–118
- description, 110–111

### executable files

- assembly libraries, 393–395
- building and running, 80

### executing instructions, 24–25

### execution flow

- branches
  - eliminating, 155–156
  - high-level, 146–152
  - optimizing, 153–158
  - predictable, coding first, 156–157
  - predicting, 153–155

- branches, conditional

- comparing flag bits, 140–143
- comparing values, 138–139
- jump, 136–138
- predicting, 154–155

- branches, unconditional

- calls, 132–135
- interrupts, 135–136
- jumps, 129–132
- predicting, 153–154

- conditional statements

- high-level, duplicating, 146–152
- if statements, 147–150
- for statements, 150–152

- instruction pointers, 127–129

- loops

- description, 144
- error prevention, 145–146
- example, 144–145
- high-level, duplicating, 146–152
- if statements, 147–150
- instructions for, 144–145
- for statements, 150–152
- unrolling, 157–158
- next instruction, determining, 127–129

### execution unit, 24–25

### exit system call, 462–463

### extending

- asm format
  - changed registers list, 377–379
  - constraints, 371

- extended format, 370
- floating-point values, 380–382
- input/output values, specifying, 370–372
- jumps, 382–383
- memory locations, 379–380
- output modifiers, 371
- place holders, alternative, 377
- place holders, description, 373–376
- place holders, referencing, 376
- registers, 372–373
- integers, 169–172
  - signed integers, 170–172
  - unsigned integers, 169–170
- extra segment pointer (ES) registers, 28**
- extra segment pointer (FS) registers, 28**
- extra segment pointer (GS) registers, 28**

## F

- F2XM1 instruction, 249**
- FABS instruction, 249–252**
- FADD instruction, 245–248**
- failed operation indicator. See flags**
- falign-functions optimization, 424**
- falign-loops optimization, 425**
- FBLD instruction, 180–182**
- FBSTP instruction, 180–182**
- fcallee-saves optimization, 424**
- FCBS instruction, 249–252**
- FCMOVB instruction, 264**
- FCMOVBE instruction, 264**
- FCMOVE instruction, 264**
- FCMOVNB instruction, 264**
- FCMOVNBE instruction, 264**
- FCMOVNE instruction, 264**
- FCMOVNU instruction, 264**
- FCMOVU instruction, 264**
- FCOM instruction, 260–262**
- FCOMI instruction, 262–263**
- FCOMP instruction, 262–263**
- FCOS instruction, 249, 254–256**
- fcprop-registers optimization, 423**
- fcrossjumping optimization, 425**
- fcse-follow-jumps optimization, 424**
- fdefer-pop optimization, 422**
- fdelayed-branch optimization, 423**
- fdelete-null-pointer-checks optimization, 424**
- FDIV instruction, 245–248**
- FDIVR instruction, 245–248**

- fexpensive-optimizations optimization, 424**
- fforce-mem optimization, 423**
- fgcse optimization, 424**
- fgcse-after-reload optimization, 425**
- fguess-branch-probability optimization, 423**
- fif-conversion optimization, 423**
- fif-conversion2 optimization, 423**
- file system management, Linux kernel, 333**
- files**

- access types, 455–456
- closing, 454–460
- memory-mapped
  - definition, 470–471
  - example, 475–479
  - mapping, 471–474
  - unmapping, 474–475
- open error return codes, 459
- open file code, 458–459
- opening, 454–460
- processing data, 467–470
- reading, 463–467, 467–470
- UNIX permissions, 456–458
- writing to
  - changing access modes, 462
  - examples, 460–462, 467–470
  - file errors, 462–463

- file-system system calls, 340, 454**

- FIMUL instruction, 181–182**

- finline-functions optimization, 425**

- FISTTP instruction, 508**

- flags. See also return codes**

- OF (overflow flag), 30, 140, 204–206
- AC (alignment check flag), 31
- AF (adjust flag), 30
- CF (carry flag), 30, 143, 204–206
- comparing flag bits, 140–143
- conditional jumps, 140–143
- control, 31
- definition, 29
- DF (direction flag), 31
- groups, 30–32
- ID (identification flag), 31
- IF (interrupt enable flag), 31
- IOPL (IO privilege level flag), 31
- NT (nested task flag), 31
- PF (parity flag), 30, 141
- RF (resume flag), 31
- SF (sign flag), 30, 141–142

## flags (continued)

- status, 30
- system, 31–32
- TF (trap flag), 31
- VIF (virtual interrupt flag), 31
- VIP (virtual interrupt pending flag), 31
- VM (virtual-8086 mode flag), 31
- ZF (zero flag), 30, 140

**FLD instruction, 187–189**

**FLDL instruction, 188–189**

**FLDZ instruction, 190**

**.float directive, 92**

**floating-point arithmetic**

- See also BCD (Binary Coded Decimal) arithmetic
- See also decimal arithmetic
- See also integer arithmetic
- See also logical operations
- See also mathematical operations
- See also SIMD integers
- absolute values, 249–252
- addition, 245–248
- binary fractions, 183–184
- binary scientific notation, 183–184
- conditional branches, 259–265
- conditional data moves, 263–265
- cosines, 254–257
- data types, 33, 184–186
- denormalized coefficients, 185
- description, 182–183
- division, 245–248
- double floating-point values in `gas`, 187
- double-extended-precision, 186–187
- double-precision
  - description, 185–186
  - moving, 194–196
- exceptions, 269
- format, 183
- IA-32 floating-point values, 186–187
- IEEE Standard 754 floating-point formats, 184
- log base 2 calculations, 257–259
- logarithmic functions, 257–259
- moving floating-point values, 187–189
- multiplication, 245–248
- nonwaiting instructions, 269
- normalized coefficients, 185
- optimizing, 270
- partial remainders, 252–254
- preset floating-point values, 189–190
- real numbers, 182–183

- return values, 400–401
- rounding, 249–252
- sign changes, 249–252
- sines, 254–256
- single-precision, 185–186
- square roots, 249–252
- SSE data types
  - description, 190–191
  - moving, 191–193
- SSE2 data types, 191–193
- SSE3 instructions, 196
- subtraction, 245–248
- trigonometric functions, 254–257
- values, `asm` format, 380–382
- waiting instructions, 269

## floating-point unit (FPU)

- BCD values, 179–180
- description, 32–33
- environment, saving and restoring, 265–266
- register stack
  - control register, 239–241
  - description, 236–237
  - status register, 237–238
  - tag register, 241
  - using, 236–237
- state, saving and restoring, 266–269
- `-floop-optimize` **optimization, 422**
- `-fmerge-constants` **optimization, 422**
- FMUL instruction, 245–248**
- FNCLEX instruction, 269**
- FNSAVE instruction, 269**
- FNSTCW instruction, 269**
- FNSTENV instruction, 269**
- FNSTSW instruction, 269**
- `-foptimize-sibling-calls` **optimization, 423**
- for **statements, 150–152**
- FPATAN instruction, 249, 257**
- `-fpeephole2` **optimization, 424**
- FPREM instruction, 249, 252–254**
- FPREM1 instruction, 249, 252–254**
- FPTAN instruction, 249, 257**

**FPU (floating-point unit)**

- BCD values, 179–180
- description, 32–33
- environment, saving and restoring, 265–266
- register stack
  - control register, 239–241
  - description, 236–237
  - status register, 237–238

- tag register, 241
- using, 236–237
- state, saving and restoring, 266–269
- fregmove **optimization, 424**
- freorder-blocks **optimization, 424**
- frerun-cse-after-loop **optimization, 424**
- FRNDINT **instruction, 249–252**
- FS (extra segment pointer) registers, 28**
- FSAVE **instruction, 266–269**
- FSCALE **instruction, 249, 257–259**
- fsched-interblock **optimization, 424**
- fschedule-insns **optimization, 424**
- FSIN **instruction, 249, 254–256**
- FSINCOS **instruction, 249, 256–257**
- FSQRT **instruction, 249–252**
- FST **instruction, 188–189**
- FSTENV **instruction, 265–266**
- FSTL **instruction, 189**
- fstrength-reduce **optimization, 423**
- fstrict-aliasing **optimization, 424**
- FSUB **instruction, 245–248**
- FSUBR **instruction, 245–248**
- fthread-jumps **optimization, 422**
- FTST **instruction, 260–262**
- FUCOMI **instruction, 262–263**
- FUCOMIP **instruction, 262–263**
- functions. See also macros**
- C library
  - conversion utilities, 326–327
  - versus Linux kernel system calls, 355–359
  - sample program, 86–89
  - tracing, 357–358
- calls
  - accessing functions, 302–303
  - optimizing registers, 424
- mathematical
  - logarithmic, 257–259
  - trigonometric, 254–257
- optimizing, 423, 424
- for optimizing code
  - falign-functions, 424
  - falign-loops, 425
  - fcaller-saves, 424
  - fcprop-registers, 423
  - fcrossjumping, 425
  - fcse-follow-jumps, 424
  - fdefer-pop, 422
  - fdelayed-branch, 423
  - fdelete-null-pointer-checks, 424
  - fexpensive-optimizations, 424
  - fforce-mem, 423
  - fgcse, 424
  - fgcse-after-reload, 425
  - fguess-branch-probability, 423
  - fif-conversion, 423
  - fif-conversion2, 423
  - finline-functions, 425
  - floop-optimize, 422
  - fmerge-constants, 422
  - foptimize-sibling-calls, 423
  - fpephole2, 424
  - fregmove, 424
  - freorder-blocks, 424
  - frerun-cse-after-loop, 424
  - fsched-interblock, 424
  - fschedule-insns, 424
  - fstrength-reduce, 423
  - fstrict-aliasing, 424
  - fthread-jumps, 422
  - funit-at-a-time, 424
  - fweb, 425
- functions, assembly (assembly language programs)**
- calling, 302–303
- command-line parameters
  - analyzing the stack, 321–323
  - anatomy of a program, 320–321
  - environment variables, viewing, 325–326
  - example, 326–327
  - viewing, 323–325
- creating for
  - assembly language programs, 301–302
  - C or C++ programs, 389–391
- description, 297–299
- global variables, 304–305
- input values, defining, 300
- output values, defining, 301
- passing data, with the stack
  - cleaning the stack, 312
  - debugging the stack, 314–317
  - example, 312–314
  - function parameters, 306–308
  - function prologue and epilogue, 308–309
  - local function data, 309–310
  - monitoring the stack, 314–317
- placement, 304
- processes, defining, 300
- registers, 304

## functions, assembly (assembly language programs)

---

### functions, assembly (assembly language programs)

#### (continued)

- separate function files
  - creating, 317–318
  - debugging, 319–320
  - executable files, creating, 318–319
- writing, 299–302

### functions, assembly (C or C++ programs). See also assembly libraries; inline assembly

- creating, 389–391
- debugging
  - assembly functions, 418–419
  - C programs, 417–418
  - listing source code, 417–418
- input values
  - mixed data type, 403–406
  - multiple, 401–403
  - ordering, 403–406
  - reading in order, 405–406
- return values
  - floating-point, 400–401
  - integer, 306–397
  - string, 397–399
- shared libraries
  - compiling with, 414–415
  - creating, 414
  - definition, 412–414
  - running programs, 415–417
- static libraries
  - ar command, 409–410
  - compiling with, 412
  - creating, 410–411
  - definition, 408–409
- using, 407–408
- funit-at-a-time **optimization, 424**
- fweb **optimization, 425**
- FYLL2X **instruction, 249**
- FYLL2XP1 **instruction, 249, 257–259**

## G

### gas (GNU assembler)

- defining integers, 172–173
- description, 41–42
- installing, 45–47
- invoking, 47
- opcode syntax, 49–50
- platforms supporting, 45
- using, 47–49

### gcc (GNU compiler)

- description, 53
- downloading and installing, 53–54
- example, 55–56, 80–81
- invoking, 54
- parameters, 54–56

### gdb (GNU debugger). See also kdbg

- breakpoints, 82–83
- command-line parameters, 57–58
- commands, 58–59
- description, 56
- downloading and installing, 56–57
- example, 59, 81–86
- invoking, 57
- memory contents, displaying, 86
- register values, displaying, 85–86
- stepping through code, 82–84
- variables, displaying, 87–88
- viewing data, 84–86

### global C variables, 367–369

### global variables, assembly language functions, 304–305

### GNU assembler (gas)

- defining integers, 172–173
- description, 41–42
- installing, 45–47
- invoking, 47
- opcode syntax, 49–50
- platforms supporting, 45
- using, 47–49

### GNU compiler (gcc)

- description, 53
- downloading and installing, 53–54
- example, 55–56, 80–81
- invoking, 54
- parameters, 54–56

### GNU debugger (gdb). See also kdbg

- breakpoints, 82–83
- command-line parameters, 57–58
- commands, 58–59
- description, 56
- downloading and installing, 56–57
- example, 59, 81–86
- invoking, 57
- memory contents, displaying, 86
- register values, displaying, 85–86
- stepping through code, 82–84
- variables, displaying, 87–88
- viewing data, 84–86

**GNU linker (ld), 49–53**  
**GNU object dump (objdump), 62–65**  
**GNU profiler (gprof), 65–69**  
 gprof (GNU profiler), 65–69  
**GS (extra segment pointer) registers, 28**

## H

**HADDPD instruction, 508**  
**HADDPS instruction, 508**  
**High Level Assembler (HLA), 42**  
**high-level languages (HLLs)**  
 compiled languages, 7–8  
 definition, 6–7  
 features, 9–10  
 hybrid languages, 9  
 interpreted languages, 8–9  
 portability, 9–10  
 standardization, 10  
 types of, 7–9  
**HLA (High Level Assembler), 42**  
**HLLs (high-level languages)**  
 compiled languages, 7–8  
 definition, 6–7  
 features, 9–10  
 hybrid languages, 9  
 interpreted languages, 8–9  
 portability, 9–10  
 standardization, 10  
 types of, 7–9  
**HSUBPD instruction, 508**  
**HSUBPS instruction, 508**  
**hybrid languages, 9**  
**Hyde, Randall, 42**  
**hyperthreading, 34**

## I

**IA-32 processor**  
 address displacement byte, 5–6  
 address size override prefixes, 4–5  
 advanced features *See also* MMX; SIMD; SSE  
 executing multiple threads simultaneously, 34  
 hyperthreading, 34  
 x87 FPU (floating-point unit), 32–33  
 ALUs (Arithmetic Logical Units), 24–25  
 branch hint prefixes, 4  
 core parts  
 allocator, 22–24  
 branch prediction unit, 22  
 complex-integer operations, 24–25  
 control unit, 19–24  
 execution unit, 24–25  
 flags, 29–32  
 floating-point operations, 24–25  
 instruction prefetch and decoding pipeline, 20–22  
 micro-operation rescheduler, 22–24  
 out-of-order execution engine, 22–24  
 overview, 17–19  
 pipelining, 20–22  
 prefetching, 20–22  
 register renaming, 22–24  
 registers, 25–29  
 retirement unit, 24  
 simple-integer operations, 24–25  
 data elements, 6  
 executing instructions, 24–25  
 floating-point values, 186–187  
 instruction code format, 4–6  
 instruction prefix, 4–5  
 layout, 3–4  
 lock prefixes, 4  
 modifiers, 5–6  
 ModR/M (addressing-form specifier) byte, 5  
 opcode, 4  
 operand size override prefixes, 4  
 processor, 18  
 product family  
 AMD processors, 36  
 Cyrix processors, 37  
 Intel processors, 35–36  
 non-Intel processors, 36–37  
 P6 processors, 35  
 Pentium 4 processors, 36  
 Pentium Pro processors, 35  
 Pentium processors, 35  
 Pentium Xeon processors, 36  
 repeat prefixes, 4  
 required bytes, 4  
 segment override prefixes, 4  
 SIB (Scale-Index-Base) byte, 5–6  
**ID (identification flag), 31**  
**identification flag (ID), 31**  
**IDIV instruction, 222–223**  
**IEEE Standard 754 floating-point formats, 184**  
**IF (interrupt enable flag), 31**  
**if statements, 147–150**  
**if-then statements, optimizing, 423**  
**IMUL instruction, 218–220**

# INC instruction

---

**INC instruction, 215–216**

**incrementing, 215–216**

**indexed memory locations, 101–103**

**indirect addressing, 103–106**

**inequality, strings, 289–291**

**init process, Linux kernel, 334–336**

**inline assembly. See also assembly libraries; functions, assembly**

asm format

altering, 369

description, 365–367

asm format, extending

changed registers list, 377–379

constraints, 371

extended format, 370

floating-point values, 380–382

input/output values, specifying, 370–372

jumps, 382–383

memory locations, 379–380

output modifiers, 371

place holders, alternative, 377

place holders, description, 373–376

place holders, referencing, 376

registers, 372–373

C macro functions, 384–385

definition, 361–362

global C variables, 367–369

macro functions, creating, 386–387

macros, 384

optimization, suppressing, 369

sample code, 362–365

volatile modifier, 369

**inline functions, optimizing, 425**

**input value removal, optimizing, 422**

**instruction blocks, reordering, 424**

**instruction cache, optimizing, 424**

**instruction codes**

current, tracking, 2

definition, 1

distinguishing from data codes, 2–3

format (IA-32 family)

address displacement byte, 5–6

address size override prefixes, 4–5

branch hint prefixes, 4

data elements, 6

instruction prefix, 4–5

layout, 3–4

lock prefixes, 4

modifiers, 5–6

ModR/M (addressing-form specifier) byte, 5

opcode, 4

operand size override prefixes, 4

repeat prefixes, 4

required bytes, 4

segment override prefixes, 4

SIB (Scale-Index-Base) byte, 5–6

handling, 2–3

instruction pointer, 2

memory location, specifying, 15

next, tracking, 2

opcodes, 3

required bytes, 3

**instruction pointers**

definition, 2

description, 127–129

**instruction prefix, 4–5**

**instructions. See also data; directives**

AAA, 227–229

AAD, 227–229

AAM, 227–229

AAS, 227–229

ADC, 206–210

ADD, 202–204

ADDB, 204

ADDPD, 506–508

ADDPDS, 499–501

ADDS, 506–508

ADDSUBPD, 508

ADDSUBPS, 508

ADDW, 204

ANDNPS, 499–501

ANDPS, 499–501

BSWAP, 112–113

CALL, 132–135

CLC, 143

CMC, 143

CMP, 138–139

CMPEQS, 503–504

CMPLEPS, 503–504

CMPLTPS, 503–504

CMPNEQPS, 503–504

CMPNLEPS, 503–504

CMPNLTPS, 503–504

CMPORDPS, 503–504

CMPPS, 501–502

CMPSP, 287–289

CMPSP, 287–289

CMPSS, 501–502

CMPSW, 287–289  
CMPUORDPS, 503–504  
CMPXCHG, 114–115  
CMPXCHG8B, 115–116  
COMISS, 501–502  
CPUID, 76–77  
CVTDQ2PD, 197  
CVTDQ2PS, 197  
CVTPD2DQ, 197  
CVTPD2PI, 197  
CVTPD2PS, 197  
CVTPI2PD, 197  
CVTPI2PS, 197  
CVTPS2DQ, 197  
CVTPS2PD, 197  
CVTPS2PI, 197  
CVTTPD2DQ, 197  
CVTTPD2PI, 197  
CVTTPS2DQ, 197  
CVTTPS2PI, 197  
DAA, 229–231  
DAS, 229–231  
DEC, 215–216  
DIV, 221–222  
DIVPS, 499–501  
ENTER, 309  
F2XM1, 249  
FABS, 249–252  
FADD, 245–248  
FBLD, 180–182  
FBSTP, 180–182  
FCHS, 249–252  
FCMOVB, 264  
FCMOVBE, 264  
FCMOVE, 264  
FCMOVNB, 264  
FCMOVNBE, 264  
FCMOVNE, 264  
FCMOVNU, 264  
FCMOVU, 264  
FCOM, 260–262  
FCOMI, 262–263  
FCOMIP, 262–263  
FCOS, 249, 254–256  
FDIV, 245–248  
FDIVR, 245–248  
FIMUL, 181–182  
FISTTP, 508  
FLD, 187–189  
FLDL, 188–189  
FLDZ, 190  
FMUL, 245–248  
FNCLEX, 269  
FNSAVE, 269  
FNSTCW, 269  
FNSTENV, 269  
FNSTSW, 269  
FPATAN, 249, 257  
FPREM, 249, 252–254  
FPREM1, 249, 252–254  
FPTAN, 249, 257  
FRNDINT, 249–252  
FSAVE, 266–269  
FSCALE, 249, 257–259  
FSIN, 249, 254–256  
FSINCOS, 249, 256–257  
FSQRT, 249–252  
FST, 188–189  
FSTENV, 265–266  
FSTL, 189  
FSUB, 245–248  
FSUBR, 245–248  
FTST, 260–262  
FUCOMI, 262–263  
FUCOMIP, 262–263  
FYL2X, 249  
FYL2XP1, 249, 257–259  
HADDDP, 508  
HADDPS, 508  
HSUBPD, 508  
HSUBPS, 508  
IDIV, 222–223  
IMUL, 218–220  
INC, 215–216  
JA, 137  
JAE, 137  
JB, 137  
JBE, 137  
JC, 137  
JCXZ, 137  
JE, 137  
JECXZ, 137  
JG, 137  
JGE, 137  
JL, 137  
JLE, 137  
JMP, 129–132  
JNA, 137

# instructions

---

## instructions (continued)

JNAE, 137  
JNB, 137  
JNBE, 137  
JNC, 137  
JNE, 137  
JNG, 137  
JNGE, 137  
JNL, 137  
JNLE, 137  
JNO, 137  
JNP, 137  
JNS, 137  
JNZ, 137  
JO, 137, 206  
JP, 137  
JPE, 137  
JPO, 137  
JS, 137  
JZ, 137  
LDDQU, 508  
LEAVE, 309  
LODSB, 283–284  
LODSL, 283–284  
LODSW, 283–284  
MAXPS, 499–501  
MINPS, 499–501  
MOVAPD, 194–196, 505  
MOVAPS, 498–499  
MOVDDUP, 196, 508  
MOVDQA, 177–178, 505  
MOVDQU, 177–178, 505  
MOVHLPS, 498–499  
MOVHPD, 194–196, 505  
MOVHPS, 498–499  
MOVL, 169–170  
MOVLHPS, 498–499  
MOVLPD, 194–196, 505  
MOVLPS, 498–499  
MOVQ, 174–176  
MOVSB  
    string moves, basic, 274–278  
    string moves, block by block, 279–280  
    string moves, byte by byte, 278–279  
    string moves, large strings, 281–282  
MOVSD, 194–196, 505  
MOVSHDUP, 196, 508  
MOVSL, 274–278, 279–280  
MOVSLDUP, 196, 508  
MOVSS, 498–499  
MOVSW, 274–278  
MOVSX, 170–172  
MOVUPD, 194–196, 505  
MOVUPS, 498–499  
MOVW, 168  
MOVZX, 169–170  
MUL, 216–218  
MULPS, 499–501  
optimizing, 424  
ORPS, 499–501  
PADDB, 490  
PADDD, 490, 506–508  
PADDQ, 506–508  
PADDSB, 490, 506–508  
PADDSW, 490, 506–508  
PADDUSB, 490  
PADDUSW, 490  
PADDW, 490  
PAND, 494  
PANDN, 494  
PAVGB, 504  
PAVGW, 504  
PCMPEQB, 494  
PCMPEQD, 494  
PCMPEQW, 494  
PCMPGTD, 494  
PCMPGTW, 494  
PEXTRW, 504  
PINSRW, 504  
PMAXSW, 504  
PMAXUB, 504  
PMINSW, 504  
PMINUB, 504  
PMULHUW, 504  
POP, 121  
POR, 494  
PSADBW, 504  
PSLL, 494  
PSRA, 494  
PSUBB, 490  
PSUBD, 490  
PSUBSB, 490  
PSUBSW, 490  
PSUBUSB, 490  
PSUBUSW, 490  
PSUBW, 490  
PUSH, 123  
PXOR, 494

RCL, 226  
 RCPPS, 499–501  
 RCR, 226  
 reordering, 424  
 REP, 278–283, 288–289  
 REPE, 283  
 REPNE, 283  
 REPNZ, 283  
 REPZ, 283  
 ROL, 226  
 ROR, 226  
 RSQRTPS, 499–501  
 SAL, 224–225  
 SAR, 225–226  
 SBB, 214–215  
 SCASB, 292–295  
 SCASL, 292–295  
 SCASW, 292–295  
 SHL, 224–225  
 SHR, 225–226  
 SQRTPS, 499–501  
 STC, 143  
 STOSB, 284–285  
 STOSL, 284–285  
 STOSW, 284–285  
 SUB, 210–214  
 SUBPS, 499–501  
 TEST, 232  
 UCOMISS, 501–502  
 XADD, 113  
 XCHG, 112  
 XORPS, 499–501

**.int directive, 92**

**integer arithmetic**  
 See also BCD (Binary Coded Decimal) arithmetic  
 See also decimal arithmetic  
 See also floating-point arithmetic  
 See also logical operations  
 See also mathematical operations  
 See also SIMD integers  
 addition  
   carry detection, 204–206  
   integer values, 202–204  
   large integers, 206–210  
   overflow detection, 204–206  
 byte integers, 162  
 decrementing, 215–216  
 defining integers in `gas` (GNU assembler), 172–173  
 division See also rotating bits; shift instructions

error detection, 223  
 signed integers, 222–223  
 unsigned integers, 221–222  
 doubleword integers, 162  
 extending integers, 169–172  
 incrementing, 215–216  
 little-endian *versus* big-endian, 162  
 multiplication See also rotating bits; shift instructions  
   overflow detection, 220  
   signed integers, 218–220  
   unsigned integers, 216–218  
 one's complement, 167  
 quadword integers, 162  
 return values, 306–397  
 shift instructions See also division; multiplication  
   description, 223  
   division, 225–226, 226–227  
   MMX (Multimedia Extension), 493–494  
   multiplication, 224–225, 226–227  
   rotating bits, 226–227  
 signed integers  
   description, 166–168  
   extending, 170–172  
 signed magnitude, 166  
 SSE instructions, 504  
 standard sizes, 162  
 subtraction  
   carry detection, 212–214  
   integer values, 210–212  
   large integers, 214–216  
   overflow detection, 212–214  
 two's complement, 167  
 unsigned integers  
   description, 164–165  
   extending, 169–170  
 word integers, 162

**integers**

MMX (Multimedia Extension)  
 description, 173–174 See also SIMD  
 moving, 174–176  
 SSE (Streaming SIMD Extensions)  
 description, 176, 483  
 instructions, 504  
 moving, 177–178

**integers, SIMD**

See also BCD arithmetic  
 See also floating-point arithmetic  
 See also integer arithmetic  
 See also mathematical operations

## integers, SIMD (continued)

- description, 173
- MMX (Multimedia Extension)
  - addition and subtraction, 488–491
  - Boolean instructions, 493–494
  - comparison instructions, 494–496
  - description, 173–174, 482
  - instructions, 490
  - logical instructions, 493–494
  - moving, 174–176
  - multiplication, 491–493
  - overflow, 489–491
  - packed values, loading and retrieving, 487–488
  - shift instructions, 493–494
- SSE (Streaming SIMD Extension) instructions
  - arithmetic, 499–501
  - comparisons, 501–504
  - description, 497
  - integers, 504
  - moving data, 498–499
  - processing data, 499–504
- SSE (Streaming SIMD Extension) integers
  - description, 176, 483
  - instructions, 504
  - moving, 177–178
- SSE2 data types
  - description, 483
  - moving, 191–193
- SSE2 instructions
  - addition, 505–508
  - description, 504–508
  - moving data, 505
  - processing data, 505–508
- SSE3 instructions, 196, 508
- Intel processors, 35–36**
- interblock instructions, enabling, 424**
- interpreted languages, 8–9**
- interrupt enable flag (IF), 31**
- interrupts, 135–136**
- IO privilege level flag (IOPL), 31**
- IOPL (IO privilege level flag), 31**

## J

- JA instruction, 137**
- JAE instruction, 137**
- Java, byte code, 9**
- JB instruction, 137**
- JBE instruction, 137**

- JC instruction, 137**
- JCXZ instruction, 137**
- JE instruction, 137**
- JECXZ instruction, 137**
- JG instruction, 137**
- JGE instruction, 137**
- JL instruction, 137**
- JLE instruction, 137**
- JMP instruction, 129–132**
- JNA instruction, 137**
- JNAE instruction, 137**
- JNB instruction, 137**
- JNBE instruction, 137**
- JNC instruction, 137**
- JNE instruction, 137**
- JNG instruction, 137**
- JNGE instruction, 137**
- JNL instruction, 137**
- JNLE instruction, 137**
- JNO instruction, 137**
- JNP instruction, 137**
- JNS instruction, 137**
- JNZ instruction, 137**
- JO instruction, 137, 206**
- JP instruction, 137**
- JPE instruction, 137**
- JPO instruction, 137**
- JS instruction, 137**

## jumps

- asm format, 382–383
- conditional branches, 136–138
- optimizing
  - branch prediction, 423
  - conditional branches, 442–447
  - functions for, 422–424
  - if-then code, 442–447
  - unconditional branches, 129–132
- JZ instruction, 137**

## K

- kdbg (KDE debugger), 60–62. See also gdb**

## L

- l command, 418**
- .lcomm directive, 95–97**
- ld (GNU linker), 49–53**
- LDDQU instruction, 508**

**lexicographical ordering, strings, 289****library files**

- definition, 8
- dynamic, 8
- static, 8

**linkers**

- definition, 8
- description, 42–43
- ld (GNU linker), 49–53

**linking, sample program, 88–89****Linux kernel**

- device files, 332
- device management, 331–333
- file system management, 333
- init process, 334–336
- memory management, 330–331
- process management, 334–336
- processes, viewing, 334–336
- swap space, 330–331
- system calls
  - versus C libraries, 355–359
  - call values, 342
  - common, 339–341
  - file system, 340
  - finding, 337–338
  - format, 341–346
  - functions, finding, 338–339
  - input values, 343–344
  - memory access, 339–340
  - process system, 341
  - return structure, 347–348
  - return values, 344–349
  - sysinfo call, 346–347
  - tracing, 349–355
  - viewing results, 348–349
- version control, 336–337

**listing source code, 417–418****little-endian values**

- versus big-endian, 162
- definition, 6
- integer arithmetic, 162
- swapping to big-endian, 112–113

**L\_LIBRARY\_PATH environment variable, 415–416****llseek system call, 475****lock prefixes, 4****locking, memory, 4****LODSB instruction, 283–284****LODSL instruction, 283–284****LODSW instruction, 283–284****log base 2 calculations, floating-point, 257–259****logarithmic functions, floating-point, 257–259****logical operations**

- bit testing, 232–233
- Boolean logic, 231–232
- MMX, 493–494

**.long directive, 92****loops. See also execution flow**

- counting down, 142
- description, 144
- error prevention, 145–146
- example, 144–145
- high-level, duplicating, 146–152
- if statements, 147–150
- instructions for, 144–145
- optimizing
  - functions for, 422–424
  - normal for-next, 438–442
  - viewing loop code, 438–440
- for statements, 150–152
- unrolling, 157–158

**M****macro functions, creating, 386–387****macros, 384. See also functions****mapping memory-mapped files, 471–474****MASM (Microsoft Assembler), 41****mathematical functions**

- logarithmic, 257–259
- trigonometric, 254–257

**mathematical operations. See also specific****mathematical operations**

- BCD
  - carry and borrow operations, 30
  - description, 178–179
  - FPU BCD values, 179–180
  - moving BCD values, 180–182
- borrow indicator, 30
- carry indicator, 30
- data corruption indicator, 30
- decimal
  - packed BCD, 229–231
  - unpacked BCD, 227–229
- floating-point
  - absolute values, 249–252
  - addition, 245–248
  - binary fractions, 183–184
  - binary scientific notation, 183–184

## mathematical operations (continued)

- conditional branches, 259–265
- conditional data moves, 263–265
- cosines, 254–257
- data types, 33, 184–186
- denormalized coefficients, 185
- description, 182–183, 185–186
- division, 245–248
- double floating-point values in `gas`, 187
- double-extended-precision, 186–187
- double-precision, 194–196
- exceptions, 269
- format, 183
- IA-32 floating-point values, 186–187
- IEEE Standard 754 floating-point formats, 184
- log base 2 calculations, 257–259
- logarithmic functions, 257–259
- moving floating-point values, 187–189
- multiplication, 245–248
- nonwaiting instructions, 269
- normalized coefficients, 185
- optimizing, 270
- partial remainders, 252–254
- preset floating-point values, 189–190
- real numbers, 182–183
- return values, 400–401
- rounding, 249–252
- sign changes, 249–252
- sines, 254–256
- single-precision, 185–186
- square roots, 249–252
- SSE data types, 190–193
- SSE2 data types, 191–193
- SSE3 instructions, 196
- subtraction, 245–248
- trigonometric functions, 254–257
- values, `asm` format, 380–382
- waiting instructions, 269

logical operations

- bit testing, 232–233
- Boolean logic, 231–232
- MMX, 493–494

mathematical operations

- BCD (Binary Coded Decimal), 30
- borrow indicator, 30
- carry indicator, 30
- data corruption indicator, 30
- overflow indicator, 30

- parity indicator, 30
- sign indicator, 30, 141–142

overflow indicator, 30

- parity indicator, 30
- sign indicator, 30, 141–142

## mathematical operations, integer arithmetic

addition

- carry detection, 204–206
- integer values, 202–204
- large integers, 206–210
- overflow detection, 204–206

byte integers, 162

decrementing, 215–216

defining integers in `gas` (GNU assembler), 172–173

division *See also* rotating bits; shift instructions

- error detection, 223
- signed integers, 222–223
- unsigned integers, 221–222

doubleword integers, 162

extending integers, 169–172

incrementing, 215–216

little-endian *versus* big-endian, 162

multiplication *See also* rotating bits; shift instructions

- overflow detection, 220
- signed integers, 218–220
- unsigned integers, 216–218

one's complement, 167

quadword integers, 162

return values, 306–397

shift instructions *See also* division; multiplication

- description, 223
- division, 225–226, 226–227
- MMX (Multimedia Extension), 493–494
- multiplication, 224–225, 226–227
- rotating bits, 226–227

signed integers

- description, 166–168
- extending, 170–172

signed magnitude, 166

SSE instructions, 504

standard sizes, 162

subtraction

- carry detection, 212–214
- integer values, 210–212
- large integers, 214–216
- overflow detection, 212–214

two's complement, 167

- unsigned integers
  - description, 164–165
  - extending, 169–170
- word integers, 162
- mathematical operations, SIMD integers**
  - description, 173
  - MMX (Multimedia Extension)
    - addition and subtraction, 488–491
    - Boolean instructions, 493–494
    - comparison instructions, 494–496
    - description, 173–174, 482
    - instructions, 490
    - logical instructions, 493–494
    - moving, 174–176
    - multiplication, 491–493
    - overflow, 489–491
    - packed values, loading and retrieving, 487–488
    - shift instructions, 493–494
  - SSE (Streaming SIMD Extension) instructions
    - arithmetic, 499–501
    - comparisons, 501–504
    - description, 497
    - integers, 504
    - moving data, 498–499
    - processing data, 499–504
  - SSE (Streaming SIMD Extension) integers
    - description, 176, 483
    - instructions, 504
    - moving, 177–178
  - SSE2 data types
    - description, 483
    - moving, 191–193
  - SSE2 instructions
    - addition, 505–508
    - description, 504–508
    - moving data, 505
    - processing data, 505–508
  - SSE3 instructions, 196, 508
- MAXPS instruction, 499–501**
- memory (RAM)**
  - access, optimizing, 123
  - cache
    - creating (pipelining), 20–22
    - L1 (level 1), 21
    - L2 (level 2), 21
  - contents, displaying, 86
  - declaring, 14
  - exclusive control (locking), 4
  - instruction code location, specifying, 15
  - linear addresses, 27
  - locations, `asm` format, 379–380
  - logical addresses, 28
  - management, Linux kernel, 330–331
  - moving data
    - indexed memory locations, 101–103
    - indirect addressing, 103–106
    - memory to registers, 99–100
    - pointers, 103
    - registers to memory, 100–101
  - reserving for data types, 92
  - storing and retrieving data, 12–13
- memory-access system calls, 339–340**
- memory-mapped files**
  - definition, 470–471
  - example, 475–479
  - mapping, 471–474
  - unmapping, 474–475
- MEPIS Linux distribution, 70–71**
- micro-operation rescheduler, 22–24**
- microprocessors. See specific processors**
- Microsoft Assembler (MASM), 41**
- MINPS instruction, 499–501**
- MMX (Multimedia Extension)**
  - addition and subtraction, 488–491
  - Boolean instructions, 493–494
  - comparison instructions, 494–496
  - description, 482
  - instructions, 490
  - integers
    - description, 173–174 *See also* SIMD
    - moving, 174–176
  - logical instructions, 493–494
  - multiplication, 491–493
  - overflow, 489–491
  - packed values, loading and retrieving, 487–488
  - shift instructions, 493–494
- mnemonics**
  - definition, 10
  - opcodes, 11
- modifiers, 5–6**
- ModR/M (addressing-form specifier) byte, 5**
- MOV . . . instructions**
  - description, 97–98
  - MOVAPD, 194–196

# MOV... instructions

---

## MOV... instructions (continued)

MOVDDUP, 196  
MOVDQA, 177–178  
MOVDQU, 177–178  
MOVHPD, 194–196  
MOVL, 169–170  
MOVLDP, 194–196  
MOVQ, 174–176  
MOVSD, 194–196  
MOVSHDUP, 196  
MOVSLDUP, 196  
MOVSS, 170–172  
MOVUPD, 194–196  
MOVW, 168  
MOVZX, 169–170

**MOVAPD instruction, 505**  
**MOVAPS instruction, 498–499**  
**MOVDDUP instruction, 508**  
**MOVDQA instruction, 505**  
**MOVDQU instruction, 505**  
**MOVHLPS instruction, 498–499**  
**MOVHPD instruction, 505**  
**MOVHPS instruction, 498–499**

**moving**  
BCD values, 180–182  
floating-point data, 263–265  
floating-point values, 187–189  
MMX (Multimedia Extension) integers, 174–176  
SSE data types, 191–193  
SSE (Streaming SIMD Extension) integers, 177–178  
SSE2 data types, 191–193  
strings  
    block by block, 279–280  
    byte by byte, 278–279  
    doubleword, 274–278  
    large strings, 281–282  
    between memory locations, 274–278  
    from memory to registers, 283–284  
    in reverse order, 282–283  
    single byte, 274–278  
    word, 274–278

**moving, data**  
conditional moves  
    CMOV... instructions, 107–110  
    description, 106–107  
    EFLAGS register, 107

data elements, defining  
    bss section, 95–97  
    data section, 91–94  
    static symbols, 94–95  
exchanging data  
    data exchange instructions, 111–118  
    description, 110–111  
immediate data, to registers and memory, 98–99  
memory access, optimizing, 123  
between memory and registers  
    indexed memory locations, 101–103  
    indirect addressing, 103–106  
    memory to registers, 99–100  
    pointers, 103  
    registers to memory, 100–101  
MOV... instructions, 97–98  
    between registers, 99  
SSE (Streaming SIMD Extension) instructions,  
    498–499  
SSE2, 505  
the stack  
    description, 119–120  
    ESP and EPB registers, 123  
    pushing and popping data, 120–122  
    pushing and popping registers, 123

**MOVLHPS instruction, 498–499**  
**MOVLDP instruction, 505**  
**MOVLPS instruction, 498–499**  
**MOVSB instruction**  
    string moves, basic, 274–278  
    string moves, block by block, 279–280  
    string moves, byte by byte, 278–279  
    string moves, large strings, 281–282

**MOVSD instruction, 505**  
**MOVSHDUP instruction, 508**  
**MOVSL instruction, 274–280**  
**MOVSLDUP instruction, 508**  
**MOVSS instruction, 498–499**  
**MOVSW instruction, 274–278**  
**MOVUPD instruction, 505**  
**MOVUPS instruction, 498–499**  
**msync system call, 472–473**  
**MUL instruction, 216–218**  
**MULPS instruction, 499–501**

**Multimedia Extension (MMX)**  
    addition and subtraction, 488–491  
    Boolean instructions, 493–494

comparison instructions, 494–496  
 description, 482  
 instructions, 490  
 integers  
   description, 173–174 *See also* SIMD  
   moving, 174–176  
 logical instructions, 493–494  
 multiplication, 491–493  
 overflow, 489–491  
 packed values, loading and retrieving, 487–488  
 shift instructions, 493–494

### **multiplication**

floating-point, 245–248  
 MMX (Multimedia Extension), 491–493  
 overflow detection, 220  
 shift instructions, 224–225, 226–227  
 signed integers, 218–220  
 unsigned integers, 216–218

`munmap` **system call**, 472–473

## **N**

**NASM (Netwide Assembler)**, 41

**nested task flag (NT)**, 31

**NetBurst technology**, 19

**Netwide Assembler (NASM)**, 41

**network device files**, 332

`next` **command**, 419

**next instruction, determining**, 127–129

`nmap` **system call**

assembly language format, 473–475  
 description, 471–473  
 main program, 477–478  
 program parts, 475–477  
 watching the program, 478–479

**non-Intel processors**, 36–37

**nonwaiting instructions, floating-point**, 269

**normalized coefficients**, 185

**NOT operator**, 231–232

**NT (nested task flag)**, 31

**null pointers, optimizing**, 424

**numbers. *See also specific mathematical operations***

BCD

carry and borrow operations, 30  
 description, 178–179  
 FPU BCD values, 179–180  
 moving BCD values, 180–182

decimal

packed BCD, 229–231  
 unpacked BCD, 227–229

floating-point

absolute values, 249–252  
 addition, 245–248

binary fractions, 183–184

binary scientific notation, 183–184

conditional branches, 259–265

conditional data moves, 263–265

cosines, 254–257

data types, 33, 184–186

denormalized coefficients, 185

description, 182–183, 185–186

division, 245–248

double floating-point values in `gas`, 187

double-extended-precision, 186–187

double-precision, 194–196

exceptions, 269

format, 183

IA-32 floating-point values, 186–187

IEEE Standard 754 floating-point formats, 184

log base 2 calculations, 257–259

logarithmic functions, 257–259

moving floating-point values, 187–189

multiplication, 245–248

nonwaiting instructions, 269

normalized coefficients, 185

optimizing, 270

partial remainders, 252–254

preset floating-point values, 189–190

real numbers, 182–183

return values, 400–401

rounding, 249–252

sign changes, 249–252

sines, 254–256

single-precision, 185–186

square roots, 249–252

SSE data types, 190–193

SSE2 data types, 191–193

SSE3 instructions, 196

subtraction, 245–248

trigonometric functions, 254–257

values, `asm` format, 380–382

waiting instructions, 269

## numbers (continued)

### logical operations

- bit testing, 232–233
- Boolean logic, 231–232
- MMX, 493–494

### mathematical operations

- BCD (Binary Coded Decimal), 30
- borrow indicator, 30
- carry indicator, 30
- data corruption indicator, 30
- overflow indicator, 30
- parity indicator, 30
- sign indicator, 30, 141–142

## numbers, integer arithmetic

### addition

- carry detection, 204–206
- integer values, 202–204
- large integers, 206–210
- overflow detection, 204–206

### byte integers, 162

### decrementing, 215–216

### defining integers in `gas` (GNU assembler), 172–173

### division *See also* rotating bits; shift instructions

- error detection, 223
- signed integers, 222–223
- unsigned integers, 221–222

### doubleword integers, 162

### extending integers, 169–172

### incrementing, 215–216

### little-endian *versus* big-endian, 162

### multiplication *See also* rotating bits; shift instructions

- overflow detection, 220
- signed integers, 218–220
- unsigned integers, 216–218

### one's complement, 167

### quadword integers, 162

### return values, 306–397

### shift instructions *See also* division; multiplication

- description, 223
- division, 225–226, 226–227
- MMX (Multimedia Extension), 493–494
- multiplication, 224–225, 226–227
- rotating bits, 226–227

### signed integers

- description, 166–168
- extending, 170–172

### signed magnitude, 166

### SSE instructions, 504

### standard sizes, 162

### subtraction

- carry detection, 212–214
- integer values, 210–212
- large integers, 214–216
- overflow detection, 212–214

### two's complement, 167

### unsigned integers

- description, 164–165
- extending, 169–170

### word integers, 162

## numbers, SIMD integers

### description, 173

### MMX (Multimedia Extension)

- addition and subtraction, 488–491
- Boolean instructions, 493–494
- comparison instructions, 494–496
- description, 173–174, 482
- instructions, 490
- logical instructions, 493–494
- moving, 174–176
- multiplication, 491–493
- overflow, 489–491
- packed values, loading and retrieving, 487–488
- shift instructions, 493–494

### SSE (Streaming SIMD Extension) instructions

- arithmetic, 499–501
- comparisons, 501–504
- description, 497
- integers, 504
- moving data, 498–499
- processing data, 499–504

### SSE (Streaming SIMD Extension) integers

- description, 176, 483
- instructions, 504
- moving, 177–178

### SSE2 data types

- description, 483
- moving, 191–193

### SSE2 instructions

- addition, 505–508
- description, 504–508
- moving data, 505
- processing data, 505–508

### SSE3 instructions, 196, 508

**O****objdump (GNU object dump), 62–65****object code disassembler, 44****object code files**

- assembly libraries, 392–393
- definition, 8

**object dump program, 62–65****.octa directive, 92****OF (overflow flag), 30, 140, 204–206****128-bit packed... data types, 162, 176, 191****one's complement, 167****opcode**

- IA-32 microprocessor family, 4
- instruction prefix, 4
- length, 4

**opcode syntax, 49–50****opcodes**

- definition, 3
- mnemonics, 11

**open file code, 458–459****open system call, 454–460****opening, files, 454–460****operand size, switching between 16-bit and 32-bit, 4****operand size override prefixes, 4****operation codes. See opcodes****optimization**

- assembly language code, generating, 425–428
- cse (Common Subexpression Elimination)

- example, 447–450

- fcse-follow-jumps function, 424
- fgcse function, 424
- fgcse-after-reload function, 425
- frerun-cse-after-loop function, 424

functions for

- falign-functions, 424
- falign-loops, 425
- fcaller-saves, 424
- fcprop-registers, 423
- fcrossjumping, 425
- fcse-follow-jumps, 424
- fdefer-pop, 422
- fdelayed-branch, 423
- fdelete-null-pointer-checks, 424
- fexpensive-optimizations, 424
- fforce-mem, 423
- fgcse, 424
- fgcse-after-reload, 425

- fguess-branch-probability, 423

- fif-conversion, 423
- fif-conversion2, 423
- finline-functions, 425
- floop-optimize, 422
- fmerge-constants, 422
- foptimize-sibling-calls, 423
- fpeephole2, 424
- fregmove, 424
- freorder-blocks, 424
- frerun-cse-after-loop, 424
- fsched-interblock, 424
- fschedule-insns, 424
- fstrength-reduce, 423
- fstrict-aliasing, 424
- fthread-jumps, 422
- funit-at-a-time, 424
- fweb, 425

optimized code

- recompiling, 429
- viewing, 429
- suppressing, 369

**optimization, target areas**

- aligning functions, 424
- branches, 153–158
  - conditional, 442–447
  - functions for, 422–424
  - if-then code, 442–447
  - prediction, 423
- calculations, 430–433
- cross-jumping, 425
- eliminating processor waits, 424
- eliminating redundant optimization sections, 425
- enable interblock instructions, 424
- enabling peepholes, 424
- enforcing strict variable rules, 424
- expense of, 424
- floating-point calculations, 270
- forcing variables to registers, 423
- if-then statements, 423
- inline functions, 425
- input value removal, 422
- instruction cache, 424
- jumps, functions for, 422–424
- loops
  - functions for, 422–424
  - normal for-next, 438–442
  - viewing loop code, 438–440

## optimization, target areas (continued)

- merging identical constants, 422
  - null pointers, 424
  - recursive function calls, 423
  - register tying, 424
  - reordering instruction blocks, 424
  - reordering instructions, 424
  - save and restore function call registers, 424
  - subexpression elimination, 424, 447–450
  - suppressing, 369
  - unnecessary copying of registers, 423
  - variables
    - enforcing strict rules, 424
    - forcing to registers, 423
    - optimized global and local variables, 436–437
    - unoptimized global and local variables, 433–436
  - web pseudo-registers, 425
- OR operator, 231–232**
- ORPS instruction, 499–501**
- out-of-order execution engine, 22–24**
- overflow, MMX, 489–491**
- overflow detection**
- OF (overflow flag), 30, 140, 204–206
  - addition, 204–206
  - multiplication, 220
  - subtraction, 212–214
- overflow flag (OF), 30, 140, 204–206**

## P

### P6 processors, 35

#### packed values

- 64-bit packed... data types, 162, 174
- 128-bit packed... data types, 162, 176, 191
- BCD, decimal arithmetic, 229–231
- integer, loading and retrieving in MMX, 487–488

#### PADDB instruction, 490

#### PADD instruction, 490, 506–508

#### PADDQ instruction, 506–508

#### PADDSB instruction, 490, 506–508

#### PADDSW instruction, 490, 506–508

#### PADDUSB instruction, 490

#### PADDUSW instruction, 490

#### PADWD instruction, 490

#### PAND instruction, 494

#### PANDN instruction, 494

#### parity flag (PF), 30, 141

#### partial remainders, floating-point, 252–254

#### PAVGB instruction, 504

#### PAVGW instruction, 504

#### PCMPEQB instruction, 494

#### PCMPEQD instruction, 494

#### PCMPEQW instruction, 494

#### PCMPGTD instruction, 494

#### PCMPGTW instruction, 494

#### peepholes, enabling, 424

#### Pentium 4 processors, 36

#### Pentium Pro processors, 35

#### Pentium Xeon processors, 36

#### percent sign (%), opcode syntax, 49

#### period (.), in directive syntax, 14

#### PEXTRW instruction, 504

#### PF (parity flag), 30, 141

#### PINSRW instruction, 504

#### pipeline decoding, 20–22

#### pipelining, 20–22

#### placeholders, asm format

- alternative, 377

- description, 373–376

- referencing, 376

#### PMAXSW instruction, 504

#### PMAXUB instruction, 504

#### PMINSW instruction, 504

#### PMINUB instruction, 504

#### PMULHUW instruction, 504

#### pointers

- data, 2

- indirect register addressing, 103

- instruction, 2

- stack, 14

#### POP instruction, 121

#### popping. *See pushing and popping*

#### POR instruction, 494

#### portability, high-level languages, 9–10

#### POSIX system call, 475

#### predicting

- conditional branches, 154–155

- unconditional branches, 153–154

#### prefetching, 20–22

#### preset floating-point values, 189–190

#### print command

- debugging C programs, 418

- register values, displaying, 85–86

#### printf function, sample program, 87–88

#### processes, managing and viewing, 334–336

#### processor instructions. *See instruction codes*

**processor wait time, eliminating, 424**

**process-system system calls, 341**

**profilers**

description, 44–45

`gprof` (GNU profiler), 65–69

**program counter, 28–29**

**program execution, controlling. See execution flow**

**program flow control**

branches

eliminating, 155–156

high-level, 146–152

optimizing, 153–158

predictable, coding first, 156–157

predicting, 153–155

branches, conditional

comparing flag bits, 140–143

comparing values, 138–139

jump, 136–138

predicting, 154–155

branches, unconditional

calls, 132–135

interrupts, 135–136

jumps, 129–132

predicting, 153–154

conditional statements

high-level, duplicating, 146–152

`if` statements, 147–150

`for` statements, 150–152

instruction pointers, 127–129

loops

description, 144

error prevention, 145–146

example, 144–145

high-level, duplicating, 146–152

`if` statements, 147–150

instructions for, 144–145

`for` statements, 150–152

unrolling, 157–158

next instruction, determining, 127–129

**program utilities**

assembly development system

Linux basics, 69–70

MEPIS Linux distribution, 70–71

using, 71–72

dump program, 62–65

linkers

description, 42–43

`ld` (GNU linker), 49–53

`objdump` (GNU object dump), 62–65

object code disassembler, 44

profilers

description, 44–45

`gprof` (GNU profiler), 65–69

**program utilities, assemblers**

choosing, 41–42

description, 40

differences between, 40

`gas` (GNU assembler)

defining integers, 172–173

description, 41–42

installing, 45–47

invoking, 47

opcode syntax, 49–50

platforms supporting, 45

using, 47–49

HLA (High Level Assembler), 42

MASM (Microsoft Assembler), 41

NASM (Netwide Assembler), 41

**program utilities, compilers**

description, 44

`gcc` (GNU compiler)

description, 53

downloading and installing, 53–54

example, 55–56, 80–81

invoking, 54

parameters, 54–56

**program utilities, debuggers**

description, 43–44

`gdb` (GNU debugger), 56–59 *See also* `kdbg`

breakpoints, 82–83

command-line parameters, 57–58

commands, 58–59

description, 56

downloading and installing, 56–57

example, 59, 81–86

invoking, 57

memory contents, displaying, 86

register values, displaying, 85–86

stepping through code, 82–84

variables, displaying, 87–88

viewing data, 84–86

`kdbg` (KDE debugger), 60–62

**programs, optimization**

aligning functions, 424

assembly language code, generating, 425–428

branches, 153–158

## programs, optimization (continued)

- conditional, 442–447
- functions for, 422–424
- if-then code, 442–447
- prediction, 423
- calculations, 430–433
- cross-jumping, 425
- cse (Common Subexpression Elimination)
  - example, 447–450
  - fcse-follow-jumps function, 424
  - fgcse function, 424
  - fgcse-after-reload function, 425
  - frerun-cse-after-loop function, 424
- eliminating processor waits, 424
- eliminating redundant optimization sections, 425
- enable interblock instructions, 424
- enabling peepholes, 424
- enforcing strict variable rules, 424
- expense of, 424
- floating-point calculations, 270
- forcing variables to registers, 423
- if-then statements, 423
- inline functions, 425
- input value removal, 422
- instruction cache, 424
- jumps, functions for, 422–424
- loops
  - functions for, 422–424
  - normal for-next, 438–442
  - viewing loop code, 438–440
- merging identical constants, 422
- null pointers, 424
- recursive function calls, 423
- register tying, 424
- reordering instruction blocks, 424
- reordering instructions, 424
- save and restore function call registers, 424
- subexpression elimination, 424, 447–450
- suppressing, 369
- unnecessary copying of registers, 423
- variables
  - enforcing strict rules, 424
  - forcing to registers, 423
  - optimized global and local variables, 436–437
  - unoptimized global and local variables, 433–436
- web pseudo-registers, 425

## programs, optimization functions

- falign-functions, 424
- falign-loops, 425
- fcaller-saves, 424
- fcprop-registers, 423
- fcrossjumping, 425
- fcse-follow-jumps, 424
- fdefer-pop, 422
- fdelayed-branch, 423
- fdelete-null-pointer-checks, 424
- fexpensive-optimizations, 424
- fforce-mem, 423
- fgcse, 424
- fgcse-after-reload, 425
- fguess-branch-probability, 423
- fif-conversion, 423
- fif-conversion2, 423
- finline-functions, 425
- floop-optimize, 422
- fmerge-constants, 422
- foptimize-sibling-calls, 423
- fpeephole2, 424
- fregmove, 424
- freorder-blocks, 424
- frerun-cse-after-loop, 424
- fsched-interblock, 424
- fschedule-insns, 424
- fstrength-reduce, 423
- fstrict-aliasing, 424
- fthread-jumps, 422
- funit-at-a-time, 424
- fweb, 425

## prologue, passing data, 308–309

ps **command**, 334–336

PSADBW **instruction**, 504

PSLL **instruction**, 494

PSRA **instruction**, 494

PSUBB **instruction**, 490

PSUBD **instruction**, 490

PSUBSB **instruction**, 490

PSUBSW **instruction**, 490

PSUBUSB **instruction**, 490

PSUBUSW **instruction**, 490

PSUBW **instruction**, 490

PUSH **instruction**, 123

**pushing and popping data and registers**, 120–123

PXOR **instruction**, 494

**Q**

`.quad` directive, 92  
 quadword integers, 162

**R****RAM (memory)**

access, optimizing, 123  
 cache  
   creating (pipelining), 20–22  
   L1 (level 1), 21  
   L2 (level 2), 21  
 contents, displaying, 86  
 declaring, 14  
 exclusive control (locking), 4  
 instruction code location, specifying, 15  
 linear addresses, 27  
 locations, `asm` format, 379–380  
 logical addresses, 28  
 management, Linux kernel, 330–331  
 moving data  
   indexed memory locations, 101–103  
   indirect addressing, 103–106  
   memory to registers, 99–100  
   pointers, 103  
   registers to memory, 100–101  
 reserving for data types, 92  
 storing and retrieving data, 12–13  
**RCL instruction, 226**  
**RCPPS instruction, 499–501**  
**RCR instruction, 226**  
**read system call, 463–467, 467–470**  
**reading, files, 463–467, 467–470**  
**read-only data section, 92**  
**real numbers, 182–183**  
**redundant sections, eliminating, 425**  
**register stack**  
   control register, 239–241  
   description, 236–237  
   status register, 237–238  
   tag register, 241  
   using, 236–237  
**registers. See also specific registers**  
   `asm` format, 372–373  
   assembly language functions, 304

  comparing values, 114–116  
   control, 29, 32  
   core groups, 25–29  
   data, 32  
   definition, 25  
   EIP (instruction pointer), 28–29  
   ESP and EPB, using manually, 123  
   exchanging values, 112  
   FDP, 33  
   FIP, 33  
   general purpose, 26–27  
   moving data  
     indexed memory locations, 101–103  
     indirect addressing, 103–106  
     memory to registers, 99–100  
     pointers, 103  
     registers to memory, 100–101  
   moving data between, 99  
   opcode, 33  
   optimizing unnecessary copying, 423  
   renaming, 22–24  
   reversing byte order, 112–113  
   segment, 27–28  
   segment register overrides, 4  
   status, 32  
   tag, 32  
   tying, 424  
   values, displaying, 85–86  
   x87 FPU (floating-point unit), 32–33  
**REP instruction, 278–283, 288–289**  
**REPE instruction, 283**  
**repeat prefixes, 4**  
**REPNE instruction, 283**  
**REPNZ instruction, 283**  
**REPZ instruction, 283**  
**resume flag (RF), 31**  
**retirement unit, 24**  
**return codes. See also flags**  
   assembly functions, in C or C++ programs  
     floating-point, 400–401  
     integer, 306–397  
     string, 397–399  
   Linux kernel system calls, 344–349  
   open file error, 459  
**RF (resume flag), 31**  
**.rodata data section, 92**

ROR instruction, 226

rotating bits, 226–227

See also division

See also multiplication

See also shift instructions

rounding, floating-point, 249–252

RSQRTPS instruction, 499–501

## S

SAL instruction, 224–225

sample code, 362–365

sample program. See assembly language, sample program

SAR instruction, 225–226

SBB instruction, 214–215

Scale-Index-Base (SIB) byte, 5–6

scanning strings, 291–295

SCASB instruction, 292–295

SCASL instruction, 292–295

SCASW instruction, 292–295

.section directive. See also specific directives

sections of programs

defining, 74

redundant, eliminating, 425

types of, 73–74

segment override prefixes, 4

SF (sign flag), 30, 141–142

shared libraries, assembly functions in C or C++ programs

compiling with, 414–415

creating, 414

definition, 412–414

running programs, 415–417

shift instructions. See also division; multiplication

description, 223

division, 225–226, 226–227

MMX (Multimedia Extension), 493–494

multiplication, 224–225, 226–227

rotating bits, 226–227

SHL instruction, 224–225

.short directive, 92

SHR instruction, 225–226

SIB (Scale-Index-Base) byte, 5–6

sign changes, floating-point, 249–252

sign flag (SF), 30, 141–142

signed integers

description, 166–168

division, 222–223

extending, 170–172

multiplication, 218–220

signed magnitude, 166

SIMD integers

See also BCD arithmetic

See also floating-point arithmetic

See also integer arithmetic

See also mathematical operations

description, 173

MMX (Multimedia Extension)

addition and subtraction, 488–491

Boolean instructions, 493–494

comparison instructions, 494–496

description, 173–174, 482

instructions, 490

logical instructions, 493–494

moving, 174–176

multiplication, 491–493

overflow, 489–491

packed values, loading and retrieving, 487–488

shift instructions, 493–494

SSE (Streaming SIMD Extension) instructions

arithmetic, 499–501

comparisons, 501–504

description, 497

integers, 504

moving data, 498–499

processing data, 499–504

SSE (Streaming SIMD Extension) integers

description, 176, 483

instructions, 504

moving, 177–178

SSE2 data types

description, 483

moving, 191–193

SSE2 instructions

addition, 505–508

description, 504–508

moving data, 505

processing data, 505–508

SSE3 instructions, 196, 508

**SIMD (Single Instruction, Multiple Data) model**

- description, 33–34
- detecting, 483–487
- feature program, 485–487

**simple-integer operations, 24–25****sines, floating-point, 254–256****.single directive, 92****single-precision, 185–186****64-bit packed... data types, 162, 174****sort routines, 116–119****source code, listing, 417–418****speculative execution, 22****SQRTPS instruction, 499–501****square roots, floating-point, 249–252****SS (stack segment) registers, 28****SSE (Streaming SIMD Extensions)**

- data types
  - description, 190–191
  - moving, 191–193
- description, 33–34
- instructions
  - arithmetic, 499–501
  - comparisons, 501–504
  - description, 497
  - integers, 504
  - moving data, 498–499
  - processing data, 499–504
- integers
  - description, 176, 483
  - instructions, 504
  - moving, 177–178

**SSE2**

- data types
  - description, 483
  - moving, 191–193
- description, 504–508
- instructions
  - addition, 505–508
  - description, 504–508
  - moving data, 505
  - processing data, 505–508

**SSE3 instructions, 196, 508****the stack**

- analyzing, 321–323
- cleaning, 312
- debugging, 314–317

- definition, 2
- description, 119–120
- ESP and EPB registers, 123
- monitoring, 314–317
- passing data in functions
  - example, 312–314
  - function parameters, 306–308
  - function prologue and epilogue, 308–309
  - local function data, 309–310
- pushing and popping data, 120–122
- pushing and popping registers, 123
- storing and retrieving data, 14

**stack segment (SS) registers, 28****standardization, high-level languages, 10****\_start label, 75****starting point, defining, 74–75****static libraries**

- assembly functions, in C or C++ programs
  - ar command, 409–410
  - compiling with, 412
  - creating, 410–411
  - definition, 408–409
- definition, 8

**static linking, 88–89****static symbols, defining, 94–95****status flags, 30****status register, 237–238****STC instruction, 143****stepping through code, 82–84****STOSB instruction, 284–285****STOSL instruction, 284–285****STOSW instruction, 284–285****strace program**

- attaching to a running program, 353–355
- description, 349–350
- parameters, 350–351
- watching system calls, 351–353

**Streaming SIMD Extensions (SSE)**

- data types
  - description, 190–191
  - moving, 191–193
- description, 33–34
- instructions
  - arithmetic, 499–501
  - comparisons, 501–504
  - description, 497

## Streaming SIMD Extensions (SSE) (continued)

integers, 504

moving data, 498–499

processing data, 499–504

integers

description, 176, 483

instructions, 504

moving, 177–178

**string, return values, 397–399**

**strings**

building your own functions, 285–286

comparing, 286–291

dictionary ordering, 289

directional handling, 31

inequality, 289–291

length, determining, 295

lexicographical ordering, 289

moving

block by block, 279–280

byte by byte, 278–279

doubleword, 274–278

large strings, 281–282

between memory locations, 274–278

from memory to registers, 283–284

in reverse order, 282–283

single byte, 274–278

word, 274–278

scanning, 291–295

**SUB instruction, 210–214**

**SUBPS instruction, 499–501**

**subtraction**

carry detection, 212–214

floating-point, 245–248

integer values, 210–212

large integers, 214–216

MMX (Multimedia Extension), 488–491

overflow detection, 212–214

**successful completion indicator. See flags**

**swap space, Linux kernel, 330–331**

**sysinfo call, 346–347**

**system calls, Linux kernel**

versus C libraries, 355–359

call values, 342

common, 339–341

file system, 340

finding, 337–338

format, 341–346

functions, finding, 338–339

input values, 343–344

memory access, 339–340

process system, 341

return structure, 347–348

return values, 344–349

sysinfo call, 346–347

tracing, 349–355

viewing results, 348–349

**system flags, 31–32**

## T

**tag register, 241**

**template for programs, 75**

**TEST instruction, 232**

**text section**

defining, 74

definition, 73

**TF (trap flag), 31**

**threads, multiple simultaneous execution, 34**

**tools**

assembly development system

Linux basics, 69–70

MEPIS Linux distribution, 70–71

using, 71–72

dump program, 62–65

linkers

description, 42–43

ld (GNU linker), 49–53

objdump (GNU object dump), 62–65

object code disassembler, 44

profilers

description, 44–45

gprof (GNU profiler), 65–69

**tools, assemblers**

choosing, 41–42

description, 40

differences between, 40

gas (GNU assembler)

defining integers, 172–173

description, 41–42

installing, 45–47

invoking, 47

opcode syntax, 49–50

platforms supporting, 45

using, 47–49

HLA (High Level Assembler), 42

MASM (Microsoft Assembler), 41

NASM (Netwide Assembler), 41

**tools, compilers**

- description, 44
- `gcc` (GNU compiler)
  - description, 53
  - downloading and installing, 53–54
  - example, 55–56, 80–81
  - invoking, 54
  - parameters, 54–56

**tools, debuggers**

- description, 43–44
- `gdb` (GNU debugger), 56–59 *See also* `kdbg`
  - breakpoints, 82–83
  - command-line parameters, 57–58
  - commands, 58–59
  - description, 56
  - downloading and installing, 56–57
  - example, 59, 81–86
  - invoking, 57
  - memory contents, displaying, 86
  - register values, displaying, 85–86
  - stepping through code, 82–84
  - variables, displaying, 87–88
  - viewing data, 84–86
- `kdbg` (KDE debugger), 60–62

**Torvalds, Linus, 70****tracing**

- C library functions, 357–358
- system calls, Linux kernel, 349–355

**trap flag (TF), 31****trigonometric functions, floating-point, 254–257****two's complement, 167****U****UCOMISS instruction, 501–502****unconditional branches**

- calls, 132–135
- interrupts, 135–136
- jumps, 129–132
- predicting, 153–154

**UNIX permissions, files, 456–458****unmapping memory-mapped files, 474–475****unpacked BCD, decimal arithmetic, 227–229****unrolling loops, 157–158****unsigned integers**

- description, 164–165
- division, 221–222
- extending, 169–170
- multiplication, 216–218

**utilities**

- assembly development system
  - Linux basics, 69–70
  - MEPIS Linux distribution, 70–71
  - using, 71–72
- dump program, 62–65
- linkers
  - description, 42–43
  - `ld` (GNU linker), 49–53
- `objdump` (GNU object dump), 62–65
- object code disassembler, 44
- profilers
  - description, 44–45
  - `gprof` (GNU profiler), 65–69

**utilities, assemblers**

- choosing, 41–42
- description, 40
- differences between, 40
- `gas` (GNU assembler)
  - defining integers, 172–173
  - description, 41–42
  - installing, 45–47
  - invoking, 47
  - opcode syntax, 49–50
  - platforms supporting, 45
  - using, 47–49
- HLA (High Level Assembler), 42
- MASM (Microsoft Assembler), 41
- NASM (Netwide Assembler), 41

**utilities, compilers**

- description, 44
- `gcc` (GNU compiler)
  - description, 53
  - downloading and installing, 53–54
  - example, 55–56, 80–81
  - invoking, 54
  - parameters, 54–56

**utilities, debuggers**

- description, 43–44
- `gdb` (GNU debugger), 56–59 *See also* `kdbg`
  - breakpoints, 82–83
  - command-line parameters, 57–58
  - commands, 58–59
  - description, 56
  - downloading and installing, 56–57
  - example, 59, 81–86
  - invoking, 57
  - memory contents, displaying, 86
  - register values, displaying, 85–86

### utilities, debuggers (continued)

- stepping through code, 82–84
- variables, displaying, 87–88
- viewing data, 84–86
- kdbg (KDE debugger), 60–62

## V

### variables. *See also* data section

- defining, 12–13, 94–95
- displaying, 87–88
- enforcing strict rules, 424
- forcing to registers, 423
- optimizing
  - enforcing strict rules, 424
  - forcing to registers, 423
  - optimized global and local variables, 436–437
  - unoptimized global and local variables, 433–436

### viewing program data, 84–86

### VIF (virtual interrupt flag), 31

### VIP (virtual interrupt pending flag), 31

### virtual interrupt flag (VIF), 31

### virtual interrupt pending flag (VIP), 31

### virtual-8086 mode flag (VM), 31

### VM (virtual-8086 mode flag), 31

### volatile modifier, 369

## W

### waiting instructions, floating-point, 269

### web pseudo-registers, 425

### word integers, 162

### write system call, 460–462, 467–470

### writing to files

- changing access modes, 462
- examples, 460–462, 467–470
- file errors, 462–463

## X

### x command, 86

### x87 FPU (floating-point unit), 32–33

### XADD instruction, 113

### XCHG instruction, 112

### XOR operator, 231–232

### XORPS instruction, 499–501

## Z

### zero flag (ZF), 30, 140

### ZF (zero flag), 30, 140



