# INDEX

**Numbers**

0-bits
- counting. *See* counting bits.
- isolating, 11
- trailing 0's
  - counting, 74, 84–87
  - identifying, 11
- turning on, 12

0-bytes, finding, 91–95

1-bits
- counting. *See* counting bits.
- identifying, 11
- isolating, 11
- right-propagating, 12
- rightmost, turning off, 11–12

**A**

absolute value
- computing, 17–18
- multibyte, 36–37
- negative of, 21

*add* instruction
- condition codes, 33–34
- propagating arithmetic bounds, 54–57

addition
- arithmetic tables, 285
- double-length, 34–35
- and logical operations, 15–16
- multibyte, 36–37
- of negabinary numbers, 225–226
- overflow detection, 26–28
- in various number encodings, 228–229

Advanced Encryption Standard, 126

alternating among values, 41–44

Alverson's method, 188–189

*and*, in three instructions, 16

arithmetic, computer *vs.* ordinary, 1

arithmetic bounds
- checking, 51–53
- of expressions, 54–55
- propagating through
  - add and subtract instructions, 54–57
  - logical operations, 58–63
- range analysis, 54
- searching for values in, 95–96

arithmetic tables, 285–288

arrays
- checking bounds. *See* arithmetic bounds.
- counting 1-bits, 72–73
- indexes, checking. *See* arithmetic bounds.
- indexing a sparse array, 73–74
- rearranging, 127
- of short integers, 36–37
- shuffling/unshuffling, 127

**B**

base –1 + *i* number system, 230–232

base –1 – *i* number system, 232–233

base –2 number system, 223–230, 239

big-endian format, converting to little-endian, 101

binary decomposition, integer exponentiation, 212–214

binary search
- counting leading 0's, 77–80
- integer logarithm, 215–221
- integer square root, 203–210

bit matrices, multiplying, 77

bit numbering, 1

bit operations
- *compress* operation, 93, 104, 108, 116–123
- computing parity. *See* parity.
- counting bits. *See* counting bits.
- cycling through bit combinations. *See* Gray code.
- extracting alternate bits, 104
- finding strings of 1-bits, 96–99

Warren.book Page 298 Monday, June 17, 2002 4:37 PM

flipping bits, 102
general permutations, 122–126
*generalized extract*, 116–122
half shuffle, 108
inner shuffle, 106–108
outer shuffle, 106–108
perfect shuffle, 106–108
reversing bits
6-, 7-, 8-, and 9-bit quantities, 102–104
definition, 101
generalized, 102
on rightmost bits. *See* rightmost bits.
searching words for bit strings, 83, 95–99
sheep and goats operation, 122–126
shuffling bits, 106–108, 116, 127
transposing a bit matrix, 108–116
unshuffling bits, 107–108, 116, 123, 127
bit vectors, 1
`bitgather` instruction, 124–126
*bitsize* function, 83
boundary crossings, powers of 2, 49–50
bounds, arithmetic. *See* arithmetic bounds.
*branch on carry and register result nonzero* instruction, 49
bytes
definition, 1
finding first 0-byte, 91–95
reversing, 92, 101–102

**C**

C language
arithmetic on pointers, 81, 190
GNU extensions, 71–72, 82
referring to same location with different types, 81
representation of character strings, 91
summary of elements, 2–4
ceiling function, identities, 139–140
Chang, Albert, 96
character strings, 91
checking arithmetic bounds, 51–53
comparison predicates
from the carry bit, 24–25
definition, 21
*number of leading zeros* (nlz) function, 21–22, 83
signed comparisons, from unsigned, 23–24
true/false results, 21
using negative absolute values, 21–22
comparisons
computer evaluation of, 25
floating-point comparisons using integer operations, 263–264
*three-valued compare* function, 19–20. *See also sign* function.
*compress* operation, 93, 104, 108, 116–123
computer algebra, 2–4
computer arithmetic, 1
condition codes, 33–34
constants
dividing by. *See* division by constants.
multiplying by, 133–136
counting bits. *See also number of leading zeros* (nlz) function; *number of trailing zeros* (ntz) function; *population count* function.
1-bits in 7- and 8-bit quantities, 71
1-bits in a word, 65–72
1-bits in an array, 72–73
*bitsize* function, 83
divide and conquer strategy, 65–66
leading 0's, with
binary search method, 77
floating-point methods, 81–82
*population count* instruction, 79–80
rotate and sum method, 69–70
search tree method, 84–85
with table lookup, 71
trailing 0's, 74, 84–87
by turning off 1-bits, 69
cryptography
Advanced Encryption Standard, 126
`bitgather` instruction, 124–126
DES (Data Encryption Standard), 125–126
Rijndael algorithm, 126
SAG method, 122–126
shuffling bits, 106–108, 125–126
Triple DES, 126
cube root, integer, 211–212
cycling among values, 41–44

Warren.book Page 299 Monday, June 17, 2002 4:37 PM

**D**

Data Encryption Standard (DES), 125–126
DEC PDP-10 computer, xi, 68
decryption. *See* cryptography.
denormalized numbers, 262
denorms, 262
DES (Data Encryption Standard), 125–126
*difference or zero* (doz) function, 37–38
distribution of leading digits, 264–267
divide and conquer strategy, 65–66
division
  arithmetic tables, 287
  doubleword by single word, 148–153
  floor, 137–138, 188
  modulus, 137–138, 188
  multiword, 140–145
  of negabinary numbers, 226–228
  nonrestoring algorithm, 148, 150–151
  notation, 137
  overflow detection, 32–33
  restoring algorithm, 148–150
  shift-and-subtract algorithms (hardware), 148–151
  short, 145–148, 151–152
  signed
    computer, 137
    long, 145–146, 153
    multiword, 144–145
    short, 146–148
  unsigned
    computer, 137
    long, 148–153
    short from signed, 145–148
division by constants
  by 3, 5, and 7, 158–160
  exact division
    definition, 190
    multiplicative inverse, Euclidean algorithm, 192–195
    multiplicative inverse, Newton's method, 195–197
    multiplicative inverse, samples, 197–198
  floor division, 188
  incorporating into a compiler, signed, 171–173
  incorporating into a compiler, unsigned, 183–184
  `magic` algorithm, 171–174
  magic numbers
    Alverson's method, 188
    calculating, signed, 162–163, 171–174
    calculating, unsigned, 182–187
    definition, 161
    sample numbers, 189–190
    table lookup, 188–189
    uniqueness, 173–175
  `magicu` algorithm, 183–185
  `magicu2` algorithm, 186–187
  modulus division, 188
  signed
    best programs for, 175–178
    by divisors ≤ –2, 168–171
    by divisors ≥ 2, 160–168
    incorporating into a compiler, 171–173
    by powers of 2, 155–156
    remainders from powers of 2, 156–157
    test for zero remainder, 200–201
    uniqueness, 173–175
  unsigned
    by 3 and 7, 178–180
    best programs for, 184–186
    by divisors ≥ 1, 180–183
    incorporating into a compiler, 183–184
    incremental division and remainder technique, 172, 183–184
    by powers of 2, 178
    remainders from powers of 2, 178
    test for zero remainder, 198–199
double buffering, 39
double-length addition/subtraction, 34–35
double-length shifts, 35–36
doublewords
  definition, 1
  dividing by single words, 148–153
doz (*difference or zero*) function, 37–38

Warren.book Page 300 Monday, June 17, 2002 4:37 PM

**E**

encryption. *See* cryptography.
end-around-carry technique, 228
estimating multiplication overflow, 31
Euclidean algorithm, 192–195
Euler, Leonhard, 271–272
even parity, 74
exact division
  definition, 190
  multiplicative inverse, Euclidean algorithm, 192–195
  multiplicative inverse, Newton's method, 195–197
  multiplicative inverse, samples, 197
exchanging
  conditionally, 41
  corresponding register fields, 39–40
  two fields in same register, 40
  two registers, 38–39
*exclusive or*
  propagating arithmetic bounds through, 62
  scan operation on an array of bits, 75
  in three instructions, 16
execution time model, 9
exponentiation
  by binary decomposition, 212–214
  in Fortran, 214–215
  powers of 2, testing for, 11–12
expressions
  arithmetic bounds, 54–55
  range analysis, 54–55
*extract* instruction, 122
extracting bits
  alternate bits, 104
  *generalized extract*, 116–122

**F**

factoring, 136
Fast Fourier Transform (FFT), 104–105
Fermat numbers, 271
FFT (Fast Fourier Transform), 104–105
*find leftmost 0-byte*, 91–94
*find rightmost 0-byte*, 91–92, 94–95
finding
  decimal digits, 95
  first 0-byte, 91–95
  first uppercase letter, 96
  length of character strings, 91
  next higher number, same number of 1-bits, 13–14
  the *n*th prime, 271–278, 282
  strings of 1-bits
    application, 96
    first string of a given length, 96–99
  values within arithmetic bounds, 95–96
flipping bits, 102
floating-point numbers
  comparing using integer operations, 263–264
  denormalized numbers, 262
  distribution of leading digits, 264–267
  gradual underflow, 262
  IEEE format, 261–263
  normalized numbers, 262
  table of values, 267–269
floating-point operations
  counting leading 0's with, 81–82
  simulating, 83
floor division, 137–138, 188
*floor* function, identities, 139–140
Floyd, R. W., 87
formula functions, 278–284
formulas for primes, 271–284
Fortran, integer exponentiation, 214–215

**G**

Gardner, Martin, 239
*generalized extract* operation, 116–122
Gosper, R. W.
  iterating through subsets, 14
  loop-detection, 87–89
gradual underflow, 262
graphics-rendering, Hilbert's curve, 258–259
Gray, Frank, 239
Gray code
  applications, 239–240
  converting integers to, 75, 236
  definition, 235
  incrementing Gray-coded integers, 237–239
  negabinary Gray code, 239
  reflected, 235–237
GRP instruction, 126

Warren.book Page 301 Monday, June 17, 2002 4:37 PM

**H**

hacker, definition, xiv
half shuffle, 108
halfwords, 1
Hamming distance, 73
high-order half of product, 132–133
Hilbert's curve. *See also* space-filling curves.
  applications, 258–259
  coordinates from distance
    description, 246
    Lam and Shapiro method, 248–250
    parallel prefix operation, 251–252
    state transition table, 246–248
  description, 241–242
  distance from coordinates, 252–254
  examples, 241, 243
  generating, 242–245
  incrementing coordinates, 254–257
  non-recursive generation, 257
  ray tracing, 258–259

**I**

IBM Stretch computer, 65
IBM System/360 computer, 49, 264–265
IEEE format, floating-point numbers, 261–263
image processing, Hilbert's curve, 258–259
incremental division and remainder technique, 172, 183–184
inequalities, logical and arithmetic expressions, 16–17
inner shuffle, 106–107
*insert* instruction, 122
instruction-level parallelism, 9
instruction set for this book, 4–9
*integer log base 2* function, 83, 215–216
integers. *See also specific operations on integers*.
  complex, 230–233
  converting to Gray code, 75, 236
  reversed, incrementing, 104–106
  reversing, 101–104
ISIGN (transfer of sign) function, 20
isolating rightmost bits, 11
iterating through subsets, 14

**J–K**

Jensen, Eric, 87

Knuth's Algorithm D, 140–144
Knuth's Algorithm M, 129–130
Kronecker, Leopold, 261

**L**

Lam and Shapiro method, 248–250, 253–254
Landry, F., 271
leading 0's, counting, 77–82. *See also number of leading zeros* (nlz) function.
leading digits, distribution, 264–267
little-endian format, converting to big-endian, 101
*load immediate* instruction, 8–9
*load* instruction, 9
*load word byte-reverse* (`lwbrx`) instruction, 92
logarithms
  binary search method, 216–218
  definition, 215–216
  log base 2, 83, 215–216
  log base 10, 215–221
  table lookup, 216, 218–221
logical operations
  with addition and subtraction, 15–16
  binary, table of, 17
  propagating arithmetic bounds through, 58–63
  tight bounds, 58–63
loop detection, 87–89
`lwbrx` (*load word byte-reverse*) instruction, 92

**M**

`magic` algorithm, 171–174
magic numbers
  Alverson's method, 188–189
  calculating, signed, 162–163, 171–174
  calculating, unsigned, 182–187
  definition, 161
  samples, 189–190
  table lookup, 188
  uniqueness, 173–175

Warren.book Page 302 Monday, June 17, 2002 4:37 PM

`magicu` algorithm, 183–185
`magicu2` algorithm, 186–187
*max* function, 37–38
Mills, W. H., 284
*min* function, 37–38
modu (*unsigned modulus*) function, 68
modulus division, 137–138, 188
Moore, Eliakim Hastings, 257–258
`mulhs` (*multiply high signed*) instruction
  division with, 158–160, 186
  implementing in software, 132
`mulhu` (*multiply high unsigned*) instruction
  division with, 178–180, 186
  implementing in software, 132
multibyte absolute value, 37
multibyte addition/subtraction, 36–37
multiplication
  arithmetic tables, 286
  by constants, 133–136
  factoring, 136
  high-order half of 64-bit product, 132
  high-order product signed from/to unsigned, 132–133
  multiword, 129–132
  of negabinary numbers, 226
  overflow detection, 29–32
multiplicative inverse
  Euclidean algorithm, 192–195
  Newton's method, 195–197
  samples, 197–198
*multiply high signed* (`mulhs`) instruction
  division with, 158–160, 186
  implementing in software, 132
*multiply high unsigned* (`mulhu`) instruction
  division with, 178–180, 186
  implementing in software, 132
*multiply* instruction, condition codes, 33–34
multiword division, 140–145
multiword multiplication, 129–132

**N**

negabinary number system
  description, 223–230
  Gray code, 239
negative absolute value, 21
negative overflow, 28
Newton's method
  description, 289–290
  integer cube root, 211
  integer square root, 203–206
  multiplicative inverse, 195–197
next higher number, same number of 1-bits, 13–14
nibbles, 1
nlz (*number of leading zeros*) function. *See number of leading zeros* (nlz) function.
nonrestoring algorithm, 148, 150–151
normalized numbers, 262
notation used in this book, 1–4
ntz (*number of trailing zeros*) function. *See number of trailing zeros* (ntz) function.
*number of leading zeros* (nlz) function
  applications, 83, 96
  *bitsize* function, 83
  comparison predicates, 21–22, 83
  counting trailing 0's, 84
  finding 0-bytes, 91–92
  finding strings of 1-bits, 96–99
  incrementing reversed integers, 106
  and integer log base 2 function, 83
  overflow detection, multiplication, 31
  random number generation, 83
  rounding to powers of 2, 47
*number of trailing zeros* (ntz) function
  applications, 87–89
  counting trailing 0's, 84–87
  finding strings of 1-bits, 83
  loop detection, 87–89
  random number generation, 83
  ruler function, 87
number systems
  base –1 + *i*, 230–232
  base –1 – *i*, 232–233
  base –2, 223–230, 239
  base efficiency, 233–234
  negabinary, 223–230, 239

Warren.book Page 303 Monday, June 17, 2002 4:37 PM

**O**

odd parity, 74
1-bits
  counting. *See* counting bits.
  identifying, 11
  isolating, 11
  right-propagating, 12
  rightmost, turning off, 11–12
*or*, in three instructions, 16
ordinary arithmetic, 1
ordinary rational division, 137
outer shuffle, 106–108
overflow detection
  definition, 26
  division, 32–33
  estimating multiplication overflow, 31
  multiplication, 29–32
  negative overflow, 28
  with *number of leading zeros* (`nlz`) instruction, 31
  signed add/subtract, 26–28
  subtraction, 27–29
  unsigned add/subtract, 29

**P**

parallel prefix operation
  *compress* operation, 119–120
  definition, 75
  generalized extract, 117
  Hilbert's curve, 251–252
  parity, 75
  swap and complement, 251–252
  *vs. insert/extract* instructions, 122
parity
  adding to 7-bit quantities, 76
  application, 77
  computing, 74–76
  definition, 74
  parallel prefix operation, 75
  scan operation, 75
Peano, Giuseppe, 241. *See also* Hilbert's curve.
Peano-Hilbert curve. *See* Hilbert's curve.
perfect shuffle, 106–108
permutations on bits, 122–126. *See also* bit operations.
planar curves, 241. *See also* Hilbert's curve.
poems, 202, 211
*population count* function
  applications, 73–74
  computing Hamming distance, 73
  counting 1-bits, 65
  counting leading 0's, 79–80
  counting trailing 0's, 84
position sensor, 239–240
powers of 2
  boundary crossings, detecting, 49–50
  identifying, 11–12
  rounding to, 45–48
  signed division, 155–156
  unsigned division, 178
PPERM instruction, 126
prime numbers
  Fermat numbers, 271
  finding the *n*th prime
    formula functions, 278–284
    Willans's formulas, 273–277
    Wormell's formulas, 277–278
  formulas for, 271–284
  from polynomials, 272
propagating
  arithmetic bounds
    *add* and *subtract* instructions, 54–57
    logical operations, 58–63
    through *exclusive or*, 62
  rightmost 1-bit, 12

**Q–R**

quicksort, 65

range analysis, 54
ranges. *See* arithmetic bounds.
ray tracing, Hilbert's curve, 258–259
rearranging arrays, 127
reflected binary Gray code, 235–236, 239
registers
  exchanging, 38–39
  exchanging conditionally, 41
  exchanging fields of, 39–40
  reversing contents of, 101–104
  RISC computers, 4–6

Warren.index.fm Page 304 Tuesday, June 18, 2002 9:26 AM

- remainders
  - arithmetic tables, 288
  - signed division
    - from non-powers of 2, 157–160
    - from powers of 2, 156–157
    - test for zero, 200–201
  - unsigned division
    - *and immediate* instruction, 178
    - incremental division and remainder technique, 172, 183–184
    - test for zero, 198–199
- *remu* function, 102–103
- restoring algorithm, 148–150
- reversing
  - 6-, 7-, 8-, and 9-bit quantities, 102–104
  - bits, 101–104
  - bytes, 92, 101–102
  - generalized, 102
  - integers, 101–104
  - *load word byte-reverse* (`lwbrx`) instruction, 92
  - register contents, 101–104
- right-propagating 1-bits, 12
- rightmost bits
  - 0-bits
    - isolating, 11
    - trailing 0's, counting, 74, 84–87
    - trailing 0's, identifying, 11
    - turning on, 12
  - 1-bits
    - identifying, 11
    - isolating, 11
    - right-propagating, 12
    - turning off, 11–12
- Rijndael algorithm, 126
- RISC
  - basic instruction set, 5–8
  - execution time model, 9
  - extended mnemonics, 7
  - full instruction set, 6–8
- rotate and sum method, 69–70
- rotate shifts, 34
- rounding to powers of 2, 45–48
- ruler function, 87

**S**

- SAG (sheep and goats) operation, 122–126
- scan operation, 75
- Schroeppel's formula, 229–230
- search tree method, 84–85
- searching. *See* finding.
- sheep and goats (SAG) operation, 122–126
- shift-and-subtract algorithm
  - hardware, 148–151
  - integer square root, 209–210
- *shift left double* operation, 35
- *shift right arithmetic* instruction, 8
- *shift right double signed* operation, 36
- *shift right double unsigned* operation, 35
- *shift right extended immediate* (`shrxi`) instruction, 179–180
- *shift right signed* instruction
  - alternative to, for sign extension, 18
  - division by power of 2, 155–156
  - from unsigned, 18–19
- shifts
  - double-length, 35–36
  - rotate, 34
- short division, 145–148, 151–152
- `shrxi` (*shift right extended immediate*) instruction, 179–180
- shuffling
  - arrays, 127
  - bits, 106–108, 116, 127
- sign extension, 18
- *sign* function, 19. *See also three-valued compare function*.
- signed bounds, 62–63
- signed comparisons, from unsigned, 23–24
- signed computer division, 137–138
- signed division
  - arithmetic tables, 287
  - best programs for, 175–178
  - by divisors ≤ –2, 168–171
  - by divisors ≥ 2, 160–168
  - incorporating into a compiler, 171–173
  - by powers of 2, 155–156
  - remainder from non-powers of 2, 157–160
  - remainder from powers of 2, 156–157
  - test for zero remainder, 200–201
  - uniqueness of magic number, 173–175

Warren.book Page 305 Monday, June 17, 2002 4:37 PM

signed long division, 145–146, 153
signed short division, 146–148
*signum* function. *See sign* function; *three-valued compare* function.
space-filling curves, 257–258. *See also* Hilbert's curve.
sparse array indexing, 73–74
square root, integer
  binary search, 207–209
  hardware algorithm, 209–211
  Newton's method, 203–207
  shift-and-subtract algorithm, 209–211
Stibitz, George, 232
Stretch computer, 65
*string length* (`strlen`) function, 91
strings. *See* bit operations; character strings.
`strlen` (*string length*) function, 91
*subtract* instruction
  condition codes, 33–34
  propagating arithmetic bounds, 54–57
subtraction
  arithmetic tables, 285
  *difference or zero* (doz) function, 37–38
  double-length, 34–35
  and logical operations, 15–16
  multibyte, 36–37
  of negabinary numbers, 225–226
  overflow detection, 26–28
  in various number encodings, 228–229
swap and complement method, 249–250
swapping pointers, 39
System/360 computer, 49, 264–265

**T**

table lookup, counting bits, 71
*three-valued compare* function, 19–20. *See also sign* function.
tight bounds
  *add* and *subtract* instructions, 54–57
  logical operations, 58–63
toggling among values, 41–44
Tower of Hanoi puzzle, 89, 239
trailing 0's. *See also number of trailing zeros* (ntz) function.
  counting, 84–87
  identifying, 11
transfer of sign (ISIGN) function, 20
transposing a bit matrix
  8 x 8, 108–111, 116
  32 x 32, 111–116
Triple DES, 126
true/false comparison results, 21
turning off 1-bits, 11–12, 69
turning on 0-bits, 12

**U**

uniqueness, of magic numbers, 173–175
unshuffling
  arrays, 127
  bits, 107–108, 116, 127
unsigned division
  by 3 and 7, 178–180
  arithmetic tables, 287
  best programs for, 184–186
  computer division, 137
  by divisors ≥ 1, 180–183
  incorporating into a compiler, 183–184
  incremental division and remainder technique, 172, 183–184
  long division, 148–153
  by powers of 2, 178
  remainders, from powers of 2, 178
  short division, from signed, 145–148
  test for zero remainder, 198–199
*unsigned modulus* (modu) function, 68
uppercase letters, finding, 96

**V–W**

Voorhies, Douglas, 259

Willans, C. P., 273–277
Wilson's theorem, 273
word parity. *See* parity.
words
  counting bits, 65–72
  definition, 1
  division
    doubleword by single word, 148–153
    Knuth's Algorithm D, 140–144
    multiword, 140–145
    signed, multiword, 144–145
    single word by single word, 145–148

Warren.book Page 306 Monday, June 17, 2002 4:37 PM

multiplication, multiword, 129–132
reversing, 101–104
searching for
first 0-byte, 91–95
first uppercase letter, 96
strings of 1-bits, 96–99
a value within a range, 95–96
word parallel operations, 12–13
Wormell, C. P., 277–278

**Z**

*zbytel* function, 91–95
*zbyter* function, 91–95
zero means $2^n$, 20–21
0-bits
counting. *See* counting bits.
isolating, 11
trailing 0's
counting, 74, 84–87
identifying, 11
turning on, 12
0-bytes, finding, 91–95