

---

# Index

This index begins with symbolic and numeric items, listed in accordance with the ASCII character code ordering (see Table 2-3), followed by the A–Z items.

- " " characters
  - enclosing string data, 60
  - enclosing string parameters, 470
- # character (preprocessor), 201
- \$ character
  - beginning a system-defined macro, 56
  - in symbols, 58
- % character
  - in format strings, 270
  - in parameters for `asm` function, 482
- & character
  - logical AND operator, 63
  - reference indicator (C language), 210
- () characters, clarifying precedence, 63
- \* character
  - multiplication operator, 63
- + character
  - addition operator, 63
  - unary positive operator, 63
- , character, separating specifiers, 55
- character
  - introducing an option (Unix/Linux), 53
  - subtraction operator, 63
  - unary negation operator, 63
- . character
  - introducing a directive, 56
  - in symbols, 58
  - symbol for location counter, 62
- / character
  - division operator, 63
  - double, for comment delimiter, 14, 56
  - examine command (`adb`), 72
- @ character
  - `@gppl`, 15
  - in floating-point class mnemonics, 247
  - in macros, 475
- : character
  - in `adb` commands, 72, 74
  - double, 55
  - terminating a label, 55
- ; character
  - double, 15, 55
  - marking instruction dependencies, 55, 138–140
- = character for symbol definition, 59
- [] characters in register indirect addressing, 111
- \ character

- in macros, 467–470
- in repeat blocks, 462–463
- ^ character for exclusive OR operator, 63
- \_ character in symbols, 58
- | character, logical OR operator, 63
- ~ character, binary complement operator, 63
- 0 prefix for octal, 58–59
- 0b prefix for hexadecimal, 58–59
- 0x prefix for binary, 59
- 1620 computer, 3
- 360 computer, 1
- 4004 processor, 7
- 8008 processor, 7
- 8080 processor, 7–8
- 8085 processor, 7–8
- 8086 processor, 7–8, 26
- 8088 processor, 8, 26
- 80286 processor, 8

## A

- a access mode, 279
- a flag, 473
- a option for requesting assembly listing file, 67
- A class of instruction, 86–88
- a.out (C default output file), 180
- Absolute path (see Path)
- Absolute symbols (see Symbols)
- Absolute value (floating-point), 239
- aCC command (HP compiler), 52, 343
- Access mode, 279
- Accumulator register, 33
- Accuracy (floating-point), 38
- acc completer, 388–390
- Actual parameters, 467–468
- Ada language as a 3GL, 3, 405
- adb debugger (Unix), 53, 70–71, 73–74
- add instruction, 88–89, 94–96
- Addition
  - floating-point, 240
  - integer, 88–89
  - logical basis, 162–163
- addl instruction, 88–89, 94–96
- Address space, 29

## Addresses

- comparing as unsigned values, 129
- definition of, 27
- number of, 32–34
- symbolic, 56

## Addressing modes

- autodecrement, 116, 190
- autodecrement deferred, 118
- autoincrement, 111, 116, 190
- autoincrement deferred, 117–118
- base addressing, 117
- branch, 135–136
- direct, 110–111, 117
- displacement, 117
- displacement deferred, 118
- immediate, 89, 110, 112
- indirect, 111
- Itanium, 110–112
- memory direct, 110–111
- memory indirect, 111
- other architectures, 116–118
- performance issues, 325
- postincrement, 101, 102, 111–112
- register direct, 89, 101–102, 110–112
- register indirect, 101–102, 111–112, 116
- register indirect deferred, 117
- relative, 117
- relative deferred, 117
- for stacks (see Stacks)

## Addressing range for branches, 135–136

- addp4 instruction, 107
- adds instruction, 88–89, 94
- Advanced load (see Load types)
- Ae option for HP-UX compilers, 53, 422
- AINTE function (FORTRAN), 249
- ALAT (advanced load address table), 308–309, 312
- .align directive, 57

## Alignment

- of floating-point data, 38, 40, 236–237
- of integer data, 101–102
- for pipeline efficiency, 298
- alloc instruction, 179–180, 198–199, 313
- Allocation, storage (see Storage allocation)

- Alpha architecture, 7, 9, 34, 84, 301
- Alphanumeric characters, 41–44
- Altivec extensions (PowerPC), 407
- `.altrp` directive, 212–213
- ALU (see Arithmetic logic unit)
- Ampersand character (see `&` character)
- and compare type, 161
- and instruction, 157–158
- AND logical function, 157
- and `.orcm` compare type, 161
- andcm compare type, 161
- andcm instruction, 157–158
- ANSI/IEEE 754 (see IEEE floating-point numbers)
- Answers for selected exercises, 495–502
- Antidependency, 298
- Application registers (see Registers)
- Approximation instructions
  - reciprocal, 253–254
  - reciprocal square root, 254–255
- APPROXPI program, 256–260
- `ar .bsp` register, 212
- `ar .bspstore` register, 212
- `ar .ccv` register, 389
- `ar .csd` register, 389–390
- `ar .ec` register, 313–316
- `ar .fpsr` register, 212, 242–243
- `ar .itc` register, 480–482
- `ar .lc` register, 143–145, 212
- `ar .pfs` register, 180–181, 198–199, 212, 456
- `ar .rnat` register, 212
- `ar .unat` register, 212
- Architecture
  - Alpha, 7, 9, 301
  - CISC, 6–7, 84
  - CRAY-1, 378–379
  - defined, 1
  - EPIC, 6, 10–11, 301
  - extensions to, 393–394
  - IA-16, IA-32, IA-64, 10
  - instruction-level parallelism, 300–301
  - Itanium, 9–11, 34–35
  - lifetime, 396
  - load/store, 33–34
  - one-address, 33
  - PA-RISC, 10–11, 34
  - PDP-8 architecture, 23, 32
  - PDP-11, 7, 9, 33–34, 83
  - Pentium, 35
  - of the piano, 2–3
  - PowerPC, 296, 301
  - RISC, 6–7, 9–11, 84, 300–301, 340–341
  - SPARC, 197–198
  - stack-based, 32–33
  - superscalar, 35, 295–296
  - three-address, 33
  - two-address, 33
  - UltraSparc, 384
  - VAX, 7, 9, 33–34, 84
  - VLIW, 296, 301
  - zero-address, 32–33
- Argument passing
  - for C, 210
  - for COBOL, 210
  - by descriptor, 209–210
  - for FORTRAN, 210
  - locations, 208–209
  - methods, 209–210
  - by reference, 209–210
  - by value, 209–210
- Argument return registers, 214, 451
- Arguments for conditional assembly, 464–465
- Arguments for macros (see Macros)
- Aries (PA-RISC to Itanium migration), 34
- Arithmetic instructions
  - as a category, 30
  - floating-point, 240–243
  - integer, 88–93
  - parallel floating-point, 385
  - parallel integer, 380
  - special cases, 91–92
- Arithmetic logic unit, 156
- Arithmetic overflow
  - floating-point, 243
  - integer, 89–90
- Arithmetic shift, 165
- Arrays
  - record structures, 104–105
  - `shladd` instruction for indexing, 91

- AS/400, 34
  - ASCII characters, 41–44
  - ASCII codes, table of, 43
  - ASCII directive, 60
  - ASCII file (see Text file)
  - ASCII mode (ftp), 415
  - `.asciz` directive, 60
  - `_Asm` functions (HP-UX C), 4, 479–481
  - `asm` keyword (C language), 4, 481–482
  - Assembler directives, 57, 60
  - Assemblers, 3, 49, 57, 66–67, 142
  - Assembly language
    - advantages and disadvantages, 4, 49–50
    - architectural dependence of, 4
    - compared with other languages, 3–4
    - for Itanium, 14–15
    - lack of portability of, 50
    - operators, 56–57
    - statement types, 54–55
    - why study, 4
  - Assembly process, 66–67
  - Asterisk character (see \* character)
  - At character (see @ character)
  - Atomic instructions, 386–390
  - Authors, 503
  - `.auto` directive, 460
  - Autodecrement addressing (see Addressing modes)
  - Autodecrement deferred addressing (see Addressing modes)
  - Autoincrement addressing
    - postincrementing as Itanium equivalent, 111
    - for stacks, 190
  - Autoincrement deferred addressing (see Addressing modes)
  - Automatic registers (see Registers)
  - Automatic variables, 191–192
- B**
- `/b` byte display mode (debuggers), 72
  - `:b` command (adb), 72
  - `b` suffix, temporary labels, 142–143
  - B class of instruction, 86–88
  - B-unit, 87–88
  - Backing store, 199–200
  - Backside cache (see Cache)
  - BACKWARD program, 181–183
  - Banked registers (see Registers)
  - Base addressing (see Addressing modes)
  - Base for number systems
    - binary, 16
    - decimal, 16
    - hexadecimal, 16
    - for machine language, 3
    - octal, 16
  - BASIC language as a 3GL, 3
  - BBEdit, 427
  - BetterTelnet, 427
  - Biased exponent, 37–41, 234–235
  - Bibliography, 485–493
  - Big-endian convention, 36–37
  - `bin` subdirectory, 53, 416
  - Binary files, 288
  - Binary mode (ftp), 416
  - Binary multiples
    - names for, 5
    - prefixes for, 5
  - Binary operators, arithmetic and logical, 62–63
  - Bit encoding of instructions, 93–96
  - Bit-field layouts for instructions, 85–86
  - Bits, numbering of, 36
  - `.body` directive, 57, 145, 180–181
  - Boolean functions, 156–157
  - Boolean operations
    - AND, 157
    - NAND, 157
    - NOR, 157
    - NOT, 92, 157
    - NXOR, 157
    - OR, 157
    - XOR, 157
  - `booth` function, 215–216
  - Booth's algorithm for multiplication, 170–173
  - `br` instruction, 130–131
  - `br.call` instruction, 179–180, 205
  - `br.cexit` instruction, 315
  - `br.cloop` instruction, 143–145
  - `br.ctop` instruction, 315
  - `br.ia` instruction, 408

`br.ret` instruction, 179–180, 205  
`br.wexit` instruction, 315–316  
`br.wtop` instruction, 315–316  
 Branch addressing, 135–136  
 Branch deallocation hint, 130–131  
 Branch instructions
 

- call, 205
- completers, 130–131
- conditional, 130–131
- ordinary, 130–131
- pipeline behavior, 298–299
- return, 205
- unconditional, 130–131

 Branch offset, IP-relative, 135–136  
 Branch penalty
 

- in general, 131, 143
- Itanium 2 processor, 297
- Itanium processor, 402

 Branch prediction, 299, 404  
 Branch prefetch hint, 130  
 Branch range, 135–136  
 Branch registers (see Registers)  
 Branch type, 130  
 Branch whether hint, 130  
`break` command (gdb), 71–73  
`break` statement (C language), 149  
 Breakpoints, 71–74  
`brl` instruction, 136, 405  
`brp` instruction, 404  
`brtype` completer (branch instructions), 130  
 Bubble sort
 

- integers (SORTINT), 284–288
- strings (SORTSTR), 273–277

 Bubbles in a pipeline, 294–300  
 Bundle template (see Template)  
 Bundle of instructions, 35, 85  
 Bus, 26  
`bwh` completer (branch instructions), 130  
`.byte` directive, 60  
 Byte-addressable memory (see Memory)  
 Bytes, numbering of, 36

## C

`/c` display mode (debuggers), 72  
`:c` command (adb), 72  
`.c` file (see File types)  
 C language
 

- as a 3GL, 3
- argument passing, 210
- compiler output, 345–349
- library functions for I/O, 269, 278
- SQUARES program, 12–13

 C++ language as a 4GL, 3  
 Cache
 

- backside, 99
- hit ratio, 120–121, 325
- Itanium 2 processor, 98–100
- Itanium processor, 399–400
- levels, 29, 98–100
- line size, 98–99
- structures, 98–100, 399–400
- subsystem, 29, 98–100

`call` branch type, 130, 205  
 Call to procedure, 205–207  
 Calling conventions, 203–214  
 Caret character (see ^ character)  
 Carriage return character, 415  
 Carry bit, 162–163  
 Case sensitivity (Unix/Linux), 53  
 Case structures, 148–150  
`cat` command (Unix/Linux), 416  
`cc` command (HP compiler), 52, 343, 422  
`cc_bundled` command (HP compiler), 52, 343, 422  
`cd` command (DOS, Unix/Linux), 416  
 CDC 6600, 340  
 Central processing unit, 26–27  
`cexit` branch type, 130, 315  
`cfm` register, 199, 205–207  
 Characters
 

- ASCII codes, 41–44
- byte storage for, 44
- control, 44
- `cr` (for line termination), 415
- `lf` (for line termination), 415

 Check load (see Load types)

- chk.a instruction, 309, 397, 405
- chk.s instruction, 310–311, 397, 405
- chrget routine, 178–179
- chrput routine, 178–179
- Circumflex character (see ^ character)
- CISC architecture (see Architecture)
- CISC instructions, power of, 325–326
- Classes of instructions (see Instruction classes)
- Classification of floating-point value, 247–248
- Clearing a register, 92, 239
- Client software, 426–428
- Clipping saturation (see Saturation)
- clock function (C), 333–334
- CLOCKS\_PER\_SEC, 333
- clloop branch type, 130, 143–145
- Closed routines, 465
- Closing a file, 279
- clr deallocation hint, 130–131, 309
- cmp instruction, 126–129
- cmp4 instruction, 126–129
- cmpxchg instruction, 389
- COBOL language
  - as a 3GL, 3, 405
  - argument passing, 210
  - SQUARES program, 13–14
- Codes
  - for operations, 56–57
  - for pseudo-operations, 56–57
- Colon character (see : character)
- COM\_C program, 345
- COM\_F program, 345
- Comma character (see , character)
- Command-line programming environments, 50, 413–417
- Command-line prompt, 414
- Commands, DOS and Unix/Linux, 416
- Comment field
  - in assembly language, 56
  - importance of, 77–78
- .common directive, 371
- Comparative instructions
  - as a category, 30
  - floating-point, 246
  - integer, 126–129
- Compare instructions
  - completers for, 127–129
  - floating-point, 246
  - parallel floating-point, 385
  - parallel integer, 379
  - parallel logical types, 160–161
  - signed integer, 127–128
  - unconditional, 146
  - unsigned integer, 128–129
- Compare relationship
  - floating-point, 246
  - signed integer, 128
  - unsigned integer, 129
- Compare type, 128, 146
- Comparing addresses, 129
- Comparing source files, 54
- Compiler warning messages, 360–361
- Compilers, 3, 49, 339–341
- Complementation of a Boolean, 92
- Completers, 55, 100 (also see individual instructions)
- Complex instruction set computer (see Architecture, CISC)
- Computer architecture (see Architecture)
- Computer languages (see Languages)
- Computer structures
  - central processing unit, 26–27
  - input and output devices, 29–30
  - memory, 29
- cond branch type, 130
- Condition codes, 124–125
- Conditional assembly, 464–465
- Conditional branch instruction, 130–131
- Constants, 58
  - assembler syntax, 58–59
  - loading into registers, 91, 103–104
- Contents of an information unit, 27
- continue command (gdb), 72
- Control characters, 44
- Control dependency, 310–311
- Control instructions
  - branches, 123–126
  - as a category, 30
- Control path, 27

- Control registers (see Registers)
  - Control speculation (see Speculation)
  - Control statements (see Statement types)
  - Control string for formatted I/O, 270, 280
  - Control structures
    - case structures
    - if-based structures, 131–134
    - loop structures, 134–135
  - Conventions for argument passing, 208–210
  - Conventions for register use, 204, 450–455
  - Conversion instructions, 248–251
  - Copying, register to register, 91
  - Coroutines, 202–203
  - Counted loops (see Loops)
  - Counter, location (see Location counter)
  - CPU (see Central processing unit)
  - `cpuid` registers, 409
  - `cr` character for line termination, 415
  - CRAY-1 architecture (see Architecture)
  - `crel` completer (compare instructions), 128–129
  - Cross-reference table, 64
  - `ctop` branch type, 130, 315
  - `ctype` completer (compare instructions), 128
  - `czx` instruction, 380, 392
- D**
- `/d` display mode (debuggers), 72
  - `:d` command (adb), 72
  - `.data` directive, 57
  - Data access instructions, 98, 101–105
  - Data conversion instructions, 248–251
  - Data dependency, 140–142, 297–300, 307–310
  - Data movement instruction
    - as a category, 30
    - `mov` pseudo-op (floating-point), 239
    - `mov` pseudo-op (integer), 91
    - `movl` instruction, 103–104
    - (see also Load, Store)
  - Data speculation (see Speculation)
  - Data stalls, 298
  - Data types
    - alphanumeric characters, 41–44
    - floating-point numbers, 37–41
    - integers, 37
    - logical, 156
  - `data1` directive, 60
  - `data2` directive, 60
  - `data4` directive, 60
  - `data8` directive, 60
  - Datapath, 6, 27, 84
  - `+DD64` option (HP-UX compilers), 53, 423
  - Dead code, 361
  - Deallocation hint, 130–131
  - Debuggers
    - adb example, 73–74
    - breakpoints, 71–74
    - capabilities of, 71
    - disassembly, 76–77
    - gdb example, 71–73
    - and optimization, 369–370
    - role of, 69–70
    - single-stepping, 75–76
    - watchpoints, 74–75
  - Declarative statements (see Statement types)
  - DECNUM program, 175–178
  - DECNUM2 program, 194–196
  - DECNUM3 program, 216–218
  - `.default` directive, 460
  - `del` command (DOS), 416
  - Delay slot (RISC), 301
  - `delete` command (gdb), 72
  - `dep` instruction, 167–168
  - `dep.z` instruction, 167–168
  - Denormal numbers, 233
  - Dependency
    - control (see Control dependency)
    - data (see Data dependency)
  - Deposit instructions, 167
  - Depth of pipeline (see Pipeline depth)
  - `%DESCR` function (FORTRAN), 210
  - Descriptor, passing argument by, 209–210
  - DET (exception detection pipeline stage), 296–297, 401–402
  - Device drivers, 30
  - `dh` completer (branch instructions), 130–131
  - `diff` command (Linux/Unix), 54
  - Digital Equipment Corporation processors

- Alpha, 7, 9
  - PDP-8, 23, 32
  - PDP-11, 7, 9, 33
  - VAX, 7, 9, 33
  - `dir` command (DOS), 416
  - Direct addressing (see Addressing modes)
  - Direct assignment of a symbol, 58
  - Direct memory access controllers, 30, 200
  - Directives, assembler, 56–57
  - Directories, 266–267
  - `disas` command (gdb), 76–77
  - Disassembly, 76–77
  - Displacement addressing (see Addressing modes)
  - Displacement addressing deferred (see Addressing modes)
  - Display modes for debuggers, 71–72
  - Display output, 268–270
  - Division
    - floating-point, 255–256
    - floating-point software routines, 256
    - IEEE requirement, 253
    - integer, 93, 173–175, 219–220
    - integer software routines, 219–220
    - `shr` instruction for powers of 2, 166
    - by ten, 173–175
    - by zero (floating-point), 243
  - DMA controllers (see Direct memory access controllers)
  - Dollar sign character (see \$ character)
  - DOS commands, 416
  - Dot character (see Period character)
  - Dot product (see Scalar product)
  - DOTCLOOP program, 143–145
  - DOTCTOP program, 316–321
  - DOTCTOP2 program, 321–324
  - DOTLOOP program, 136–137
  - DOTPROD program, 107–109, 113–116
  - `.double` directive, 60
  - Double precision, 38–40
  - Double word, 35–36
  - `dpnt` branch whether hint, 130, 404
  - `dptk` branch whether hint, 130, 404
  - +DSblended option, 407
  - +DSitanium option, 407
  - +DSitanium2 option, 407
  - +DSmckinley option, 407
  - Dual issue, 401
  - Dynamic binding, 219
  - Dynamic optimization, 369
- ## E
- `/e` display mode (adb), 72
  - `ecc` command (Intel C/C++ compiler), 53, 342, 422
  - Editors, use of, 52
  - `efc` command (Intel FORTRAN compiler), 342, 422
  - Effective address, 111–112, 116
  - Ei (symbol for binary prefix), 5
  - ELF binary file format, 210
  - `.else` directive, 464
  - Encapsulation, 178–179
  - End of file, 278, 280, 283
  - Endian conventions (big- and little-), 36–37
  - `.endif` directive, 464–465
  - `.endm` directive, 466–467
  - `.endp` directive, 57
  - `.endr` directive, 460–463
  - Enforced lazy mode (RSE), 199
  - EOF condition, 278, 280, 283
  - EPIC architecture (see Architecture)
  - EPIC instruction-level parallelism, 301
  - Epilog counter, 205, 313–316
  - Epilog phase of loop, 314
  - Epilogue, 210–211
  - `eq` compare relationship, 128
  - Equal sign (see = character)
  - `.err` directive, 472
  - Error detection and printing, 288
  - exa- (decimal prefix), 5
  - exbi- (binary prefix), 5
  - Exceptions, 243
  - `excl` completer, 326–327
  - EXE (execute pipeline stage), 296–297, 401–402
  - Executable file, 49
  - Execution control with debugger, 71
  - Execution units (B, F, I, M), 87–88, 305
  - `exit` branch predict hint, 404



- .exitm directive, 472
- EXP (instruction dispersal pipeline stage), 296–297, 401–402
- .explicit directive, 460
- Explicit parallelism, 302
- Explicitly parallel instruction computer (see Architecture, EPIC)
- Exponent, 37–41, 234–235
- Expressions, 62–63
- External symbol (see Symbols)
- extr instruction, 166–168
- extr.u instruction, 166–168
- Extract instructions, 166–167
- F**
- f command (adb), 72
- .f file (see File types)
- f option (Linux compilers), 342
- f suffix, temporary labels, 142–143
- F class of instruction, 86–88
- F-unit, 87–88
- f90 command (HP compiler), 343, 422
- fabs instruction, 239
- Factorial function, 337, 471–472
- fadd instruction, 240
- famax instruction, 242
- famin instruction, 242
- fand instruction, 252
- fandcm instruction, 252
- fault completer, 326–327
- fclass instruction, 247–248
- fclose function (C), 278–279
- fcmp instruction, 246
- fcvt instruction, 249–250
- Feature size, 395
- FET (instruction prefetch pipeline stage), 401–402
- Fetch, 427
- fetchadd instruction, 388–390
- few prefetch hint, 130
- .fframe directive, 212–213
- fgets function (C), 278, 279–280
- fib function, 370–373
- FIB1 function, 329–331
- FIB2 function, 331–332
- Fibonacci numbers, 328–334, 337, 370–373
- Fields in assembly language statements
  - comments, 56, 77–78
  - labels, 55
  - operators, 55
  - specifiers, 55
- Fields in record structures, 104–105
- Figure of merit for loops, 140
- File name, 278–279
- File pointer, 278–279
- File storage (logical and physical), 265–267
- File systems, 266–267
- File types, default naming of
  - .c (Unix/Linux), 225
  - .f (Unix/Linux), 225
  - .o (Unix/Linux), 356
  - .s (Unix/Linux), 52
- Fill form of load instruction
  - floating-point, 237–238
  - integer, 101–102
- Flag value on stack, 195–196
- Floating-point instructions
  - arithmetic, 240–243
  - compare, 246
  - compared to integer instructions, 232
  - conversion, 248–251
  - load, 236–238
  - logical, 252–253
  - store, 235–236
- Floating-point numbers
  - double extended precision, 234
  - double precision, 38–41, 234
  - IEEE special values, 233
  - IEEE standards, 37–41
  - memory representation, 37–41
  - natural alignment of, 38, 40
  - register representation, 39–40, 234–235
  - single-precision, 39–40, 234
- Floating-point registers (see Registers)
- Floating-point status register (FPSR), 242–243
- Flynn’s classification, 378–379
- fma instruction, 241
- fmax instruction, 242
- fmerge instruction, 239

fmin instruction, 242  
 fmix instruction, 385  
 fmpy instruction, 240  
 fms instruction, 241  
 fneg instruction, 239  
 fnegabs instruction, 239  
 fnma instruction, 241  
 fnmpy instruction, 240  
 fnorm instruction, 241–242  
 fopen function (C), 278–279  
 for instruction, 252  
 Formal parameter, 462, 466–467  
 Format
 

- assembly language statements, 55–56
- Itanium instructions, 85–86

 Format control string for I/O, 270, 280  
 Formatted I/O, 270, 280  
 FORTRAN language
 

- as a 3GL, 3
- argument passing, 210
- compiler output, 349–352
- SQUARES program, 13

 Forwarding, 299  
 fpabs instruction, 385  
 fpack instruction, 385–386  
 fpamax instruction, 385  
 fpamin instruction, 385  
 fpcmp instruction, 385  
 fpcvt instruction, 385  
 fpma instruction, 385  
 fpmax instruction, 385  
 fpmerge instruction, 385  
 fpmin instruction, 385  
 fpmmpy instruction, 385  
 fpms instruction, 385  
 fpneg instruction, 385  
 fpnegabs instruction, 385  
 fpnma instruction, 385  
 fpnmpy instruction, 385  
 fprcpa instruction, 385  
 \_\_fpreg data type, 480  
 fprintf function (C), 278, 280  
 fprsqrta instruction, 385  
 FPSR (floating-point status register), 242–243

fpswap instruction, 385  
 fpsxt instruction, 385  
 fputs function (C), 278–280  
 Fraction, 37–41  
 Frame marker, 205–207  
 Frame size, 191–192  
 frcpa instruction, 253–254  
 FreeBSD, 425  
 Free Software Foundation, 417  
 frsqrta instruction, 254–255  
 fscanf function (C), 278, 280  
 fselect instruction, 252–253  
 fsub instruction, 240  
 fswap instruction, 385  
 fsxt instruction, 385  
 ftp program, use of, 415–416  
 Full adder, 162–163  
 Full stop (see . character)  
 Functional units (see Execution units)  
 Functions, 203  
 Fused multiply–add instruction, 241  
 fxor instruction, 252

## G

/g display mode (debuggers), 72  
 g77 command (Linux compiler), 341  
 gcc command (Linux compiler), 53, 341  
 gdb debugger (Linux), 53, 71–73  
 ge compare relationship, 128  
 GEM compilers, 341  
 General registers (see Registers)  
 get (ftp command), 415  
 getchar function (C), 178–179  
 getf instruction, 250  
 getput (encapsulated C routines), 178–179  
 gets function (C), 269–270  
 gettimeofday function (Unix/Linux), 222–224  
 Gettysburg address, 283–284  
 geu compare relationship, 129  
 Gi (symbol for binary prefix), 5  
 gibi- (binary prefix), 5  
 giga- (decimal prefix), 5  
 .global directive, 57

- Global label (see Label)
  - Global pointer (see Registers)
  - Global symbols (see Symbols)
  - GNU Project, 417
  - `goto` instructions, 123
  - `gp` register, 103, 113, 451
  - `@gprel` assembler indicator, 15, 218–219
  - Groups of instructions, 137–138
  - `gt` compare relationship, 128
  - `gtu` compare relationship, 129
  - Guard bits, 234
- H**
- `h` completer, 380
  - `/h` display mode (adb), 72
  - Half adder, 162
  - Harvard architecture, 99–100
  - Hazards, pipeline
    - branch effects, 298–299
    - data stalls, 298
    - multiple-issue effects, 300
    - producer–consumer effects, 299–300
  - Hewlett-Packard processors
    - 7xxx series, 11
    - 8xxx series, 11
    - PA-RISC, 10–11, 23, 34
  - HEXNUM program, 96–98
  - HEXNUM2 program, 163–164
  - Hidden bit, 37–41
  - High-level languages (see also Languages)
    - defined, 3–4
    - portability of, 49–50
    - standardization of, 49–50
  - Hints
    - for branch instructions, 130–131
    - for load instructions, 102
    - for store instructions, 101
  - Hints for exercises, 495–502
  - Hit ratio (see Cache)
  - HORNER program, 243–245
  - Horner’s rule, 243–244
  - HP-UX software, 422–423
  - Hyphen character (see – character)
- I**
- `/i` display mode (debuggers), 72
  - I class of instruction, 86–88
  - I-unit, 87–88
  - `ia` branch type, 130, 408
  - IA-16, IA-32, IA-64 architectures, 10
  - IA-32 instruction set mode, 408–409
  - `ias` (Intel assembler), 422
  - IBM processors
    - 1620, 3
    - 360, 1, 7
    - 801, 340
    - AS/400, 34
    - System/36, 34
    - System/38, 34
  - Identifier (synonym for symbol), 58–59
  - IEEE floating-point numbers
    - ANSI/IEEE specification, 38
    - denormal, 233
    - double precision, 38–41
    - infinity, 233
    - NaN (not a number), 38, 233
    - single precision, 39–41
    - special values, 38, 233
    - zero, 233
  - `.if` directive, 464–465
  - `.ifdef` directive, 464–465
  - `.ifndef` directive, 464
  - `.ifnotdef` directive, 464
  - `if...then...else` structures, 131–134, 146–147
  - ILP (see Instruction-level parallelism)
  - ILP32 scheme, 422
  - Immediate addressing (see Addressing modes)
  - Immediate data in instructions, 86, 95–96
  - `imp` importance hint, 404
  - Imperative statements (see Statement types)
  - Implementation
    - changes to, 394–397
    - defined, 1
    - of the piano, 2–3
    - version, determining, 409
  - IN instruction for I/O, 30
  - `in0–in7` stacked registers, 198, 205–207, 451
  - `#include` directive (preprocessor), 201

- `.include` directive, 64
- Indefinite repeat block, 462–463
- Indexing of arrays (see Arrays)
- Indirect addressing (see Addressing modes)
- `@inf` mnemonic for `fclass`, 247
- Infinity as IEEE number, 233
- Information units
  - address of, 27–28
  - bytes, 35–36
  - contents of, 27
  - double words, 35–36
  - quad words, 35–36
  - size of, 28–29, 35
  - words, 35–36
- Inline assembly, 479–483
- Inline functions, 327, 366–369
- INLINE program, 367
- Inner product (see Scalar product)
- Input stacked registers, 198
- Input/output
  - C functions, 268–270, 277–280
  - system, 29–30
- Instruction architectures
  - load/store, 33–34
  - one-address, 33
  - stack-based, 32–33
  - three-address, 33
  - two-address, 33
  - zero-address, 32–33
- Instruction bundles, 35, 85
- Instruction categories
  - arithmetic, 30, 88–93, 380, 385
  - comparative, 30, 380, 385
  - control, 30
  - data access, 98, 101–105
  - data movement, 30
  - logical, 30
  - semaphore, 31, 386–390
- Instruction classes (A, B, F, I, M, X), 86–88
- Instruction completers (see Completers)
- Instruction components
  - operand specifiers, 31
  - operation code (opcode), 31
- Instruction encoding at the bit level, 93–96
- Instruction execution cycle, 31–32
- Instruction groups (see Groups)
- Instruction issue, 295, 306–307
- Instruction-level parallelism
  - difficult outside loops, 406
  - EPIC, 301
  - RISC, 300–301
  - throughput advantage, 300
  - VLIW, 301
- Instruction pipelining, 294–295
- Instruction pointer, 27, 31–32, 449–450
- Instruction power, 325–326
- Instruction reordering, 327
- Instruction retiring, 295
- Instruction set architecture, 5–6, 10, 32–34
- Instruction size, 325
- Instruction slots, 85
- Instruction templates, 302–307
- Instruction widths, 83–85
- Instructions (see Itanium instruction set)
- `int` data type (C), 178–179
- Integer division (see Division)
- Integer multiplication (see Multiplication)
- Integer sizes
  - byte, 37
  - double word, 37
  - quad word, 37
  - word, 37
- Integers
  - in floating-point registers, 235
  - representations of, 16–20, 37
  - signed, 18–20, 37
  - unsigned, 17–18, 37
- Intel assembler (`ias`), 422
- Intel processors
  - 4004, 7
  - 8008, 7
  - 8080, 7–8
  - 8085, 8
  - 8086, 7–8, 26
  - 8088, 7, 26
  - 80286, 8
  - Intel386, 8, 46
  - Intel486, 8

- Itanium, 9–11, 397–405
    - Itanium 2, 10–11
    - Pentium, 8
  - Intel386 CPU, 8, 46
  - Intel486 CPU, 8
  - Interface, user-visible, 1
  - Interrupt handling, 200
  - Interval time counter, 480–482
  - Intrinsic functions, 479–481
  - `invalida` instruction, 309–310
  - I/O (see Input/output)
  - IO\_C program, 179–180
  - `-ip` option (Intel compilers), 342
  - IP (see Instruction pointer)
  - IPF (see Itanium Processor Family)
  - IPG (IP generation pipeline stage), 296–297, 401–402
    - `-ipo` option (Intel compilers), 342
    - `.irp` directive, 462
    - `.irpc` directive, 463
  - ISA (instruction set architecture), 5–6, 10
  - ISAM files, 288
  - Issue, multiple (see Multiple issue)
  - Issue ports, 400–401
  - Issuing of instructions, 295
  - Itanium architecture, 9–11
  - Itanium instruction set
    - by function, 430–437
    - by opcode, 438–447
  - Itanium 2 processor
    - cache, 98–100
    - contrasted with first Itanium processor, 397–399
    - execution units, 87–88
    - latency factors, 402–403
    - pipelines, 296–297
  - Itanium processor
    - cache, 399–400
    - contrasted with Itanium 2 processor, 397–399
    - execution units, 400–401
    - latency factors, 402–403
    - pipelines, 401–402
  - Itanium Processor Family, 10–11
- J**
- `/j` display mode (`adb`), 72
  - Java, 3, 42
  - JMPE instruction (IA-32 mode), 408
  - Jump table, 148
- K**
- Kernel phase of loop, 314
  - Keyboard input, 268–270
  - Keyword parameters, 469–470
  - Ki (symbol for binary prefix), 5
  - kibi- (binary prefix), 5
  - kilo- (decimal prefix), 5
- L**
- `l` completer, 380
  - Label
    - global, 55
    - local, 56
    - local in macros, 470–471
    - macro-generated, 475
    - temporary, 142–143
  - Label field, 55
  - Languages
    - 1GL, 2GL, 3GL, 4GL, 3
    - ANSI C, 3
    - artificial intelligence, 3
    - assembly, 3–4
    - database access, 3
    - high-level, 3–4
    - machine, 3
    - natural language, 3
    - object-oriented, 3
  - Latency, 131, 140–142, 295, 299, 402–403
  - LC, location counter, 64–66
    - `.lcomm` directive, 60
  - `ld` instructions, 101–102, 389–390
  - LDC instruction (PA-RISC), 387
  - `ldf` instructions, 236–237
  - `le` compare relationship, 128
  - `leu` compare relationship, 129
  - `lf` character for line termination, 415
  - `lfetch` instructions, 326–327

- Libraries in linking process, 53
  - `lincoln.txt` test file, 283
  - Line feed character, 415
  - Line numbers in listing file, 64–66
  - Line prefetch instructions, 326–327
  - Line size (see Cache)
  - Line terminators in text files, 279–280, 415
  - Linear congruential method, 221
  - Link map, 68–69
  - Linkers, 49, 68–69
  - Linking process, 68–69
  - Linux operating system
    - client software, 426
    - commands, 53, 416
    - I/O software, 267
    - line terminator, 415
    - online documentation, 417
    - support for Unicode, 42
  - LISP language as a 4GL, 3
  - Listing file, 51, 57, 64–67
  - Little-endian convention, 36–37
  - Live quantity in software pipeline, 317
  - Load hints
    - floating-point, 237–238
    - integer, 102
  - Load instructions
    - floating-point, 236–238
    - floating-point pair, 238
    - integer, 101–102
    - semaphore, 389–390
  - Load types
    - advanced, 308–309
    - check, 308
    - floating-point, 236–238
    - integer, 102
    - speculative, 310–312
    - speculative advanced, 311–312
  - `loc0–loc127` stacked registers, 198, 205–207, 451
  - Local labels (see Label)
  - Local stacked registers, 198
  - Local variables, 205–207, 328
  - Locality, 136
  - Location counter
    - . character as symbol for, 62
    - in listing file, 64–66
    - maintained by assembler, 57
    - multiple instances, 61–62
    - in object file sections, 473
  - Logging in and out, 413–414
  - Logical data, 156
  - Logical difference, 158
  - Logical functions
    - binary, 156–157
    - unary, 156–157
  - Logical instructions
    - as a category, 30
    - floating-point, 252–253
    - integer, 157–158
  - Logical mask, 159, 163–164
  - Logical product, 158
  - Logical shift, 165
  - Logical sum, 158
    - .long directive, 60
  - `long long int` (C data type), 178–179
  - Long shift instruction, 166
  - `loop` branch predict hint, 404
  - Loop count register, 143–145, 315
  - Loop structures, 134–135, 314–316
  - Loops
    - counted, 134, 314–315
    - modulo-scheduled, 313–324
    - unrolling, 312–313, 361–366
    - while, 315–316
  - Lower case usage, 53
  - Lower case, converting to upper, 159
  - LP64 scheme, 423
  - `ls` command (Unix/Linux), 416
  - `lt` compare relationship, 128
  - `ltu` compare relationship, 129
- ## M
- `-M` option for requesting map file, 68
  - `-m` option (Linux compilers), 342
  - M class of instruction, 86–88
  - M-unit, 87–88
  - Macintosh

- client software, 426–427
- line terminator, 415
- .macro directive, 466–467
- Macro libraries, 51
- Macros
  - actual parameters, 466–470
  - assembler, 200, 465–472
  - default values, 469–470
  - defining, 466–467
  - formal parameters, 466–470
  - invoking, 467–468
  - keyword parameters, 469–470
  - names, 466–467
  - positional parameters, 468–469
  - purposes, 466
  - recursive, 471–472
  - self-redefining, 471
  - string parameters, 470
- MacSFTP, 427
- MacSSH, 427
- Maintainability, writing for, 77–78
- man command (Unix/Linux), 417
- many prefetch hint, 130
- Map file, 51, 68–69
- Masking, 106, 159, 163–164
- Math coprocessor, 231
- MATRIX program, 376
- MAX-2 additions to PA-RISC, 384
- Maximum
  - floating-point, 242
  - parallel floating-point, 385
  - parallel integer, 380
- MAXIMUM program, 150–151
- McKinley code name, 10, 398
- md command (DOS), 416
- MDMX extensions (MIPS), 407
- mebi- (binary prefix), 5
- mega- (decimal prefix), 5
- Memory
  - as an array, 28
  - byte-addressable, 28
  - holding instructions and data, 27
  - information units, 27–29
  - word size, 28
- Memory direct addressing (see Addressing modes)
- Memory indirect addressing
  - (see Addressing modes)
- Memory-mapped I/O, 30
- Memory stacks, 190–194
- Merced code name, 10–11, 398
- Merge (see fpmerge instruction)
- Mi (symbol for binary prefix), 5
- mi (Macintosh text editor), 427
- Minimum
  - floating-point, 242
  - parallel floating-point, 385
  - parallel integer, 380
- minline-divide option (gcc), 341
- Minus sign character (see - character)
- MIMD computing systems, 378–379
- MIPS compilers, 340
- MISD computing systems, 378
- mix instruction, 380
- mkdir command (Unix/Linux), 416
- MMX instructions, 378, 407
- Modular programming, 200–203
- Modulo-scheduling of loops, 313–324
- MONEY macro, 473–476
- Moore’s law, 395
- more command, 416
- Motion video instructions, 407
- Motorola processors
  - 680x0 series, 231
  - PowerPC, 29, 296, 301
- mov instruction, 105
- mov pseudo-instruction
  - floating-point, 239
  - integer, 57, 91
- movl instruction, 103–104
- Multimedia instructions, 379–381, 407
- Multiple-issue effects, 300
- Multiplication
  - Booth’s algorithm, 170–173
  - floating-point, 240–241
  - integer, 92–93, 170–173, 251–252, 381–384
  - parallel instructions for 32-bits, 381–384
  - pppy2 instruction for 16-bits, 92–93
  - scalar product of vectors, 107–109

- shl instruction for powers of 2, 166
- shladd instruction for special cases, 90–91
- unsigned integers, 173
- Multiply-defined symbols, 67
- Multiway branching, 147–150
- mux instruction, 380
- mv command (Unix/Linux), 416

## N

- NaN (see Not a number)
- NAND function, 157
- @nat mnemonic for fclass, 247
- NaT bit, 310–311, 450
- Natural alignment
  - of floating-point data, 38, 40, 236–237
  - of integer data, 101–102
- NaTVal (see Not a thing value)
- nc completer (check load), 308–309
- ne compare relationship, 128
- @neg mnemonic for fclass, 247
- Negation
  - of a floating-point value, 239
  - of an integer value, 91
- Nesting
  - of conditionals, 146–147
  - of macros, 467
  - of parentheses in expressions, 63
- Newline character, 269–270
- nge compare relationship, 246
- ngt compare relationship, 246
- NiftyTelnet SSH, 427
- nle compare relationship, 246
- nlt compare relationship, 246
- nm command (Unix/Linux), 52–53, 69–70
- nolib\_inline option (Intel compilers), 343
- nop instructions, 66, 305–306
- NOR function, 157
- @norm mnemonic for fclass, 247
- Normalization, floating-point, 241
- Not a number, 38, 233
- Not a thing value, 235
- NOT operation, 92, 157
- NotePad, 427

- nt1 completer
  - floating-point load instructions, 237
  - integer load instructions, 102
  - line prefetch instruction, 327
  - semaphore instructions, 388–389
- nt2 completer
  - line prefetch instruction, 327
- nta completer
  - floating-point load instructions, 237
  - floating-point store instructions, 236
  - integer load instructions, 102
  - integer store instructions, 101
  - line prefetch instruction, 327
  - semaphore instructions, 388–389
- NUE (native user environment), 423–425
- NUL character, terminating a string, 60
- Number of addresses within instructions, 32–34
- Number conversion, 175–178
- Number systems
  - base, for machine language, 3
  - binary, 16–18
  - decimal, 16–17
  - floating-point, 37–41
  - hexadecimal, 17–18
  - IEEE floating-point, 37–41
  - integers, 16–20
  - octal, 16
  - one's complement, 18–19
  - sign and magnitude, 18–19
  - signed integers, 18–20
  - two's complement, 18–20
- Numeric ranges
  - floating-point numbers, 39
  - integers, 37
- NXOR function, 157
- nz test bit type, 160

## O

- .o file, 356
- o option (Unix/Linux), 53
- +O0 option (HP-UX compilers), 343
- O0 option (Linux compilers), 53, 342
- +O1 option (HP-UX compilers), 343



- O1 option (Linux compilers), 342
- +O2 option (HP-UX compilers), 343
- O2 option (Linux compilers), 342–343
- +O3 option (HP-UX compilers), 343
- O3 option (Linux compilers), 342–343
- +O4 option (HP-UX compilers), 343
- Object file, 49, 51, 57, 69
- Object file sections, 473
- Object libraries, 51
- Object-oriented languages, 3
- od command (Unix/Linux), 417
- +Ofast option (HP compilers), 343
- +Ofaster option (HP compilers), 343
- Offset
  - in branch instructions, 135–136
  - in displacement addressing, 117
  - location counter value as, 62
- +Olimit option (HP compilers), 343
- On-chip cache, 98–100, 399–400
- One-address instruction set, 33
- One's complement representation, 18–19
- Opcode, 31, 56–57, 84–86, 87–88
- Opcode extension fields, 87–88, 93–96
- Open routines, 466
- Opening a file, 279
- OpenVMS operating system, 415, 426
- Operand, 31
- Operand specifiers, 31, 55
- Operation code (see Opcode)
- Operator field, 55
- Operators
  - assembly language, 56–57
  - binary and unary, 62–63
- Optimization
  - and debugging, 369–370
  - dynamic, 369
  - enabling, 356–361
  - factors in a program, 325–328
  - inline functions, 366–369
  - inhibition of, 53, 345
  - levels for cc, aCC, and E90, 343
  - levels for ecc and efc, 342–343
  - levels for gcc and g77, 341–342
  - loop unrolling, 361–366
    - post-compilation, 369
    - profile-guided, 369
    - static, 369
- Options for Unix/Linux commands, 53
- or compare type, 161
- or instruction, 157–158
- OR logical function, 157
- or . andcm compare type, 161
- orcm compare type, 161
- ord compare relationship, 246
- Ordinateur (French for computer), 183
- Organization of a computer, 1
  - Os option (Linux compilers), 342
  - +Osize option (HP compilers), 343
- OUT instruction for I/O, 30
- out0–out7 stacked registers, 199, 205–207, 451
- Output dependency, 298
- Output stacked registers, 199
- Overflow
  - floating-point, 243
  - integer, 89–90
- P**
- PA-RISC processors, 10–11, 46
- pack instruction, 380
- Packing data not encouraged, 78
- padd instruction, 380
- Palindromes, 188
- Parallel logical compare instructions, 160–161
- Parallel operations
  - floating-point, 384–386
  - integer, 379–381
- Parameters
  - actual, 466–470
  - default values, 469–470
  - formal, 462, 466–470
  - keyword, 469–470
  - null, 469
  - positional, 468–469
- Parentheses, clarifying precedence, 63
- Pascal language as a 3GL, 3
- Passing arguments (see Argument passing)
- passwd command (Unix/Linux), 414

- Passwords, 414
- Path (for directory searches), 416
- `pavg` instruction, 380
- `pavgsub` instruction, 380
- PC (see Program counter)
- PCI bus, 30
- `pcmp` instruction, 380
- PDP-8 architecture, 23, 32
- PDP-11 architecture, 7, 9, 33–34, 83
- PDP-11 emulation, 34
- pebi- (binary prefix), 5
- Pentium CPU, 8
- Performance considerations, 293
- Performance for loops, 138–140
- Period character (see `.` character)
- `perror` function (C), 269, 288
- peta- (decimal prefix), 5
- PFE (Programmer's File Editor), 427
- `pfm` (previous frame marker), 205–207
- `ph` completer (branch instructions), 130
- Pi (symbol for binary prefix), 5
- Piano architecture and implementation, 2–3
- Pipeline bubbles, 294–300
- Pipeline depth, 294
- Pipeline hazards (see Hazards)
- Pipeline stages
  - for a generic processor, 294–295
  - Itanium 2 processor, 296–297
  - Itanium processor (initial implementation), 401–402
- Pipelined loops, 312–316
- Pipelining
  - bubbles, 295
  - floating-point, 296
  - hardware, 294–300
  - hazards, 297
  - software, 313
  - stalls, 295
  - superpipelining, 295
  - superscalar, 295
- PL/I language as a 3GL, 3
- Plus sign character (see `+` character)
- `pmax` instruction, 380
- `pmin` instruction, 380
- `pmopy2` instruction, 92–93, 380
- `pmopyshr2` instruction, 380
- Pointers, 103–104, 107, 422–423
- Polynomial evaluation, 241, 243–245
- Pop item from a stack, 190
- `popcnt` instruction, 380, 392
- Portability of programs, 50
- `@pos` mnemonic for `fclass`, 247
- Position-independent code, 218–219
- Positional coefficients, 16
- Positional parameters, 468–469
- Postcompilation optimization, 369
- Postincrement addressing (see Addressing)
- Postincrementing
  - with floating-point load, 236–238
  - with floating-point load pair, 238
  - with floating-point store, 235–236
  - with integer load, 102
  - with integer store, 101
- PowerPC (see Motorola processors)
- `pr` register, 212
- Precedence of operators, 63
- Precision, floating-point, 39–40, 240
- Precision completer, 240, 250
- Predicate (see Qualifying predicate)
- Predicate registers (see Registers)
- Predication, 131–135, 245–246, 316–323
- Prediction, branch (see Branch prediction)
- Prefetch hint, 130
- Prefetching (see Advanced loads, Line prefetch instructions, Speculative loads)
- Preprocessors, 201
- Preserved registers (see Registers)
- `print` command (gdb), 72–73
- `printenv` command, 417
- `printf` function (C), 269–270
- `@priunat` preservation, 212
- Privilege level, 205
- `.proc` directive, 57
- Procedural dependency (see Control dependency)
- Procedure frame, 191–192
- Procedures, 203
- Producer–consumer effects, 299–300
- Profile-guided optimization, 369

- progbits type, 473
  - Program counter, 27, 31
  - Program optimization, 293
  - Program segmentation, 200–203
  - Program size, 326
  - Programming environments
    - command-line, 50
    - tools, 52–53
  - Programs
    - APPROXPI, 256–260
    - BACKWARD, 181–183
    - booth (function), 215–216
    - chrget (function), 178–179
    - chrput (function), 178–179
    - COM\_C, 345
    - COM\_F, 345
    - DECNUM, 175–178
    - DECNUM2, 194–196
    - DECNUM3, 216–218
    - DOTCLOOP, 143–145
    - DOTCTOP, 316–321
    - DOTCTOP2, 321–324
    - DOTLOOP, 136–137
    - DOTPROD, 107–109, 113–116
    - fib function, 370–373
    - FIB1 function, 329–331
    - FIB2 function, 331–332
    - getput (encapsulated C routines), 178–179
    - HEXNUM, 96–98
    - HEXNUM2, 163–164
    - HORNER, 243–245
    - INLINE, 367
    - IO\_C, 179–180
    - MATRIX, 376
    - MAXIMUM, 150–151
    - MONEY (macro), 473–476
    - RANDC, 225
    - RANDF, 225–226
    - random (function), 222–224
    - random\_ (function) for FORTRAN, 222–224
    - SCANFILE, 280–284
    - SCANTERM, 270–273
    - SCANTEXT, 168–169
    - SORTINT, 284–288
    - SORTSTR, 273–277
    - SQUARES, 12–15, 71, 73–74, 95–96, 112–115
    - TESTFIB, 333–334
  - Prolog phase of loop, 314
  - .prologue directive, 145, 180–181, 212
  - Prologue section, 145, 210–213
  - Prompt, command-line (see Command line)
  - pr.rot (rotating predicate registers), 452
  - psad1 instruction, 380
  - Pseudo-operations, 56–57, 89, 91
  - Pseudo-random numbers, 220–221, 256–260
  - pshl instruction, 380
  - pshladd2 instruction, 380
  - pshr instruction, 380
  - pshradd2 instruction, 380
  - psp (previous stack pointer), 212
  - psub instruction, 380
  - Push item onto a stack, 190
  - put (ftp command), 415
  - putchar function (C), 178–179
  - puts function (C), 269–270
  - pwd command (Unix/Linux), 416
- Q**
- q command (adb), 72
  - @qnan mnemonic for fclass, 247
  - Qoption option (ecc), 68
  - qp (qualifying predicate), 55, 86
  - .quad directive, 60
  - Quad word, 35–36
  - Qualifying predicate, 55, 86, 315–316
  - quit command (gdb), 72
  - Quotation marks (see " character)
  - Quotient in division, 219–220
- R**
- r access mode, 279
  - :r command (adb), 72
  - r command (adb), 72
  - r completer, 380
  - ra command (adb), 72

- Radix control, 58–59
- RANDC, 225
- RANDF, 225–226
- random (function), 222–224
- random\_ (function) for FORTRAN, 222–224
- Random numbers, 220–226, 257–260
- RAR dependency, 307
- RAW dependency, 307
- rd command (DOS), 416
- readelf (Linux utility), 210–211
- real4 directive, 60
- real8 directive, 60
- Reciprocal
  - approximate, 253–254
  - approximate square root, 254–255
  - known, for division, 173–175
- Record structures, 104–105
- Recurrence relationship, 329
- Recursion, 327–329, 333–334, 370–373
- Recursive macros, 471–472
- Reduced instruction set computer (see Architecture, RISC)
- %REF function (FORTRAN), 210
- Reference, passing argument by, 209–210
- REG (register read pipeline stage), 296–297, 401–402
- Register-level programming, 27
- Register direct addressing (see Addressing modes)
- Register indirect addressing (see Addressing modes)
- Register indirect deferred addressing (see Addressing modes)
- Register-level programming, 27
- Register naming
  - gdb debugger, 71
  - Itanium assemblers, 14
- Register renaming, 298, 314
- Register stack. 198
- Register stack engine. 199–200
- Register windows (SPARC), 197–198
- Registers
  - adding to an architecture, 407
  - application, 105, 143, 453–455
  - for argument passing, 205–207
  - automatic, 450–451
  - banked, 200, 451
  - branch, 130, 452
  - constant, 450–451, 453
  - conventions for use, 204, 450–455
  - in the CPU, 27
  - cpuid, 409
  - floating-point, 27, 39–40, 212–213, 453
  - function returned values, 214
  - general, 450–451
  - global pointer, 103, 113, 451
  - integer, 27, 450–451
  - predicate, 126, 451–452
  - preserved, 204
  - read-only, 454
  - register rename base, 314
  - rotating, 199, 313–314
  - scratch, 204
  - special, 450–451
  - stack pointer, 190–192, 451
  - stacked, 198–199, 205–209, 451
  - state management, 455–457
  - system control, 457–458
  - system information, 457
- .regstk directive, 214
- rel completer
  - integer store instructions, 101
  - semaphore instructions, 388–390
- Relative addressing (see Addressing modes)
- Relative deferred addressing (see Addressing modes)
- Relative path (see Path)
- Relocatable symbols (see Symbols)
- Remainder, in division, 175, 219–220
- REN (rename registers pipeline stage), 296–297, 401–402
- ren command (DOS), 416
- Repeat blocks
  - indefinite, 462–463
  - simple, 460–461
- Representation of numbers
  - integers, 16–20
  - one’s complement, 18–19
  - sign and magnitude, 18–19

- signed integers, 18–20
  - two's complement, 18–20
- `.rept` directive, 461
- Reserved opcodes, 88
- `.restore` directive, 212
- `ret` branch type, 205
- `ret0-ret3` function return value registers, 214, 451
- Retiring of instructions, 295
- Return from procedure, 205–207
- RISC architecture (see Architecture)
- RISC instruction-level parallelism, 300–301
- `rm` command (Unix/Linux), 416
- `rmdir` command (Unix/Linux), 416
- ROT (rotate instructions pipeline stage), 296–297, 401–402
- Rotate using `shrp` instruction, 166
- Rotating stacked registers (see Registers)
- Rounding, 242–243
- `rp` register, 212, 452
- `rrb` register, 314, 456
- RSE (see Register stack engine)
- `run` command (gdb), 72

## S

- `.s` file type (see File types)
- `-S` option (Linux/Unix compilers), 345
- `:s` command (adb), 72
- `/s` display mode (debuggers), 72
- Saturation, clipping, 381
  - `.save` directive, 144–145, 180–181, 212–213
  - `.sbt1` directive, 64
- Scalar product of vectors, 107–109, 136–137, 143–145, 316–324
- `scanf` function (C), 269–270
- SCANFILE program, 280–284
- SCANTERM program, 270–273
- SCANTEXT program, 168–169
- Scope, frame and stack, 190–192
- Scratch area, stack, 191–192
- Scratch registers (see Registers)
- `.section` directive, 473
- Secure client software, 426–428
- Seed, 221

- Self-redefining macros, 471
- Semaphore instructions
  - as a category, 31
  - Itanium support, 386–390
  - ordering completers, 388–389
- Semicolon character (see `;` character)
- `set` command (gdb), 72
- `setf` instruction, 250
- Shared library functions, 203
- Shell program, 417
- Shift instructions, 165–166, 380
  - `shl` instruction, 165
  - `shladd` instruction, 90–91
  - `shladdp4` instruction, 107
  - `shr` instruction, 165
  - `shr.u` instruction, 165
  - `shrp` instruction, 166
- `si` command (see `stepi` command)
- SI units, 5
- Sign and magnitude representation
  - floating-point numbers, 37–41, 234–235
  - integers, 18–19
- Sign extension, 106
- Sign-manipulation instructions, 105–107
- Signed integers (see Integers)
- Significand, 37–41, 234–235
- Signum function, 154, 262
- SIMD computing systems, 378–379
- Simics, 421
  - `.single` directive, 60
- Single-stepping with debugger, 75–76
- SISD computing systems, 378
- Ski simulator, 420–421, 423–425
  - `.skip` directive, 57, 60
- Slash character (see `/` character)
- Slots for instructions in bundle, 85
- Smalltalk language as a 4GL, 3
- `@snan` mnemonic for `fclass`, 247
- `sof` field (cfm register), 199, 206–207
- Software-pipelined loops, 312–316
- `sol` field (cfm register), 199, 206–207
- `sor` field (cfm register), 199, 207
- Sorting
  - integers, 284–288

- strings, 273–277
- SORTINT program, 284–288
- SORTSTR program, 273–277
- Source files, comparing variants, 54
- Source program, 49
- sp register, 190–192, 451
- Space character, use in statements, 55–56
- SPARC register windows, 197–198
- Special registers (see Registers)
- Special values (IEEE), 38, 233
- Specifier field, 55
- Speculation
  - data, 307–310
  - control, 310–312
- Speculative load (see Load types)
- .spill directive, 213
- Spill form of store instruction
  - floating-point, 236
  - integer, 101
- Split issue, 304–305
- spnt branch whether hint, 130
- sptk branch whether hint, 130
- SQL language as a 4GL, 3
- Square brackets (see [ ] characters)
- Square root
  - floating-point software routines, 256
  - IEEE requirement, 253
- SQUARES program, 12–15, 71, 73–74, 95–96, 112–115
- scanf function (C), 290
- SSE extensions, 379, 407
- st instructions, 101, 390
- Stack-based instruction set, 32–33
- Stack pointer (see Registers)
- Stack unwinding (see Unwind tables)
- Stacked registers (see Registers)
- Stacks
  - addressing, 190–194
  - CISC architectures, 190
  - for argument passing, 208–209
  - Itanium architecture, 191–192
  - load/store architectures, 190–191
  - memory, 190–194
  - register, 196–200
  - user-defined, 192–194
- Stalls, data, (see Data stalls)
- Standard C library, 268
- Standard error, 268
- Standard input, 268
- Standard output, 268
- Starting address, 32
- State examination with debugger, 71
- State management by an architecture, 125–126
- State management registers (see Registers)
- Statement format, assembly language, 55–56
- Statement types, assembly language
  - control, 55, 64
  - declarative, 54
  - imperative, 54
- Static binding, 219
- Static initialization, 474
- Status field completer, 240
- stderr, 268
- stdin, 268
- <stdio.h> header file, 178, 268
- stdout, 268
- stepi command (gdb), 72
- Stepwise development, 50–53
- stf instructions, 235–236
- Stop (double semicolon), 15, 138–140, 460
- Storage allocation, 49, 60
- Store hints, 101, 236
- Store instructions
  - floating-point, 235–236
  - integer, 101
  - semaphore, 389–390
- Store types
  - floating-point, 235–236
  - integer, 101
- Streaming single-instruction, multiple data extensions (see SSE extensions)
- String of characters, 44
- string directive, 60
- String parameters, 470
- stringz directive, 60
- sub instruction, 88–89, 94
- Subroutines, 201–202
- Subtraction

- floating-point, 240
- integer, 88–89
- Subword instructions, 384
- Superpipelining, 295
- Superscalar parallelism, 35, 87, 295–296
- switch statement (C language), 149
- sxt instruction, 106
- Symbol table, 57, 64, 66–67
- Symbolic addresses, 56
- Symbolic assembler (see Assemblers)
- Symbolic debugger (see Debuggers)
- Symbols
  - absolute, 64
  - characters used in, 58
  - external, 67, 68
  - global, 57, 64, 67
  - multiply-defined error, 67
  - relocatable, 64
  - scope of, 59
  - undefined, 67–68
- System/36, 34
- System/38, 34
- System calls for I/O
  - Linux, 267
  - Unix, 267
- System control registers (see Registers)
- System information registers (see Registers)
- System libraries in linking process, 53

## T

- Tab character, use in statements, 56
- tbit instruction, 160
- tebi- (binary prefix), 5
- telnet program, use of, 413–414
- Template
  - assembler-selected, 303
  - codes, 302
  - in instruction bundle, 85, 302–307
  - manually assigned, 303
- Temporary label (see Label)
- tera- (decimal prefix), 5
- Tera Term, 427–428
- Terminal I/O, 268–270

- Terminators, line (see Line terminators)
- Terms in expressions, 63
- Test relationship, 310
- TESTFIB program, 333–334
- .text directive, 57
- Text editors, 414–415
- Text file I/O, 277–280
- Text files, comparing variants, 54
- Text mode (ftp), 415
- TextWrangler, 427
- Threads, 203
- Three-address instruction set, 33
- Ti (symbol for binary prefix), 5
- Tilde character (see ~ character)
- Time from operating system, 222–224
- .title directive, 64
- tnat instruction, 310
- tp register, 451
- trel completer (test bit instruction), 160
- trunc completer, 249
- Truncation
  - floating-point, 242–243, 249
  - integer, 166
- Two-address instruction set, 33
- Two-pass assembler, 66–67
- Two's complement representation, 18–20
- type command (DOS), 416
- .type directive, 180
- Typed language, 61
- Types of instructions (see Instruction classes)

## U

- uint64\_rem\_min\_lat (Intel open source), 220, 224
- UltraSparc, 384
- Unary operators, arithmetic and logical, 63
- Unbiased rounding, 242–243
- unc compare type, 146, 246
- Unconditional branch, 130–131
- Unconditional compare, 146
- Underflow, floating-point, 243
- Underscore character (see \_ character)
- Unformatted line I/O, 269–270, 279–280

Unicode, 42

Unix operating system

- commands, 416
- I/O software, 267
- line terminators, 415
- on-line documentation, 417

unord compare relationship, 246

@unorm mnemonic for `fclass`, 247

unpack instruction, 380

Unrolling of loops (see Loop unrolling)

Unsigned integers (see Integers)

Unwind information, 181, 210–211

Uploading files, 415

Upper case, converting to lower, 159

Upper case usage, 53

Usernames, 414

UTF-8, UTF-16, UTF-32, 42

## W

w access mode, 279

+w option (acc), 53, 369

/w display mode (gdb), 72

-w2 option (ecc), 53, 360

-Wa option (gcc), 67

-Wall option (gcc), 53, 360

WAR dependency, 307

Warnings from compilers, 360–361

watch command (gdb), 72

Watchpoints, 71, 74–75

WAW dependency, 307

Weights of digits, 16

wexit branch type, 130, 315

While loops (see Loops)

Width of instructions, 83–85

Windows

- 64-bit, 425
- client software, 427–428
- line terminators, 415

Wired or, 160

-Wl option (gcc), 68

WLD (word-line decide pipeline stage), 401–402

Word (data width), 35–36

Word counting, 168–169

.word directive, 60

-wp-ipo option (Intel compilers), 342

WRB (write-back pipeline stage), 296–297, 401–402

Writing programs, conventions for, 77–78

wtop branch type, 130, 315

## X

-x option (nm command), 53

/x display mode (debuggers), 72

x command (gdb), 72–73

X class of instruction, 86–88

XCHG instruction (IA-32 architecture), 387

xchg instruction, 388

xma instruction, 251

xmpy instruction, 251

xor instruction, 157–158

XOR logical function, 157

## Y

Y2K problem, 405



**Z**

-z option (gcc), 472

z test bit type, 160

@zero mnemonic for `fclass`, 247

Zero as IEEE number, 38, 40, 233

Zero-address instruction set, 32–33

Zero extension, 106–107

Zero latency, 131, 140

`zxt` instruction, 106–107