

INDEX

Numbers

- .2byte directive, 17
- 8-bit excess-127 exponent, 94
- .8byte directive, 17
- 13-bit immediate constants, 107
- 16-bit unsigned immediate
 - limitation, 107
- 16-bit values, 56
- 32-bit registers, 11
- 32-bit variables, 56–57
- 64-bit registers, 11
- 128-bit decimal output (conversion to string), 510
- 128-bit operations
 - logical AND, 465
 - NOT, 467
 - shift-left, 467–468, 709–711
 - XOR, 465
- 128-bit value comparisons, 446–450
- 192-bit addition, 442
- 192-bit shift-left operation, 469
- 256-bit comparisons, 449
- 256-bit logical OR operation, 466
- 256-bit subtraction, 446

A

- AARCH64, xxviii
- ABI (application binary interface),
 - 30–33
- absolute difference instructions,
 - 669–671
- absolute value comparisons, 690–691
- access, memory, 119, 135–137
 - page boundary, 128
 - violation, 181
- accessing data
 - at the end of an MMU page, 128
 - pointer data, 174
 - pushed on the stack, 165–166

- accessing elements of an array, 146
 - of a column-major array, 210
 - of a single-dimension array, 197
- accumulated errors in a floating-point calculation, 324–325
- accuracy, 324
- Acorn RISC Machine, xxvii
- activation records, 244–247
 - construction at runtime, 244
- addition
 - 192-bit, 442–443
 - bytes and half-words, 473
 - different-sized operands,
 - 472–475
 - extended-precision, 442
 - mixed-size, 473
 - pairwise, 664–666
- addition instructions, 28, 442–443, 659–660
 - add across vector, 647, 667
 - add with carry, 443
 - add with narrowing, 663–664
 - horizontal add, 665, 667–668
 - Neon, 647, 659–660, 666–668
 - pairwise, 665
 - saturating, 667
 - vertical, 664
- addresses, 11
 - alignment, 263
 - base, 146
 - expressions, 149
- addressing modes, 140–149
 - indirect-plus-offset, 143
 - for Neon load and store instructions, 633
 - post-indexed, 145
 - pre-indexed, 144–145
 - scaled-indexed, 146–149
 - scaled indirect, 143

- address space location randomization (ASLR), 23–25, 128
- adr instruction, 25, 153
- Advanced RISC Machine, xxvi
- .a files, 883
- aggregate data types, 186
- aliases (aka instruction mnemonics), 745
 - of registers, 22
- alignment
 - an address to some boundary, 263
 - bit strings, 705
 - data, 138–140
 - stack, 155
 - variable, 19–21
 - choosing alignment in memory, 140
- alignment directives, 6, 19–21, 139, 263, 780
- allocation
 - storage for strings, 803
 - variables in a data section, 138
- and instruction, 61, 704–705
 - Neon, 648
- AND operation, 58, 648
 - 128-bit, 465
 - truth table, 58
- aoaa.inc*
 - header file, 771
 - include file, 10, 26, 36
- Apple Silicon, xxvii
- application binary interface (ABI), 30–33
- application programming interfaces (APIs), 33
- architecture, CPU, 11
- args macro, 779
- arguments. *See* macros
- arithmetic
 - with different-sized operands, 472–475
 - expressions, 303–312
 - translating into assembly language, 303
 - floating-point, 322
 - infinite-precision, 323
 - logical systems, 314
 - mixed-size, 472
- operators
 - in CPP expressions, 745–746
 - precedence of, 308
 - real, 322
 - shift-right operation, 84
 - extended-precision, 472
 - SIMD operations, 659
- ARM64, xxviii
- armasm64 tool, xxix–xxx
- ARM memory access, 119
 - application binary interface, 31
 - pointer, 174
- ARM SVE (scalable vector extensions), 667
- ARMv8, xxviii
- arrays, 194–212
 - access
 - elements of a column-major array, 210
 - elements of a single-dimension array, 195
 - four-dimensional, 207
 - stepping through elements of an array, 144
 - three-dimensional, 207
 - two-dimensional row-major, 206
 - of arrays, 207–208
 - bubble sort, 198–203
 - column-major ordering, 204, 209–210
 - declarations, 195
 - indexing into, 146, 195
 - mapping to memory, 203–212
 - multidimensional, 203
 - packed, 731
 - row-major ordering, 204–209
 - of structs, 218
- ASCII character set, 55, 99–102
- ASCII groups, 100
- ASLR (address space location randomization), 23–25, 128
- asr instruction, 321
 - shift operator (Operand2), 109
 - for sign-extension, 473
- assembly/C hybrid programs, 8

- assembly language
 - instructions, 22
 - programming style, 228–230
 - source files
 - sections, 6
 - suffix, 4
 - standard entry sequence, 248
 - statement format, 229
 - string type, 802, 805
- assembly time
 - computing string length at, 189
 - constants, 744
- assignments, 304
- associativity, 307–308
- automatic variables, 250

B

- backspace character, 100
- .balign directive, 21, 139
- b.al (branch always) instruction, 76, 357
- base address, 146
- bash shell, xxxi
- bfm (bit field move) instruction,
 - 726–728
- bfxil (bit field extract and insert)
 - instruction, 727
- big-endian data organization, 133
- big-endian to little-endian conversion,
 - 134, 646
- binary-coded decimals (BCD), 54, 98–99
- binary conversions
 - even/odd—divide-by-two, 47
 - to hexadecimal, 49–50
- binary digits, 47
- binary fractions, 94
- binary logic, 46
- binary numbering system, 45–48
- binary point, 94
- binary-to-hexadecimal string
 - functions, 483
- b instruction, 75
- bitonic sorting, 694
- bits, 47, 53
 - arrays, 731
 - clearing, 61
 - bit fields, 727
 - vectors, 727
 - to zero, 59

- coalescing, 729–731
- data, 703
- extraction, 704, 713, 715
- fields, 85–93, 726–728
- first and last clear, 704, 734
- first and last set, 704, 734
- forcing, 61
- guard, 324
- insertion
 - into bit arrays, 732
 - bit sets into another bit
 - string, 706
 - bit strings, 719–726
 - if true, 648
 - in vectors, 648
- inversion, 61, 704, 709
- manipulation, 648, 703–704
- masking, 61, 704
- most and least significant, 48
- movement, 714
- Neon, 648
- numbering, 54
- offset, 704
- operations, 58–65
- packed arrays of, 731
- pattern search, 736
- in PSTATE, 93
- reversal, 712
- runs, 704
- scattering, 735
- searching for, 734, 736
- selecting, 648
- sets, 704
- setting, 59, 61, 704
- sign, 65
- starting position, 719
- testing, 715
 - byte, 54
 - instructions, 691,
 - 704–706, 710
- bit strings, 704
 - alignment, 705
 - arrays, 731
 - coalescing, 729
 - distributing, 729
 - extraction, 726
 - insertion, 719–726
 - merging, 735

- bit strings (*continued*)
 - packed arrays of, 731
 - packing and unpacking, 719
 - scattering bits from, 735
 - selectively inverting bits, 60
 - test for 1 bits, 717
- bitwise operations, 60–61
- bitwise select instructions, 648
- BMP (Basic Multilingual Plane),
 - Unicode, 847
- Bn* registers, 623
- Boolean constant representation, 313
- Boolean evaluation, 319
 - short-circuit, 380–384
- Boolean expressions, 313, 319
- Boolean logical systems, 314
- Boolean values, 53
- branch and link instructions, 29–30,
 - 230, 235, 284–285
 - indirect through register, 235
- branch avoidance via computation, 388
- branches, conditional, 77–78, 355
 - unsigned, 80
- branch instructions, 74–82, 357
 - indirect, 358
 - opposite, 82
 - unconditional, 77
- break statement in assembly language,
 - 420–421
- `.bss` section, 124–126
 - space in an executable file, 125
- bubble sort, 198–203
- build* shell script, 37
- bus error, 155, 286
- byte-addressable memory, 14
- `.byte` directive, 55
- byte macro, 780
- bytes, 53–55
- byte variable declarations, 55

C

- caches, 16
- callee and caller register
 - preservation, 239
- calling conventions, 258
- call tree, 242
- canonical equivalence, 849
- carriage returns, 100
- carry condition code, 14
- carry (C) flag, 296, 719
 - settings after `cmp`, 296
- case labels, 399
- case statement, 389
- `-c` (compile-only) command line
 - option, 38
- C/C++ preprocessor, 742
- C/C++ standard library, 5
 - calling functions, 33–36
 - function names in *aoaa.inc*, 774
 - math, 347
- cEL exception level, 14
- central processing unit (CPU), 11
- characters, 55, 99–103
 - combining, 852
 - acute accent characters, 849
 - constants, 101
 - data, 99
 - delimiter, 566
 - names, 848
 - strings, 187–194, 795
- char data type, 102
- C integer types, 441
- clearing bits, 59, 61, 704
 - bit fields, 727
 - vectors, 648
- clipping during saturation, 72
- `cmp` instruction, 78, 295–297
- `cmtst` instruction, 706
- code indentation, 229
- `.code` macro, 229, 784
- code movement, 387
- code points, 102, 847
- code sections, 121
 - in an assembly language program, 6
- code size optimization, 482
- code snippets, xxx
- column-major ordering, 204, 209–210
- command line interpreter, xxxi
- command line defines, 758
- common pointer problems, 180–186
- commutative operators, 311
- compare and branch instructions,
 - 425, 715
- comparisons
 - 128-bit value, 447
 - 256-bit, 449

- absolute value, 690–691
- dates, 92–93
- extended-precision, 446–447, 449
- floating-point, 336–343
- ordered, 97
- scalar, 688, 690
- strings, 824, 829
- unordered, 97, 336
- vector, 687–693
 - integer, 688
- compile-time language (CTL),
 - 741–742
 - constants, 744
 - expressions, 745
 - loops, 763
- complement method, 65
- complex arithmetic expressions, 307
- composite data types, 186–221
- conditional assembly, 760
- conditional branches, 77–78
 - signed and unsigned, 80
- conditional compilation, 746
 - debugging and testing code using, 748
- conditional execution, 74
- conditional instructions, 297–299, 711
 - branch, 77, 80
 - compare, 299, 314
 - and conjunction, 315–318
 - or disjunction, 318–319
 - encoding of Boolean expressions, 314
 - equates to define useful bit patterns, 786
 - flag settings after `cmp`, 295–296
 - increment, 298
 - inversion, 298
 - negation, 298
 - select/move, 298, 343
 - set, 299
- conditional macros in C++, 756
- conditional statements, 372
- condition codes, 14. *See also* flags
 - defines, 317–318
- conditioning inputs, 614
- conditions, 298, 318
- constant expressions, 150
- constant pool, 130
- constants
 - declaration, 21
 - floating-point, 97, 334
 - large, 111–113
 - literal, 21, 49
 - manifest, 21, 170
 - newline, 170
 - read-only variables as, 170
 - symbolic, 170
- constant values, 21
 - 13-bit immediate, 107
 - 64-bit immediate, 103
 - character literal, 101
 - floating-point, 97
 - hexadecimal literal, 49
 - manifest, 170
 - `n1` (newline), 170
 - symbolic, 170
 - using read-only data as constants, 170
- `.const` directive, 122
- continue statement, 422
- control bus, 11
- control characters, 100
- control structures, 355
- control transfer instructions, 74
- conversions
 - 128-bit decimal output to string, 510–516
 - ASCII digit to numeric value, 101
 - between upper- and lowercase, 100
 - binary
 - even/odd–divide-by-two, 47
 - to hexadecimal, 49–50
 - break statements into pure assembly, 421
 - continue statements into pure assembly language, 422
 - decimal
 - to binary, 47
 - extended-precision unsigned to string, 510–516
 - formatted to string, 517–528
 - signed to string, 509
 - string to integer, 566–578
 - unsigned to string, 495–509
 - endian, 134, 646

- conversions (*continued*)
 - fixed-point, 344, 684
 - floating-point, 683–686
 - to and from integer, 344, 683–684
 - to string, 529–565
 - forever statements into pure assembly, 419
 - for statements into pure assembly, 420
 - half-precision to single-precision, 685
 - hexadecimal
 - to binary, 49
 - digit to a character, 478
 - to strings, 478–495
 - string to numeric, 578–587
 - if statements to pure assembly, 371
 - integer
 - to floating-point, 344, 683–684
 - to string, 509–510
 - non-commutative arithmetic operators to assembly language, 310
 - numeric value to ASCII digit, 101
 - recursive, 495
 - repeat...until statements into pure assembly, 417
 - strings
 - to floating-point, 588–602
 - to integers, 566–587
 - to numeric, 566–602
 - coprocessors, 327
 - copying string data, 818
 - cos() function, 347
 - CPP (C/C++ preprocessor), 742
 - arithmetic expressions in, 745–746
 - compile-time constants, 744
 - conditional compilation, 746
 - debugging and testing code, 748
 - defined function, 746
 - defined symbol, checking, 746
 - #endif statement, 746
 - #error directive, 743
 - expressions
 - compile-time, 745
 - else, 746
 - if, 746–747
 - iteration, 757
 - macro arguments, 749
 - expansion, 750
 - separator, 750
 - macros, 749
 - composition, 752
 - conditional, 756
 - definition line limitation, 753
 - eval, 758
 - vs. Gas macros, 790
 - if_else, 756
 - iteration with, 757
 - recursive, 752
 - redefining, 759
 - undefining, 759
 - zero-argument macros, 749
 - processing __VA_ARGS__ argument lists, 757
 - text concatenation, 754
 - #undef statement, 759
 - variable argument lists, 751
 - #warning directive, 743
 - warnings vs. errors, 744
 - C preprocessor, 8
 - CPU (central processing unit), 11
 - Creative Commons 4.0 license, xxxii
 - CTL (compile-time language), 742
- ## D
- dangling pointer, 182
 - data alignment, 138–140
 - data declaration directives, 16–17, 122
 - label field in, 18
 - data representation, 45, 169
 - data sections, 122
 - variable allocation, 138
 - data types, composite, 186–221
 - date comparison, 92–93
 - DBCS (double byte character set), 846
 - decimal conversions, 47, 495–509, 566–578
 - decimal numbering system, 46
 - decisions, 371
 - declarations
 - arrays, 195
 - byte variables, 55
 - character variable, 102
 - constants, 21

- floating-point variables, 97–98
- pointers, 174–175
- variables in Gas, 16–18
- decoding ARM instructions, 104
- defined function in CPP, 746
- definite loops, 419
- deinterleaving data, 636, 642
- delimiter characters, 566
- denormalized values, 94, 96, 342
- descriptors, string, 189–190
- destructuring code, 386–388
- different-sized operands in arithmetic, 472–475
- digits, binary, 47
- directives, 17–18
 - alignment, 6, 19–21, 139, 263, 780
 - .bss, 124–126
 - reducing executable file size using, 125
 - .byte, 55
 - .code, 229, 784
 - .const, 122
 - .data, 16–17, 122
 - else, 761
 - end, 233, 761, 782–783
 - enter, 784
 - equate, 170–171
 - error, 743, 760
 - .exitm, 770
 - external, 864
 - .fill, 196
 - floating-point, 97–98
 - .global, 233, 864
 - if, 761
 - .include, 862
 - indefinite repeat, 764
 - leave, 784
 - .pool, 130–131, 334
 - proc, 233
 - public, 233
 - .purgem, 771
 - .rept....endr, 763–764
 - .req, 22
 - .rodata, 122–124, 170
 - .section, 122–124, 126
 - .set, 21, 170
 - .space, 196
 - .struct, 217
 - .text, 121–122
 - .warning, 760
 - wastr, 263, 783
- displacements, 132
- displaying error and warning messages, 743
- distributing bit strings, 729
- div128 algorithm, 511–516
- division, 294, 457–465, 679–680
 - extended-precision, 457
 - integer, 294
 - simulating div, 321
 - unsigned, 294
 - vector, 679–680
- DN (default NaN enable) bit in FPCR, 330
- Dn registers, 623
- domain conditioning, 614
- double-byte character sets (DBCS), 846
- .double directive, 97
- double loads and stores, 155
- double macro, 782
- double-precision floating-point declarations, 17
- double words (dwords), 53
- dtoStr (double word to string) function, 482
- dup instruction, 631
- duplicate include files/operations, preventing, 863
- .dword directive, 57
- dyadic operations, 58
- dynamic linking, 369
- dynamic memory allocation, 178
- dynamic range, 323
- dynamic string allocation, 803
- DZC (division by zero cumulative) flag in FPSR, 331, 335

E

- editor, 4
- effective address (EA), 146, 153
- effective memory address, 143
- element access in an array, 146
 - column-major, 210
 - single-dimension, 195
 - stepping through, 144

- else directives, 761
- else statements, 746
- end directives, 233, 761, 782–783
- endian byte organization, 133–135
- endian conversions, 134, 646
- end macros, 782–783
- enter macro, 784
- entry sequence, standard, 248
- eor instruction, 61, 709
 - exclusive–OR NOT, 709
 - Neon, 648
- equates, 21
 - directives, 170–171
 - public, 784
- error directives, 743, 760
- errors
 - bus, 155, 286
 - messages during assembly, 743
- even/odd–divide-by-two binary
 - conversion, 47
- exclusive-OR (XOR) operation,
 - 58–60
 - 128-bit, 467
 - vectors, 648
- executable file size, reducing using `.bss`
 - directive, 125
- `.exitm` directive, 770
- exploits, 129
- exponents
 - biased, 95
 - excess-127, 94–95
 - excess-1,023, 95
 - floating-point, 95
- expressions, 307
 - addresses, 149
 - arithmetic, 303, 307
 - Boolean, 312–319
 - CPP, 745–747
 - in an `#if` statement, 747
 - and temporary values, 311
- extended multiplication, 672
- extended-precision
 - arithmetic, 441
 - addition, 442
 - division, 457
 - multiplication, 450
 - subtraction, 445–446
 - comparisons, 446–450
- conversions
 - string-to-numeric, 566, 578
 - unsigned decimal to string
 - conversion, 510–516
- hexadecimal output, 494
- I/O, 478
 - formatted I/O, 517
- negation, 465
- shift operations, 467
 - arithmetic shift-right, 472
 - logical shift-right, 472
 - shift-left, 467
- extend operators, 110
 - Operand2 extension
 - operators, 474
- external directives, 864
- external symbols, 6, 865
- extract instruction, 643, 713, 715
- extraction
 - bits, 704, 713, 715
 - bit strings, 726–727, 735

F

- false, representation of, 313
- false precision, 325
- `fcvt` instruction, 343–344
- field, 213
- file I/O (input/output) functions, 901, 907–915
- files* library, 901
- `.fill` directive, 196
- first clear bit, 704, 734
- first set bit, 704, 734
- fixed-point conversions, 344, 684
- flags, 28. *See also* condition codes
 - carry (C), 296, 719
 - `cmp` instruction effect on, 295–296
 - in FPSR, 331–332, 335
 - negative (N), 295, 718
 - overflow (V), 296, 719
 - sign (N), 295, 718
 - zero (Z), 295, 716
- floating-point
 - calculations, 322
 - accumulated errors in, 324
 - vector multiply, 671
 - comparisons, 336–343
 - absolute value, 690–691

- Neon, 689–691
 - scalar, 690
 - vector, 689
- condition code flags, 331–332, 335
- constants, 97, 334
- conversions, 683–686
 - double- to single-precision, 685
 - to and from integers, 344, 683
 - to and from string, 529–565, 588–602
- data movement instructions, 332
- declarations, 97–98
 - double-precision, 17
- directives, 97–98
- exponents, 95
- formats, 93–96
 - single-precision, 17, 94–95
- immediate instructions, 630
- implied bit, 94
- infinity representation, 97
- normalized, 96
- operands, immediate, 334
- parameters, 346
- registers, 346
 - control, 328, 330
 - status, 328
- string output, 529
- underflow, 326
- values
 - implied bit in, 94
 - rounding, 686
 - subnormal, 342. *See also* denormalized values
- `fmov` instruction, 333
 - with immediate operands, 334
- forcing a 0 result, 59
- forcing bits to 0 or 1, 61
- `forever/endfor` loops, 418
- `for` loops, 419
- formatted decimal to string
 - conversions, 517–528
- four-dimensional array access, 207
- FPU (floating-point unit), 327
 - data movement instructions, 332
- frame pointer (FP) register, 13, 246
- `free()` function, 120, 178, 182
- functional macros, 749
- function results, 32

- functions. *See* procedures
- FZ16 (flush to zero, half-precision) bit
 - in FPCR, 330

G

- Gas, 3
 - literal constants, 49
 - macros, 765
 - vs. CPP macros, 790
 - variables, 16–18
- Gas/GCC hybrid programs, 8
- GCC, 7
- general protection fault, 121
- general-purpose registers, 11
- generating errors and warnings during
 - assembly, 760
- `getErrno` macro, 785
- `.global` directive, 233, 864
- global names, 6
- global variables, 300
- glyphs, 848
- GNU assembler, 3
- `goto` macro, 785
- granularity (MMU page), 127
- grapheme cluster, 848–849
- guard digits or bits, 324

H

- half-precision to single-precision
 - conversion, 685
- half-word data type (hwords), 53, 55–56
 - variables, 56
- hardware stack, 155
- header files, 863
 - `aoaa.inc`, 771
 - multiple inclusion prevention, 772, 863
- heaps, 120
 - storage of strings, 803
- “Hello, world!” program, 33, 40
 - stand-alone version, 899
- hexadecimal
 - conversions
 - to binary, 49–50
 - digit to character, 478
 - to string, 478–495
 - string to numeric, 578–587

- hexadecimal (*continued*)
 - literal constants, 49
 - numbering system, 45, 48–50, 54
 - output, extended-precision, 494
 - high-level language (HLL) control structures, 355
 - Hn (16-bit halfword) registers, 623
 - HO (high-order) bit, 48, 65–66, 72, 495, 651
 - of the mantissa, 94, 96
 - HO byte, 56–57, 133
 - in a half word, 55–56, 112
 - in a word, 56–57
 - HO half word in a word, 57
 - HO nibble, 55–57, 98
 - in a byte, 55
 - in a half word, 56
 - in a word, 57
 - horizontal operations, 646
 - addition, 665, 667–668
 - minimum and maximum values, 682
 - hword, 780
 - .hword directive, 17, 56
 - hybrid programs, 8
- I**
- i64toStr function, 509–510
 - i64toStrSize, 522
 - identifiers, 6
 - idioms (aka machine idiosyncrasies), 319
 - IEEE-754
 - infinity representation, 97
 - not-a-number, 97
 - standard floating-point formats, 93–94
 - .if directives, 761
 - if statements, 371
 - in CPP, 746–747
 - if...else, 372
 - CPP macro, 756
 - rearranging expressions to improve performance, 385
 - immediate constants
 - 13-bit, 107
 - 64-bit, 103
 - in vector compare, 688–689
 - and vector registers, 688–689
 - immediate floating-point operands, 334
 - implied bits, 94
 - improving loop performance, 428
 - include directives, 862
 - #include vs. .include, 10, 772
 - include files, 10
 - aaaa.inc, 10, 26, 36, 771
 - nested, 862
 - preventing duplicate, 863
 - inclusive-OR operation, 59
 - increment conditionally, 298
 - indefinite repeat directives, 764
 - expanding vararg lists, 767
 - indentation, code, 229
 - indirect branch instructions, 235, 358
 - indirect jump tables, 391
 - indirect-plus-offset addressing mode, 143
 - scaled, 143–144
 - infinite loops, 415, 418
 - in assembly language, 419
 - infinite-precision arithmetic, 323
 - infinity representation, 97
 - input conditioning, 614
 - input/output (I/O) devices, 11
 - input/output program, 901
 - insertion
 - bit set into another bit string, 706
 - bits into bit arrays, 732
 - bits in vectors, 648
 - bit strings into other bit strings, 719–726
 - data into lanes of a vector register, 626
 - instructions, 626
 - instructions. *See also* conditional instructions
 - absolute difference, 669–671
 - adc, 443
 - adcs, 443
 - add, 28, 442
 - Neon, 660
 - addhn, 660
 - addhn2, 660
 - addp, 660
 - adds, 28, 443
 - addv, 647, 667
 - adr, 25, 153
 - adrp, 25, 153

- and, 61, 704–705
 - Neon, 648
- ands, 705
- asr, 84–85, 321
- assembly language, 22
- b, 75
- b.al, 76, 357
- bcc, 77
- bcs, 77
- beq, 77, 80
- bfm, 726, 728
- bfxil, 727
- bge, 80
- bgt, 80
- bhi, 80
- bhs, 80
- bic, 704
 - Neon, 648
- bif, 648
- bit, 648
- bl, 29, 230, 235
- ble, 80
- blo, 80
- blr, 29, 284
- bls, 80
- blt, 80
- bmi, 77
- bne, 77, 80
- bnge, 82
- bnge (macro), 787
- bngt, 82
- bngt (macro), 787
- bnhi, 82
- bnhi (macro), 787
- bnhs, 82
- bnhs (macro), 787
- bnle, 82
- bnle (macro), 787
- bnlo, 82
- bnlo (macro), 787
- bnls, 82
- bnls (macro), 787
- bnlt, 82
- bnlt (macro), 787
- bpl, 77
- br, 235, 358
- branch, 74–82
- bsl, 648
- bvc, 77
- bvs, 77
- cbnz, 425, 715
- cbz, 425, 715
- ccmn, 299
- ccmp, 299, 314
- cmeq, 688
- cmge, 688
- cmgt, 688
- cmhi, 688
- cmhs, 688
- cmn, 297
- cmp, 295
- cmtst, 706
- conditional, 297–299, 711
- cset, 298
- cset, 298, 711
- csetm, 298, 711
- csinc, 298
- csinv, 298
- csneg, 298
- data movement, floating-point, 332
- decoding, 104
- dup, 631
- eon, 709
- eor, 61, 709
 - Neon, 648
- ext, 643
- extr, 715
- fabd, 670
- facge, 691
- facgt, 691
- fadd, 660
- faddp, 660
- fcmeq, 689
- fcmeq, 689
- fcmg, 689
- fcvt, 343
- fcvtas, 683
- fcvtau, 684
- fcvtl, 685
- fcvtl2, 685
- fcvtms, 683
- fcvtmu, 684
- fcvtn, 685
- fcvtns, 683
- fcvtn2, 685

instructions (*continued*)

fcvt_{nu}, 684
fcvt_{ps}, 683
fcvt_{pu}, 684
fcvt_{xn}, 685
fcvt_{xn2}, 685
fcvt_{zs}, 683
fcvt_{zu}, 684
fmax, 681
fmax_{nm}, 681
fmax_{nmp}, 682
fmax_{nmv}, 683
fmax_p, 682
fmax_v, 683
fmin, 681
fmin_{nm}, 681
fmin_{nmp}, 682
fmin_{nmv}, 683
fmin_p, 682
fmin_v, 683
fmla, 673, 676–678
fmls, 673, 677–678
fmov, 333
fmul, 672, 676, 678
fmul_x, 672, 677, 678
FPU data movement, 332
frecps, 679
frinta, 686
frinti, 686
frint_m, 686
frint_n, 686
frint_p, 686
frint_x, 686
frint_z, 686
frsq_{rte}, 687
frsq_{rts}, 687
fsqrt, 687
fsub, 668
goto, 785
insert, 626
ld1 through ld4, 632–638
ldnp, 333
ldp, 155, 333
ldr, 23, 27, 73, 130
ldrb, 144, 474
ldrh, 144, 474
ldr_{sb}, 474
ldr_{sh}, 474
ldr_{sw}, 474
ldur, 144, 332
lsl, 82, 320, 714
lsr, 83, 321, 714
mla, 671, 675
mlal2, 676
mls, 671, 676
mov, 27
movn, 113
movz, 112
mrs, 331, 716
msr, 331
mul, 211
 Neon, 671, 675
mvn, 61, 62, 704
mvni, 630
not, 61, 62
 Neon, 648
orn, 704
 Neon, 648
orr, 61, 704
 Neon, 648
raddhn, 660
raddhn2, 660
rbit, 712
ret, 29, 230, 235
rev, 135
rol, 470
ror, 85, 715
rsubhn, 669
rsubhn2, 669
sabal, 670
sabal2, 670
saba, 670
sabd, 669
sabd1, 670
saddalp, 660
sadd1, 660
sadd12, 660
saddlp, 660, 666
saddw, 660
saddw2, 660
sdiv, 457
shadd, 660
shl, 649
shsub, 669
smadd1, 453
smax, 681

smaxp, 682
 smaxv, 683
 smin, 681
 sminp, 682
 sminv, 683
 smlal, 672, 676
 smlal2, 672
 smls, 676
 smlsl, 672
 smlsl2, 672, 676
 smnegl, 452
 smsubl, 453
 smul, 450
 smulh, 453
 smull, 452
 Neon, 672
 smull2, 672, 676
 sqadd, 660
 sqdmlal, 673
 sqdmlal2, 673
 sqdmlsl, 673
 sqdmlsl2, 673
 sqdmulh, 674–675
 sqdmull, 673
 sqdmull2, 673
 sqrdmulh, 674
 sqsub, 668
 square root, 686–687
 srhadd, 660
 sri, 710
 ssubl, 668
 ssubl2, 668
 ssubw, 668
 ssubw2, 669
 st1 through st4, 632–638
 stnp, 333
 store, 26, 144, 332–333, 632–638
 stp, 156, 333
 str, 23, 332
 stur, 144, 332
 sub, 28
 Neon, 668
 subhn, 669
 subhn2, 669
 subs, 28
 svc, 892–894
 sxtb, 474
 sxth, 474
 sxtw, 474
 tbnz, 715
 tbz, 715
 trn1 and trn2, 639
 tst, 704
 uaba, 670
 uabal, 670
 uabd, 669
 uabd1, 670
 uaddalp, 660, 666
 uabd12, 670
 uadd1, 660
 uadd1p, 660, 665
 uadd12, 660
 uaddw, 660
 uaddw2, 660
 ubfiz, 714
 ubfm, 714
 ubfx, 713
 udiv, 457
 uhadd, 660
 uhsub, 669
 umadd1, 453
 umax, 681
 umaxp, 682
 umaxv, 683
 umin, 681
 uminp, 682
 uminv, 683
 umlal, 672, 676
 umlal2, 672, 676
 umlsl, 672
 umlsl2, 672, 676
 umnegl, 452
 umsub1, 453
 umul, 450
 umulh, 453
 umull, 452
 Neon, 672
 umull2, 672, 676
 uqadd, 660
 uqsub, 668
 urhadd, 660
 usub1, 668
 usub12, 668
 usubw, 668
 usubw2, 668
 uxtb, 474

- instructions (*continued*)
 - uxth, 474
 - uxtw, 474
 - uzp1 and uzp2, 642
 - xor, 704
 - zip1 and zip2, 641
- integer
 - comparisons, 688
 - conversion
 - to floating-point, 344, 683–684
 - to string, 509–510
 - division, 294
 - types in C, 441
- integral rounding, 686
- interleave load/store instruction
 - addressing modes, 633
- interleaving and deinterleaving
 - data, 635–636, 642
 - registers, 639
- invert conditionally, 298
- inverting bits, 61, 704
 - in a bit set, 709
 - in a bit string, 60
- invoking a macro inside another
 - macro, 752
- IOC (invalid operation cumulative) bit
 - in FPSR, 331
- iSize function, 517
- iteration with CPP macros, 757
- itoStrSize function, 522
- IXC (inexact cumulative) bit in
 - FPSR, 332

J

- jumps, indirect, 358
- jump tables
 - indirect, 391
 - with noncontiguous entries, 392
 - sparse, 399–402
 - with vector comparison, 692

K

- KCS floating-point standard, 93

L

- labels, statement, 356
- lanes in vector registers, 625
 - rearranging, 641
- large constants, 111–113
- large parameter objects, 273
- last clear bit, 704, 734
- last-in, first-out (LIFO) data
 - structure, 161
- last set bit, 734
- Latin-1 character set, 849
- ld (loader/linker) program, 7
- lea macro, 142, 153, 356
- least significant bit, 48, 54
- leave macro, 784
- left-associative operators, 308
- left-shift operation, 82
- length-prefixed strings, 188–189, 796
- library files, 883
 - program size and, 886
- lifetime of a variable, 250
- line feed, 100
- linking, dynamic, 369
- link register (LR), 13, 29, 235
- listings, xxx
- literal constants, 21
 - hexadecimal, 49
- little-endian data organization, 133
- little-endian to big-endian conversion,
 - 134, 646
- load and store architecture, 23
- load and store instructions, 23, 27, 73,
 - 130, 332–334
 - double, 155–156
 - interleave addressing modes, 633
 - Neon, 632–638
- loading floating-point constants into an
 - FPU register, 334
- LO (low-order) bit, 48
 - of the mantissa, 338
- LO byte, 56, 133
 - in a half word, 56
 - in a word, 57
- local labels, 234–235
- locals macro, 779
- local variables, 250, 300
- location counter, 18, 125, 131
 - . operator, 132, 171, 189
- logic, binary, 46
- logical operations, 58–60, 313
 - on bits, 58–65
 - Neon, 647

- shift-left, 82
 - shift-right, 84
 - extended-precision, 472
 - vectors, 648
- logical systems, 314
- LO half word in a word, 57
- LO nibble, 55, 101
 - in a byte, 55
 - in a half word, 56
 - in a word, 57
- lookup tables, 644
 - creating, 615
- loops, 415–434
 - control variables, 416, 426
 - definite, 419
 - infinite, 418
 - performance improvements, 428
 - register usage, 426
 - unraveling, 432, 763

M

- machine code encoding, 103–110
- machine idioms (aka idiosyncrasies), 319
- machine state, saving, 237
- macros, 741, 765–771. *See also under* CPP
 - arguments, 749
 - expansion, 750
 - separator, 750
 - creating inside other macros, 787
 - functional, 749
 - Gas, 765
 - invocations inside other macros, 752
 - opposite branch, 787
 - parameters, 765–766
 - expansion, 766
 - with string constants, 768
 - recursive, 752, 769
 - writing, 787
- magic numbers, 170
- makefiles, 37
 - syntax, 876
- malloc() function, 120, 178
 - and memory alignment, 804
- manifest constants, 21, 170
- manipulating bits, 648
 - in memory, 703–704
 - in PSTATE, 93
- mantissa, 94

- mask bits, 704
- masking, 61
- math library in C stdlib, 351
- matrix transposition, 639
- maximum values, 681–683
- memory, 11
 - access, 119, 135–137
 - page boundary, 128
 - unaligned, 16
 - violation, 181
 - addresses, 11, 19, 143
 - addressing modes, 120, 140–149
 - alignment, 804
 - allocation, 178, 803
 - byte addressable, 14
 - choosing variable alignment in
 - memory, 140
 - free() function, 178, 182
 - leaks, 183
 - malloc() function, 178
 - manipulating bits in, 703
 - mapping arrays to, 203–212
 - MMU pages, 24, 127
 - accessing data at the end
 - of, 128
 - boundaries, 128
 - faults when reading
 - memory, 797
 - granularity, 127
 - organization, 120–126
 - multi-byte data
 - organization, 133
 - performance, 481
 - pointer problems, 180–186
 - reading from memory on a 16-bit
 - CPU, 136–137
 - read operation, 15
 - stack, 120, 155
 - removing data from, 163–165
 - subsystem, 14–16
 - variables, 19, 299
 - declarations, 16–18
 - write operation, 15
- memory-management unit, 24, 127
- mergeBits function, 725–726
- minimal procedure, 235
- minimum values, 681–683

- misaligned data and the system
 - cache, 140
 - mixed-size arithmetic, 472
 - addition, 473
 - MMU (memory-management unit)
 - pages, 24, 127
 - accessing data at the end of, 128
 - boundaries, 128
 - faults when reading memory, 797
 - granularity, 127
 - modules in source code, xxx
 - modulo, 294–295
 - monadic operators, 60
 - most significant bit, 48, 54
 - move, conditionally, 298
 - move instructions, 27, 62, 112–113, 331, 716
 - Neon, 626–630
 - moving bits, 714
 - moving data between registers, 625–626
 - multi-byte data organization in memory, 133
 - multidimensional arrays, 203–212
 - multiple inclusion of header file, prevention of, 772
 - multiple instructions on a single source line, 230
 - multiple lines per statement, 230
 - multiplication instructions, 211, 450–457
 - extended, 672
 - floating-point, 672
 - multiply and accumulate, 671
 - multiply and subtract, 671
 - Neon, 671–678
 - of a register value by ten, 320
 - saturation, 673–675
 - signed, 450
 - unsigned, 211, 450
 - vector, 671–678
 - by a vector element, 678
 - multiway branch after vector comparison, 692
- N**
- NaN (not-a-number) values, 97, 330, 335
 - narrowing shift-right instructions, 655
 - N (negative/sign) condition code, 14
 - negation, 28, 465
 - conditionally, 298
 - extended-precision, 465
 - large values, 465
 - negative (N) flag, 295, 718
 - Neon instructions, 621
 - absolute difference, 669–671
 - addition, 659–660, 666–668
 - cmtst, 706
 - comparison
 - integer, 688
 - floating-point, 689–691
 - scalar, 688–689
 - signed, 689
 - conversion
 - between floating-point formats, 685
 - floating-point to integer, 683–684
 - division, 679
 - dup, 631
 - ext, 643
 - insert, 626
 - load and store, 632–638
 - logical, 648
 - minimum and maximum, 681–683
 - move, 626–630
 - multiplication, 671–678
 - rounding floating-point to integral values, 686
 - shift, 649, 710
 - square root, 687
 - subtraction, 668–669
 - transpose, 639
 - unzip, 642
 - zip, 641
 - Neon operations, 327
 - arithmetic, 659
 - logical, 647
 - shift, 649
 - nested include files, 862
 - nibble data type, 54
 - nl (newline) constant, 170
 - noncontiguous jump table entries, 392
 - nonvolatile registers, 31, 300, 346
 - normal forms, 849–850

- not-a-number (NaN) values, 97, 330, 335
- not instruction, 61–62
 - Neon, 648
- NOT operations, 60, 648
 - 128-bit, 467
- NUL character, 189, 263
- NULL pointer references, 121
- numbering systems, 46–54
 - binary, 45–46
 - decimal, 46
 - hexadecimal, 45, 48–50, 54
 - positional, 46
 - radix, 48
 - two's complement, 56, 65
- numbers, signed and unsigned, 65–70
- numeric conversion, 478–602
- numeric representation, 50–53
- NZCV register, 93

O

- OFC (overflow cumulative) bit in FPSR, 331
- offsets, 132, 704
- one's complement format, 94
- Operand2, 106–110, 474
 - allowable fields, 107
 - encoding, 106–110
 - extension operators, 110
 - shift operators, 109
- operation code, 104
- operations
 - arithmetic, 659
 - precedence, 308
 - bit, 58–65
 - on different-sized operands, 472–475
 - dyadic, 58
 - extension, 71
 - horizontal, 646
 - logical, 58–60, 313
 - AND, 58
 - OR, 59
 - NOT, 60, 648
 - push and pop, 155–158
 - reducing on a vector, 647
 - rotate, 85
 - saturation, 72

- shifts, 82–84
- sign contraction, 72
- stack, 120
- string, 795
- two's complement, 66
- vertical, 646
- write memory, 15
- XOR, 59–60
- operators
 - \@, 770
 - ., 132, 171, 189
 - commutative, 311
 - extend, 110
 - left- and right-associative, 308
 - monadic, 60
 - precedence, 308
 - shift (Operand2), 109
- opposite branch instructions, 82
 - macros, 787
- opposite condition defines, 318
- ordered comparisons, 97
- OR operation, 59
 - 256-bit, 466
 - vectors, 648
- OS function call, 892
- overflow condition code, 14
- overflow (V) flag, 296, 719

P

- .p2align directive, 263
- packed data, 85–93
- packing bit strings, 719
- page boundary memory access, 128
- pages, MMU, 24, 127
 - accessing data at the end of, 128
 - boundaries, 128
 - faults when reading memory, 797
 - granularity, 127
- pairwise addition, 664–666
- pairwise minimum and maximum values, 681–682
- parameters
 - expansion in macros, 766
 - of strings, 768
 - floating-point, 346
 - large objects as, 273
 - reference, 256
 - value, 255

- parameters (*continued*)
 - variable-length, 263
 - variadic, 42
- passing parameters, 32
 - by reference, 256
 - efficiency, 258
 - by value, 255
- PC (program counter) register, 13
- PC-relative addressing, 24
- performance analyzers, 479
- performance improvement
 - of loops, 428
 - rearranging expressions in if statements, 385
- permutation of data in vectors, 645
- pie (position-independent executable)
 - command line option, 38
- pointers
 - accessing data, 174
 - in assembly language, 174–186
 - constants, 141, 175
 - dangling, 182
 - declarations, 175
 - invalid address value in, 181
 - problems, 180–186
 - to strings, 190
 - type checking, 183
 - uninitialized, 180
 - wild, 182
- pointer variables, 178
- pool directive, 130–131, 334
- .pool section, 142
- pop operations, 157–158
- positional numbering systems, 46
- position-independent code, 128–130
- position-independent executables (PIE), 23, 128
 - procedural, 284
- post-indexed addressing mode, 145
- precedence rules, 308
 - of arithmetic operators, 308
- precision, false, 325
- pre-indexed addressing mode, 144–145
- preprocessors, 8, 742
- preserving registers
 - callee and caller, 239
 - in loops, 427
 - on the stack, 159
 - volatile, 907
- printf() function, 33
- proc directive, 233
- procedures, 230–234
 - in ARM assembly, 6
 - invocation, 230
 - minimal, 235
 - range of a function, 612
 - pointers, 284
- profilers, 479
- program counter (PC) register, 13
- program-counter-relative addressing, 24
- program listings, xxx
- programming in the large, 862
- programming languages
 - C, 441
 - FORTRAN, 405
- program size and object/library files, 886
- PSTATE (processor state) register, 13
 - manipulation, 93
- public equate, 784
- public symbols, 233
- pure assembly applications, 890
- .purgem directive, 771
- push operations, 155–157

Q

- QC (saturation cumulative) bit in FPSR, 332
- Q_n registers, 624
- qto5tr function, 494
- quad words (qwords), 53
- Quicksort, 278
- quiet NaN, 335
- .qword directive, 57, 781

R

- radix, 48
- range of a function, 612
- Raspberry Pi, xxvii–xxix, 481, 488, 728
- rbit instruction, 712
- read, memory, 15
 - reading 16 bits at an odd address, 137
 - reading a byte on a 16-bit CPU, 136

- read-only data, 120
 - sections, 122–124
 - read-only variables as constants, 170
 - real arithmetic, 322
 - reciprocals, 679, 687
 - recursive conversion, 495
 - recursive header files, 863
 - recursive macros
 - CPP, 752
 - Gas, 769
 - reducing code size, 482
 - reference parameters, 256
 - register-indirect jump instruction, 358
 - registers, 11, 14–16
 - 32-bit, 11
 - 64-bit, 11
 - aliases, 22
 - floating-point, 346
 - frame pointer, 13, 246
 - general-purpose, 11
 - interleaving and deinterleaving, 639–642
 - link, 13, 29, 235
 - moving data between, 625–626
 - nonvolatile, 31, 300, 346
 - NZCV, 93
 - preservation, 159, 427
 - callee and caller, 239
 - in loops, 427
 - on the stack, 159
 - volatile, 907
 - program counter, 13
 - PSTATE, 13
 - special-purpose, 11
 - stack pointer, 13, 146, 155
 - usage and loops, 426
 - as variables, 299
 - vector, 623
 - volatile, 31, 300, 346
 - Xn, 146
 - XZR, 146
 - zeroing out bits in, 727
 - remainder, 294–295
 - removing data from the stack, 163–165
 - repeat...until loops, 417–418
 - representation, 45
 - Boolean constant, 313
 - numeric, 50–53
 - .rept...endr statements, 763
 - .req directive, 22
 - ret instruction, 29, 230, 235
 - return addresses, 13
 - reversing bits, 712
 - rev instruction, 135
 - right-associative operators, 308
 - right shifts, 83
 - arithmetic, 84
 - RISC (reduced instruction set computer), xxvi
 - .rodata (read-only data) objects
 - declaration vs. value, 170
 - directive, 170
 - section, 122–124
 - rol instruction simulation using ror, 470
 - rotate operations, 85
 - through carry left, 468
 - through carry right, 472
 - rotate-left, 85
 - vector, 710
 - rotate-right, 85, 715
 - Operand2, 109
 - vector, 711
 - rounding, 345
 - during floating-point calculations, 324
 - floating-point values, 686
 - integral, 686
 - rounding mode control, 330–331
 - row-major ordering, 204–209
 - running an assembly language program, 7–10
 - run of 0 bits, 704
 - runtime language, 742
- ## S
- salign macro, 780
 - saturation (QC) bit in FPSR, 332
 - saturation operations, 72
 - multiplication and double, 673–675
 - shift-left, 650
 - shift-right with narrowing instructions, 655
 - vector saturating accumulate, 666
 - scalable vector extensions (SVE), 667
 - scalar floating-point comparisons, 690

- scalar instructions, 621
 - satürating addition, 667
- scalars, 625
- scaled-indexed addressing mode, 146–149
- scaled indirect addressing mode, 143
- scaling factors, 147
- scope of a variable, 250, 865
- searching
 - for a bit, 734
 - for a bit pattern, 736
 - for the first or last set bit, 734
- sections
 - in an assembly language source file, 6–7
 - .bss, 124–126
 - code, 121
 - .data, 122
 - .pool, 130, 142
 - read-only, 122–124
 - .section directive, 122–124, 126
 - flags in, 126
 - .text, 121–122
- security, 23
- segmentation fault, 121, 181
- selecting bits, 648
- separate assembly, 866
- separate compilation, 866
- .set directive, 21, 170
- setting bits, 704
 - to 0, 59
 - to 1, 61
- shared libraries, 23
- shell interpreter, xxxi
- shift-and-insert instructions, 652
- shift-and-rotate instructions, 704, 709–711
- shift operations, 82–85
 - Neon, 649
 - operators, 82–83, 109, 320–321, 714
 - extending (Operand2), 110
 - shift-left, 82
 - 128-bit, 467–468, 711
 - 192-bit, 469
 - extended-precision, 467
 - shift-right, 83–84
 - accumulating, 654
 - arithmetic, 84
 - extended-precision, 472
 - narrowing, 655
 - simulating rol instruction using ror, 470
- short-circuit Boolean evaluation, 319, 380
 - vs. complete Boolean evaluation, 382
- .short directive, 17
- side effects, 382
- signaling NaN, 335
- sign bit, 65
- sign condition code, 14
- signed comparison flag settings, 296
- signed conditional branches, 80
- signed division, 294
- signed integer-to-string conversion, 509–510
- signed multiplication, 450
- signed numbers, 65–70
 - complement method, 65
- sign extension, 110
 - and contraction, 71–72
 - values using asr, 473
- sign (N) flag, 295, 718
- significant digits, 323–324
- simulating div instruction, 321
- simulating rol instruction using ror, 470
- sin() function, 347
- single-dimension array access, 195
- .single directive, 97
- single-instruction, multiple data (SIMD) instructions, 14, 58, 621
- single-instruction, single-data (SISD) instructions, 621
- single-precision floating-point format, 94–95
 - conversion, 685
 - declarations, 17
 - exponent range, 95
 - precision, 95
- Sn registers, 623
- sorting, 198–203
 - quicksort, 278
- source code modules, xxx

- source files
 - editor, 4
 - merging during assembly, 862
 - sections, 6–7
 - .S source file suffix, 4
 - on assembly language source files, 743
 - .space directive, 196
- sparse jump tables, 399–402
- special-purpose kernel-mode registers, 11
- square root instructions, 686–687
- stack pointer (SP) register, 13, 146, 155
- stacks, 120, 155
 - accessing data on, 165–166
 - alignment, 155
 - cleanup, 163
 - pointer, 13
 - removing data from, 163–165
 - temporary storage on, 155
- stand-alone assembly code, 889
- stand-alone programs in assembly language, xxxiv
- starting bit position, 719
- state, machine, 405
- statements
 - break, 421
 - case, 389
 - conditional, 372
 - continue, 422
 - else, 746
 - if, 371
 - labels, 356
 - repeat...until, 417
 - on the same source line, 230
 - spread across multiple source lines, 230
 - switch, 389
 - while, 415
- state variable, 405
- static base (SB) register convention, 301
- store instructions, 26, 144, 332–333, 632–638
 - double, 156
- str.buf macro, 802–803, 807
- strength-reduction optimizations, 321
- string allocation
 - dynamic, 803
 - on the heap, 803
 - str.alloc function, 810
 - string.allocPtr field, 805
 - str.malloc function, 804
- string comparisons, 824, 829
- string expansion in macro parameters, 768
- string length, 187
 - computing at assembly time, 189
 - functions, 802
 - length, 796
 - zero-terminated string, 796
- string operations, 795
- strings, 189–194, 795
 - character, 187–194, 795
 - copying data, 818
 - data type for assembly language, 801–802, 805
 - functions, 190
 - length, 802
 - stdlib, 797–798
 - str.bufInit, 805
 - str.cpy, 818
 - str.free, 814
 - strlen(), 798
 - str.substr, 836
 - strtoh128, 584
 - Unicode, 857
 - length-prefixed, 188–189
 - pointers, 190
 - storage allocation, 803
 - zero-terminated, 187
 - problems with, 796
- string-to-numeric conversions
 - to floating point, 588–602
 - functions, 566
 - to integer, 566–587
- str.literal macro, 803, 807
- structs (structures), 212–220
 - arrays of, 218
 - .struct directive, 217
 - struct macro, 214, 779
- subnormal floating-point values, 342.
 - See also* denormalized values
- substituting text, 22

- substring function, 836
- subtraction
 - 256-bit, 446
 - extended-precision, 445–446
 - instructions, 28, 668–669
- surrogate code points, 847
- svc (supervisor call) instruction, 892–895
- swapping byte order, 134
- switch statements, 389–405
 - default clause in, 396
 - restrictions in simple implementations, 393
 - search implementations of, 403
- symbolic constants, 170
- symbols
 - checking if defined in CPP, 746
 - external, 6, 865
 - public, 233
- system bus, 11
- system cache and misaligned data, 140
- system register names, 93

T

- tables, 605
- tan() function, 347
- temporary values, 311
- temporary storage on stack, 155
- test bit instructions, 704–706, 715
- testing bits, 705–706
- text concatenation, 754
- .text directive, 121
- text editor, 4
- .text section, 6, 121–122
 - constants in, 130
- textual substitution, 22
- three-dimensional array access, 207
- Thumb instruction set, 103
- tokens, 754
- trampoline, 32, 369
- transfer instructions, 74, 144
 - for zero- and sign-extension, 474
- translating arithmetic expressions into assembly language, 293
- transpose instructions, 639
- tricky programming, 319
- true, representation of, 313

- truncation during floating-point operations, 324, 330
- truth tables, 58–60, 378
- two-dimensional row-major array access, 206
- two's complement numbering system, 56
 - notation, 65
- two's complement operation, 66

U

- u64toStr function, 500
 - u64toStrSize, 522
- u128toStr function, 511
- UFC (underflow cumulative) bit in FPSR, 332
- unaligned memory access, 16
- unconditional branch instruction, 77
- #undef statement, 759
- underflow, 326
- Unicode, 845
 - in assembly language, 853
 - Basic Multilingual Plane, 847–848
 - canonical equivalence, 849
 - character names, 848
 - character set, 55, 102, 846
 - code point, 847
 - combining characters, 852
 - encodings, 850
 - glyphs, 848
 - normal forms, 849–850
 - normalization, 849
 - string functions, 857
 - surrogate code points, 847
 - Unicode Text Formats, 850–851
- uninitialized data section, 124
- uninitialized pointers, 180
- unions, 220
- unordered comparisons, 97, 336
- unpacking bit strings, 719
- unraveling loops, 432, 763
- unsigned
 - conditional branches, 80
 - conversion
 - decimal to string, 495
 - integer to string, extended-precision, 510
 - division, 294

- multiplication, 211, 450
 - extended multiplication, 672
 - numbers, 65–70
- using registers for variables, 299
- uSize function, 517
- utoStrSize function, 522

V

- `__VA_ARGS__` symbol
 - expansion, 751
 - processing argument lists, 757
- value parameters, 255
- values, temporary, 311
- vararg parameter lists, 767
- variable argument lists, 751
- variable-length parameters, 263
 - lists, 33
- variables
 - 32-bit, 56–57
 - alignment, 19–21
 - choosing in memory, 140
 - automatic, 250
 - bytes, 55
 - declarations, 16–18
 - dword, 57
 - Gas, 16
 - global, 300
 - half-word, 56
 - lifetime, 250
 - local, 250, 300
 - loop control, 426
 - memory addresses, 19
 - names, 16
 - pointer, 178
 - qword, 57
 - read-only, 170
 - scope, 250, 865
 - state, 405
 - word, 57
- variadic parameters, 42
- V (overflow) condition code, 14
- vector, 128-bit shift-left, 467–468, 711
- vector comparisons, 687–693
 - compare immediate, 688–689
 - jump tables with, 692
 - multiway branch, 692
 - floating-point, 689
- vector division, 679–680

- vector instructions, 621
 - bit test, 691
- vector load immediate, 628
- vector multiplication, 671–678
- vector permutations, 645
- vector registers, 623
 - deinterleaving data in, 642
 - and immediate constants, 628
 - lanes in, 625, 641
 - moving data between, 625–626
- vector rotation, 710–711
- vector saturating accumulate
 - instructions, 666
- vener, 369
- vertical addition, 664
- vertical operations, 646
- Vn registers, 623
 - lane types, 624
- volatile registers, 31, 300, 346
 - preservation, 907
- Von Neumann architecture machine, 11
- vparmn macro, 42, 167, 775–777

W

- .warning directive, 760
- warning messages during assembly, 743
- #warning statement, 743
- warnings vs. errors, 744
- wastr (word-aligned string)
 - directive, 263
- wastr macro, 783
- while loops, 415–417
 - synthesis in assembly, 416
- wild pointers, 182
- word data type, 56, 57
- .word directive, 57
- word macro, 781
- WZR (zero) register, 28

X

- X16 and X17 registers used for
 - dynamic linking, 369
- X31 register, 146
- Xcode, 4
- XOR (exclusive-OR) operation, 59–60
 - 128-bit, 467
 - vectors, 648
- XZR (zero) register, 28, 146

Z

- zero-argument macros, 749
- zero condition code, 14
- zero extension, 71–72, 110, 112
- zero (Z) flag, 295, 716
 - setting after a multiprecision OR, 466
 - settings after `cmp`, 295

- zeroing out bits in a
 - register, 727
- zero-terminated strings, 17,
 - 187–188
 - length, 796
 - problems with, 796
- zip instructions, 641