INDEX

Symbols

∧ (AND), 54
`*/` (asterisk, slash), end C comment, 26
`->` (dash, greater than), pointer select `struct` field, 378
`.` (dot operator), 372, 390
μ (mu), symbol for micro, 91
¬ (NOT), 55
Ω (omega), symbol for ohms, 89
∨ (OR), 55
`%` (percent), C integer division remainder, 39
`/` (slash), C integer division quotient, 39
`/*` (slash, asterisk), begin C comment, 26
`~` (tilde), C++ destructor naming, 388
`>>` (two greater thans), C++ extraction operator, 394
`<<` (two less thans), C++ insertion operator, 394
`//` (two slashes), start a comment, 194
⊻ (XOR), 81

A

A32, xxii, 167
A64, xxii, 167
AArch32, 167, 473
AArch64, xxii, 167
- exception levels, 469, 472–474
- memory access permission levels, 469
- procedure call standard, 212

AC (alternating current), 86
active components, 86, 96–102
adder, 108–114
- full, 109
- full from two half adders, 111
- half, 108
- *n*-bit, 112
- ripple-carry addition and subtraction, 112–114

addition
- binary, 41
- decimal, 38–39

address bus, 2, 3
addressing modes, 213–214
adjacency rule, 72
Advanced Microcontroller Bus Architecture (AMBA), 439
alternating current (AC), 86
ALU (arithmetic logic unit), 168–169
American Standard Code for Information Interchange (ASCII), 19
ampere, 86
application software, 468
arbitrary-precision arithmetic, link to libraries, 434
arithmetic logic unit (ALU), 168–169
Armv8-A processor, 167
array, 361
- accessing single element, 361, 365
- in assembly language, 367–371
- in C, 362–367
- character, 21, 27, 28, 361
- element data type, 361
- index, 361, 368–371
- index register in `ldr` and `str` instructions, 369–370
- integer, 362
- pass by pointer, 363

ASCII, 19, 335
assembler, 185
- algorithm, 237–239
- directive. *See* assembler directives
- local symbol table, 238
- opcode table, 238
- operations supported on literal values, 452
- two-pass, 238

assembler directives, 190–193
  .align, 192, 193, 221
  .arch, 191
  .ascii, 303
  .asciz, 303
  .byte, 303
  .cfi, 190
  .comm, 394
  .data, 273
  .else, 452
  .elsif, 452
  .endif, 452
  .equ, 223, 238
  .file, 191
  .global, 193
  .hword, 303
  .ident, 191
  .if, 452
  .include, 383
  .local, 394
  memory allocation, 303
  .quad, 266, 303
  .rodata, 224
  .section, 191
  .size, 191
  .skip, 274
  .string, 221, 303
  .text, 192, 224
  .type, 193
  .weak, 405
  .word, 303
  .zero, 274
assembly language, xxii
  compiler-generated, 188
  inline, 318–322
  instruction description notation, 195–196
  line organization, 189–190
  and machine language, 185, 195
  symbolic names, 194
assembly listing file, 228–230
automatic local variable. *See* variable

B

base, 13
battery (power supply), 86
baud, 463
BCD (binary coded decimal), 22–23
big-endian, 34
binary
  converting to unsigned decimal, 14
  C syntax for, 11–12
  digit (bit), 10
  number system, 10, 12–16
binary coded decimal (BCD), 22–23
binary point, 410, 411
bitmask, 324
bitmasking, 323, 414
  in assembly language, 330–332
  in C, 324–327
bits, representing groups of, 10–12
Boole, George, 53
Boolean algebra
  conjunction, 53
  disjunction, 53
  literal, 56
  negation, 53
  rules
    annulment value, 58, 60
    associative, 57
    commutative, 58
    complement, 60
    De Morgan's law, 61
    distributive, 58, 59
    elementary algebra, different from, 59
    elementary algebra, same as, 57
    idempotent, 61
    identity value, 58
    involution, 59
  term, 56
Boolean expressions, 56–61
  minimization of, 66
    minimal product of sums, 67
    minimal sum of products, 67
    using algebraic manipulations, 68
    using Karnaugh maps, 71–80
Boolean functions, 61–66, 81
  canonical forms, 62
    comparison of, 65–66
  canonical product, 64
  canonical sum, 62
  conjunctive normal form, 64
  disjunctive normal form, 63
  full conjunctive normal form, 64

full disjunctive normal form, 63
maxterm, 64
minterm, 63
product of maxterms, 64
product of sums, 64
product term, 63
sum of minterms, 63
sum of products, 63
sum term, 64
Boolean operators, 54
AND, 54
combining basic, 81
NAND, 103
NOR, 103
NOT, 55
OR, 55
precedence, 56
XOR, 81
borrowing (subtraction), 39
branch, 170, 244
conditional, 144, 244–246
condition codes, 245
software instruction, 174
unconditional, 244
branch predictor, 144
branch table, 265
bus (address, control, and data), 2, 170
byte, 11
order in memory, 33–35

C

C. *See also* function; preprocessor directives; variable
compiling, 3, 186–188
freestanding environment, 26
header file, 26
hosted environment, 26, 186, 475
I/O libraries, 23
program execution, 3
programming, 23
comments in, 25
first program, 25
C++. *See also* object
calling assembly language functions, 406–408
calling C functions, 406–408
compiler, 390
exceptions, 391
function overloading, 392
implicit argument to function, 392
I/O libraries, 23
standard library, 24
C++ class
access (private, public, and protected), 389
and C record, 388
`class` declaration, 389
constructor, 388
compiler-generated, 399
default, 388, 400
overloading, 403–405
data member, 388
in-class initializer, 400
destructor, 388
default, 400
explicitly defaulted functions, 400
member functions, 388
defining, 392
`struct` declaration, 389
C++ Core Guidelines (Stroustrup and Sutter), 388
C++ Crash Course (Lospinoso), 387, 400
C standard library, 24, 26
`printf` function, 25, 26
`puts` function, 256
`random` function, 256
relationship to operating system, 24
`scanf` function, 25, 28
stdio.h header file, 26
cache memory, 154–155, 161, 168–169
locality of reference, 155
speed, 155
call stack. *See* stack
capacitor, 90–93, 162
carry
addition, 38
flag. *See* condition flags
subtraction, 39
central processing unit. *See* CPU
characters
code point, 19
glyph, 19
storage of, 19–22
circuit symbols
adder, 112
AND gate, 55

circuit symbols *(continued)*
capacitor, 88
decoder, 116
inductor, 88
MOSFET, 98
multiplexer, 119
NAND gate, 103
NOR gate, 103
NOT gate, 55
OR gate, 55
PAL, 124
PLA, 121
resistor, 88
ROM, 122
switch, 88
XOR gate, 81
clock, 136
clock signal, 136
CMOS, 100–102
AND gate, 102
power consumption, 102
switching time, 102
compiler steps
assembly, 187
compilation, 186
linking, 187
preprocessing, 186
complement, 43, 55
diminished radix, 43
radix, 43
complementary metal-oxide semiconductor. *See* CMOS
component types, 86
active, 96–102
passive, 87–95
COM port, 462
computer subsystems, 1–2
conditional branch point, 144
conditionals, 255–267
condition flags, 37, 168–169, 174
carry flag (`C`), 37, 41, 113
negative flag (`N`), 114
overflow flag (`V`), 37, 48, 113
zero flag (`Z`), 114
conductor, 86
conjunction, 54
control bus, 2, 3
control flow, 243
branches, 244
conditionals, 255
fundamental constructs, 243
imperative vs. declarative programming, 243
iteration, 246
control unit, 9, 168–169
conversion
binary to unsigned decimal, 14
character to integer, 335
decimal to integer, 341
hexadecimal to integer, 334
integer to decimal, 350
unsigned decimal to binary, 15
to uppercase, 325, 331
coulomb, 86
count-controlled loop, 253
counter, 140
CPU, 2, 167
accessing features in assembly language, 314–322
execution state (AArch64 vs. AArch32), 167
internal bus, 169, 172
registers, 172
subsystems, 168–170

D

data bus, 2, 3
data segment, 192
data size
byte, 174
doubleword, 174
extension in `add` and `sub` instructions, 217–219
halfword, 174
quadword, 174
word, 174
data types
converting between `int` and `char`, 25
integral and register sizes, 175–176
DC (direct current), 86
debugger. *See* `gdb` debugger
decimal number system, 10, 12–16
addition, 38–39

converting unsigned decimal to binary, 15
subtraction, 39–40
decimal point, 411
decoder, 114–116, 163
decoder ring, 51
De Morgan's law, 61
dereference operator, 278
direct current (DC), 86
direct memory access (DMA), 443
disjunction, 55
division, 349
in assembly language, 354–358
in C, 349–354
by constant, 354
negative numbers, 351
by powers of two, 340
remainder, 351
using multiplication and shifts, 354
DMA (direct memory access), 443
controller, 443
`do-while` loop, 253–255
compared to `while` and `for`, 254
DRAM (dynamic random-access memory), 162, 438
duality, principle of, 61

E

editor, 2, 6
electric field, 90
electronics, 85
ELF (Executable and Linkable Format), 192
embedded systems, xxiii
Enable input, 114, 118, 133, 137, 140
endianness, 33
EPROM (erasable programmable read-only memory), 123
exception, 468
CPU response to, 471–474
events, 470
asynchronous, 470, 473
synchronous, 470
handler, 471, 473
levels, CPU, 468–470
increasing and decreasing, 472
and privilege, 468
page fault, 441
supervisor call, 474
system registers for handling, 471
vector table, 472
Executable and Linkable Format (ELF), 192
execution state, 167

F

farad, 91
file descriptors (`STDIN_FILENO`, `STDERR_FILENO`, and `STDOUT_FILENO`), 247
filename extensions, 187
finite state machine, 128
firmware, 469
fixed-point numbers, 411
fraction in decimal, 415–419
fraction in powers of two, 411–415
flip-flop, 136–149
asynchronous inputs, 138
D, 137–140, 143, 148, 156–159
negative-edge triggering, 138
positive-edge triggering, 138
primary and secondary portions, 137, 158
floating-point arithmetic errors, 425–434
absorption, 427
associativity, 429
cancellation, 428
general comments, 434
in intermediate results, 430–433
rounding, 426
floating-point hardware, 422–425
convert precision, 425
registers, 173, 422
floating-point numbers, 419
representation. *See* IEEE 754 floating-point standard
`for` loop, 252–253
compared to `while` and `do-while`, 254
formal parameters, 270
fractional values in binary, 410
frame pointer (`fp`), 213, 279
frame record, 279
freestanding environment, 26
freestanding programs, 474
linking, 475

function, 23
  arguments to. *See* function arguments
  data interaction, 271
  declaration, 270
  definition, 26, 270
  format, 26
  input to, 271
  output from, 271
  overloading, 403
  parameter list, 26, 270, 395, 403
  prototype, 26, 270, 274
  return value, 276, 326
  scope, 270
function arguments, 26, 271
  more than eight, 281–288
  pass by pointer, 276
  pass by reference, 276
  pass by value, 276
  passing in assembly language, 278–281
  passing in C, 277–278
  passing in registers, 220–222, 278–281

G

gate, 54
  AND, 55
  NAND, 102
    as universal, 104–106
  NOR, 102
    as universal, 106
  NOT, 55, 101
  OR, 55
  XOR, 81
`gdb` commands
  `b` (break), 29
  `c` (continue), 29
  `h` (help), 29
  `i r` (info registers), 29
  `l` (list), 29
  `n` (next), 177
  `print` (print), 29
  `printf` (print formatted), 30
  `r` (run), 30
  `s` (step), 177
  `sho arc` (show architecture), 177
  `si` (step instruction), 177
  `x` (examine memory), 30
`gdb` debugger, 29
  breakpoint, 31
  as learning tool, 29, 198–204
  TUI mode, 198–204
  usage example, 30–33
  viewing registers, 176–182
general-purpose I/O. *See* GPIO
gibibyte, 441
global offset table (GOT), 240, 275
global variable. *See* variable
Goldberg, David, 434
GOT (global offset table), 240, 275
GPIO, 443. *See also* RP1
  chips, 443, 448
  command line utilities, 446
  compiling with `libgpiod` library, 447
  datasheet link, 448
  GPFSEL registers, 448, 454
  GPSET registers, 455
  header pins, 444
  installing `libgpiod` library, 446
  lines and header pins, 444
  programming in C and Python, 445
  programming libraries, 445–446
Gray code, 74
Gray, Frank, 74

H

Harvard architecture, 154
  in cache, 154, 155
header file, 274
  C functions, 274
  C++ class, 388
  C standard library functions, 26
heap memory, 192
henry, 93
hexadecimal, 10
  to binary, 10
  characters, UTF-8 code for, 19
  C syntax for, 11
  to `int`, 333
  to signed decimal, 45
  to unsigned decimal, 41
  using, 11
hidden parameter, 395
hosted environment, 26
hypervisor, 469, 473, 476

I

IEEE 754 floating-point standard, 420–422
- biased exponent, 421
- denormalized, 421
- exponent, 420
- general pattern, 420
- hidden bit, 421
- infinity, 422
- link to, 420
- normalized form, 421
- not a number (NaN), 422
- patterns of different precisions, 420–421
- significand, 420
- special cases, 421
- zero, 421

`if` conditional, 256–257
`if-else` ladder, 261, 267
`if-then-else` conditional, 258–262
immediate value, 196
inductor, 93–95
`info` program, 5
information hiding, 276
input/output. *See* I/O
instruction, 2
- data processing, 195, 232–236
- execution cycle, 170–171
- fetching, 3, 170
- load and store, 195, 231–232
- logic (AND, OR, and XOR), 327
- operation code (opcode), 190
- program flow control, 196, 236–237
- queue, 170
- register, 170

instructions
- `add`, 217, 218
- `adr`, 221
- `adrp`, 221
- `and`, 328
- `asr`, 337
- `b`, 244
- `b.`*`cond`*, 245
- `bic`, 454
- `bl`, 222
- `blr`, 222
- `br`, 244
- `cbnz`, 245
- `cbz`, 245
- `cinc`, 317
- `cmp`, 250
- `cneg`, 418
- `csel`, 267
- `eor`, 328
- `eret`, 472
- `fadd`, 424
- `fcmp`, 425
- `fcvt`, 425
- `fdiv`, 424
- `fmul`, 424
- `fsub`, 424
- `hvc`, 476
- `ldp`, 214, 215
- `ldr`, 216, 369
- `ldrb`, 217
- `ldrsb`, 217
- `ldrsw`, 217
- `lsl`, 337
- `lsr`, 337
- `mov`, 197
- `movk`, 197
- `movn`, 197
- `movz`, 197
- `mrs`, 471
- `msub`, 357
- `mul`, 347
- `orr`, 328
- `ret`, 197
- `ror`, 337
- `sdiv`, 357
- `smc`, 476
- `smulh`, 348
- `smull`, 347
- `stp`, 212
- `str`, 217, 369
- `strb`, 217
- `sub`, 218
- `tbnz`, 246
- `tbz`, 246
- `udiv`, 357
- `umulh`, 348
- `umull`, 347

instruction set architecture (ISA), xxi
integer codes, circular nature of, 51
integers, signed, 44
- adding and subtracting, 47–50

integers, unsigned, 22
  adding and subtracting, 38–43
integral data types, 173, 175–176
integral values, 409
intellectual property (IP), 167
interpreter, 3
interrupt, 468, 470
  controller, 443
  handler, 443
interrupt-driven I/O, 443
invert (NOT), 55
I/O, 2, 437. *See also* RP1
  device controller, 439
    data, status, and control
      registers, 440
  devices, 437
    accessing, 439–442
    programming, 442–443
    timing, 438–439
  functions
    `get_int`, 358
    `get_uint`, 358
    `put_int`, 358
    `put_uint`, 358
    `read_str`, 293
    `write_char`, 293
    `write_str`, 293
  interrupt-driven, 443
  memory-mapped, 440
  polled, 442, 460–465
  port-mapped, 440
  programmed, 443
`iostream` library, 24
IP (intellectual property), 167
ISA (instruction set architecture), xxi
iteration, 246–255
  recursion vs., 246, 306

J

joule, 95
jump table, 265

K

Kahan summation algorithm, 434
Karnaugh, Maurice, 71
Karnaugh map, 71–80, 142
  adding more variables, 79
  don't care cells, 79
  Gray code, 74
  larger groupings, 78
  ordering cells in, 74
  simplifying products of sums, 77
  simplifying sums of products, 71
  with three variables, 75
kibibyte, 441

L

latch, 128–136
  D, 134
  feedback in, 128
  SR, gated, 133
  SR using NAND gates, 131
  SR using NOR gates, 128
  SR with *Enable*, 133
leaf function, 280
least significant digit, 13
LED
  blinking in assembly language,
      RPi 3 and 4, 448–456
  blinking in assembly language,
      RPi 5, 456–459
  blinking in C, 445–448
  circuit for blinking, 445
Levine, John R., 192
linker
  algorithm, 239
  global symbol table, 240
Linkers & Loaders (Levine), 192
link register (`lr`), 174, 279
Linux Command Line, The (Shott), 5
Linux commands
  `apt`, 5, 7, 446
  `as`, 198, 229
  `code`, 7
  `g++`, 391
  `gcc`, 3, 28, 30, 177, 186, 188, 190,
      198, 447
  `gdb`, 30, 177, 199
  `info`, 5
  `ld`, 475
  `make`, 188
  `man`, 5
  `objdump`, 235
  `python`, 3
  `sudo`, 5, 7, 446, 451
little-endian, 33, 34
load–store architecture, 216
local variable. *See* variable

logic circuit, 107
    combinational, 107
    sequential, 107, 127
        designing, 140–149
logic instructions, 327
    immediate data in, 329
    shift register values in, 329
loop control variable, 246
looping, 246–255
Lospinoso, Josh, 387, 400

M

machine code, viewing, 228
machine instruction, xxi, 2
magnetic field, 93
`main` function, 26, 207
    in assembly language, 222–224
    minimum in C, 188
main memory, 153–154, 162
    organization, 153
man page, 5
mask, 324
mass storage, 152–153
    block size, 153
maxterm, 64
Mealy state machine, 128
mebibyte, 441
memory, 2, 151. *See also* cache memory; main memory
    access permission levels, 469
    addresses, 17
    addressing, 163
    addressing modes, 213–214
    cost, 152
    data storage in, 17–22
    hardware, 156–164
    hierarchy, 151
    layers, 152
    memory management, 440
    offline, 152
    page, 441
    page table, 441, 469
    physical, 441
    program characteristics in, 302–303
    random access, 17
        dynamic, 162–164
        static, 161
    read-only, 17
    read/write, 160–161
    segments, 192
    speed, 152
    timing, 438
    virtual, 440
    volatile, 153
memory management unit. *See* MMU
metal-oxide-semiconductor field-effect transistor. *See* MOSFET
minterm, 63
MMU, 441
    page fault exception, 441
    TLB, 442
Moore state machine, 128, 130
MOSFET, 97–100
    channel, 97
    drain, 97
    gate, 97
    N-channel, 98
    P-channel, 99
    power consumption, 100
    source, 97
    switching time, 100
most significant digit, 13
multiplexer (MUX), 117–119, 160
multiplication, 340
    in assembly language, 344–348
    in C, 340–344
    by powers of two, 340
    using shift and add, 344
MUX (multiplexer), 117–119, 160

N

name decoration, 299, 392
name mangling, 299
negation, 46
newline character (`\n`), 21
nonvolatile memory (NVM), 17
not a number (NaN), 422
numerical accuracy, 434
`nzcv` register, 174

O

object, 387
    attribute, 387
    class, 388
    creating, 390–392
    instance, 387
    instantiation, 388, 390
    message, 387

object *(continued)*
method, 387
in stack frame, 395
octal, 11
C syntax for, 11
digits, 11
ohms, 89
Ohm's law, 88
overflow flag. *See* condition flags
overloading, 403

P

PAL (programmable array logic), 124
Pascal, 21
passing arguments. *See* function arguments
passive components, 86–95
`pc` (program counter), 168–172, 174
`pc`-relative addressing, 213
PIC (position-independent code), 213
PIE (position-independent executable), 213
pipeline, 144
PLA (programmable logic array), 120, 143, 148
PLD (programmable logic device), 119–125
PLT (procedure linkage table), 240
polled I/O, 442, 460–465
positional notation, 12
position-independent code (PIC), 213
position-independent executable (PIE), 213
POSIX standard, 247
potential difference, 86
power consumption, 95–96
power supply, 86
preprocessor directives, 186
`#define`, 186, 249
`#endif`, 274
`#ifndef`, 274
`#include`, 26, 186
`printf`, 25, 26
conversion specifiers, 27
privilege level, 468
procedure linkage table (PLT), 240
program, 2
creating and executing, 2–4
documentation, 25
program counter (`pc`), 168–172, 174
programmable array logic (PAL), 124
programmable logic array (PLA), 120, 143, 148
programmable logic device (PLD), 119–125
programmable read-only memory (PROM), 123
programmed I/O, 443
programming environment, 4
programming tool documentation
`info`, 5
`man`, 5
PROM (programmable read-only memory), 123
propagation delay, 135
pseudo-op, 190
PSTATE, 472–474
pull-down device, 99
pull-down resistor, 100
pull-up device, 99
pull-up resistor, 100
Python, xxii, 2–3
program execution, 3

R

radix, 13
radix point, 411
RAM (random-access memory), 17, 153
Raspberry Pi Imager, 4
Raspberry Pi models
and book programs, xxii
cache sizes, 154
GPIO chip connections, 448, 456
main memory sizes, 153
Raspberry Pi OS, xxii, 4, 467
read-only memory (ROM), 17, 122
real numbers, 419
record, 371
in assembly language, 380–384
in C, 372–380
location of data in (field, element, and member), 371
passing to a function, 374
in stack frame, 373–374, 380, 383
recursion, 305–313
base case, 305
iteration vs., 246, 306

saving arguments in registers, 308
stack usage, 309, 313
register file, 160
register–memory architecture, 216
registers, 156, 172–175
A64, 172–175
bit numbering, 173
content, saving, 220
`fp` (frame pointer), 213, 279
general-purpose, 173–174, 220
for handling exceptions, 471
hardware implementation, 156–158
`lr` (link register), 174, 279
`nzcv` (condition flags), 174, 267
passing arguments in, 220–222
`pc` (program counter), 170, 174
shift, 159
sizes, 174
`sp` (stack pointer), 174
32-bit, 174
`wzr` (zero register), 174
`xzr` (zero register), 174
relative addressing, 213
resistor, 88–90
connected in parallel, 90
connected in series, 89
ripple-carry addition and subtraction, 112–114
ROM (read-only memory), 17, 122
Roman numeral system, 13
rounding, 410, 425–427
ties-to-even rule, 410
RP1, 439
documentation, 456
GPIO, 456–459
registered I/O (RIO), 458
RIO register offsets, 458
STATUS and CTRL registers, 458

S

SAM (sequential access memory), 17
`scanf`, 25
conversion specifiers, 27
scientific notation, 420
scope, variable, 270
secure monitor call, 469, 476
selection, 255–267
semiconductor, 97
doping, 97
holes, 97
N-type, 97
P-type, 97
sentinel character, 21
sentinel value, 252
sequential access memory (SAM), 17
sequential logic circuit, designing, 140–149
branch predictor, 144
counter, 140
serial-in parallel-out (SIPO), 159
Shannon, Claude, 53
shifting bits, 333
in assembly language, 337–340
in C, 333–337
in logic instructions, 329
shift register, 159
Shott, William, 5
signal voltage levels, active high and active low, 108
sign bit, 45
signed numbers, 14
sign-magnitude code, 43
SIPO (serial-in parallel-out), 159
SoC (System on a Chip), 167, 439
software
application vs. system, 468
privilege level, 468
source code, 2
southbridge, 439
Spolsky, Joel, 21
SRAM (static random-access memory), 161, 438
stack, 207–210. *See also* stack frame
data structure, 208–210
empty, 209
full, 209
growth (ascending/descending), 209
local variables on, 215–216
overflow/underflow, 209
pointer, 174, 208
segment, 192
stack frame, 210–216
designing, 281, 308, 337–338, 344–345, 380, 383
discipline, 288–292
frame pointer, 213
frame record, 279, 281

stack frame *(continued)*
    local variables in, 284
    objects in, 395
    overall pattern, 288
    passing arguments in, 284
    records in, 373–374, 380, 383
    in recursion, 310–313
    saving registers in, 281
standard error, 247
standard input, 247
standard output, 186, 247
state diagram, 128–130
state transition table, 130
static local variable. *See* variable
static random-access memory (SRAM), 161, 438
`stdio` library, 24
string, text, 21
    C-style, 21
Stroustrup, Bjarne, 388
`struct`
    accessing field by pointer in C, 377–378
    accessing field in C, 372
    declaring new data type of, 375
    defining in C, 372
    defining with tag in C, 376
    pass by pointer, 377
    pass by value, 377
    tag, 374–376
structure. *See* record
subfunctions, 207
subsystems, computer, 1–2
subtraction
    binary, 42
    borrow in, 39
    decimal, 39–40
subtractor, 113
supervisor call, 474–476
    register contents for, 476
Sutter, Herb, 388
`svc` (supervisor call) instruction, 475
`switch` conditional, 262–267
switches, 9, 88
    CMOS, 100–102
    MOSFET, 97–100
    representing with bits, 10
    transistors, 96–102
switching algebra, 53
system call functions
    `close`, 453
    file descriptors, 247
    `mmap`, 452
    `munmap`, 453
    `open`, 451
    `read`, 24, 247
    `write`, 25, 247
System on a Chip (SoC), 167, 439
system software, 468
system state, 127
systems programming, xxiii

T

T32, 167
text editor, 6–7
text segment, 192
text string, 21
time constant, 92, 94
timing considerations, 437
    I/O device, 438
    memory, 438
TLB (translation lookaside buffer), 442
Tour of C++, A (Stroustrup), 388
transistors, 96–102
    switching time, 102
tristate buffer, 118, 160
truth table, 54
    AND, 55
    NOT, 55
    OR, 55
    XOR, 81
two's complement, 43–47
    computing, 46

U

UART (universal asynchronous receiver/transmitter), 460
    16550 datasheet, link to, 463
    16550 registers, 461
    data bits, 461
    polling in C, 462–465
    start and stop bits, 460
Unicode UTF-8, 19
unsigned numbers, 14
uppercase vs. lowercase, ASCII, 323
UTF-8, 19

V

variable
- automatic local, 192, 224, 279, 292–299
- declaration in C, 270
- definition in C, 27, 270
- global, 192, 271–276, 293–299
- local, 215–216, 270
- name scope in C, 270
- static local, 192, 293–302

very large scale integration (VLSI), 139
virtual memory, 440
- page, 441

volt, 86
von Neumann, John, 153
von Neumann architecture, 153
von Neumann bottleneck, 154

W

watt, 95
"What Every Computer Scientist Should Know About Floating-Point Arithmetic" (Goldberg), 434
`while` loop, 247–252
- compared to `do-while` and `for`, 254

Y

"Your Turn" exercises, how to use, xxv–xxvi

Z

zero register, 174