

Index

- Abstraction, 6, 263
 - implementation paradigm, xi–xii
 - modules and, 2–3
- Adder gates, 29–39
- Addresses, 45, 104
 - direct addressing, 60–61
 - indirect addressing, 61
 - machine language and, 60–61, 63
 - mapping and, 84–91, 137–141
 - memory and, 81–82 (*see also* Memory)
 - program size limits and, 106
 - registers and, 45, 83–86
 - subroutines and, 153–159
 - symbol table and, 105
 - VM-Hack mapping and, 139–143, 161–168
- Addressing instruction (*A*-instruction), 64–65, 108–110, 115
- Algorithms
 - efficiency and, 249, 272–273
 - graphics and, 257–263
 - mathematics and, 248–252
 - memory management and, 252–256
 - operating systems and, 272–273 (*see also* Operating systems)
 - runtime and, 249
 - syntax and, 250
- ALU. *See* Arithmetic Logic Unit
- Analysis-synthesis paradigm, 223
- And function, 8–9, 20
 - implementation of, 26
 - multi-bit versions of, 21–23
- Application Program Interface (API)
 - notation, 19
- Architecture, x, 79, 99–101
 - bottom-up, 3–4
 - chip set, 2
 - CPU and, 82–83
 - Hack, 5–6, 85–98
 - hardware, 2
 - I/O and, 84–85
 - Jack, 175–176 (*see also* Jack)
 - machine language and, 106–107
 - memory and, 81–82
 - modifications and, 277–279
 - modules and, 2–3
 - optimization and, 80
 - registers and, 83–84
 - sequential chip hierarchy and, 47–50
 - standards and, 84
 - stored program concept and, 80
 - top-down, 3
 - VM and, 121–151 (*see also* Virtual Machine)
 - von Neumann, 62, 79–81, 85
- Aristotle, 6
- Arithmetic addition, 37
- Arithmetic Logic Unit (ALU), 2, 6, 39
 - Boolean arithmetic and, 29, 32, 35–38
 - combinational chips and, 46–47
 - CPU and, x, 82–83, 94
 - description of, 35–38
 - Hack and, 86

- Arithmetic Logic Unit (ALU) (cont.)
 - operating systems and, 248–249
 - visualized chip operations and, 292
- Arrays, 81
 - data translation and, 224–231
 - Jack and, 175, 184–185, 191, 265, 269
 - operating systems and, 256, 265, 269
 - stack processing and, 124–127
 - variable-length, 256
 - Virtual Machine (VM) and, 137
- ASCII code, 71, 89, 252
- Assembler, x, 5, 71–72, 75–76, 118–120, 277
 - hash table, 115
 - implementation of, 112–116
 - labels and, 105
 - machine language specification and, 107
 - macros and, 117
 - mnemonics and, 108, 114
 - program size limits and, 106
 - symbols and, 60, 104–106, 110–111, 114–116, 143, 164
 - syntax and, 104, 107–110
 - test scripts and, 103–104
 - as translator program, 104–107, 163–164
 - variables and, 105
- Best-fit, 254
- Big-Oh notation ($O(n)$), 249
- Binary code, 5, 108. *See also* Boolean logic
 - code generation and, 223–246
 - graphics and, 257–263
 - Jack and, 174
- Binary search, 251
- Bitmaps, 259–263, 269
- Bit shifting, 60
- Bit-wise negation, 60
- Boolean arithmetic, x
 - addition, 30
 - algebra and, 8–10
 - ALU and, 29, 32, 35–38
 - binary numbers and, 30–32
 - CPU and, 29
 - least significant bits (LSB), 30
 - memory and, 42–47
 - most significant bits (MSB), 30
 - radix complement method, 31
 - signed binary numbers, 31–32
 - stack processing and, 126–127
- Boolean logic
 - abstraction of, 11
 - algebra and, 8–10
 - canonical representation, 9
 - conditional execution, 62
 - gates and, 8, 11–13
 - hardware construction and, 13–14
 - HDL and, 14–17
 - machine language and, 57–77
 - repetition, 61–62
 - subroutine calling, 62
 - truth tables, 8
 - two-input functions, 9–10
- Bootstrap code, 165
- Buses, 21–22, 286–287
- C#, 4–5, 112, 121, 147, 169
 - Jack and, 174, 196
- C++, 112, 147, 253
- Canonical representation, 9
- Case conventions, 108
- Central Processing Unit (CPU), 6, 29, 59
 - ALU and, 82–83, 94
 - architecture and, 82–83
 - control unit and, 82–83
 - description of, 82–83
 - emulators and, 306–309
 - Hack and, 62–63, 85–96
 - instruction memory and, 82
 - program counter and, 84
 - registers and, 82
 - testing and, 306–309
 - von Neumann architecture and, 81
- Character output, 259–263, 269
- Chips, 2. *See also* Gates
 - adder, 29–39
 - API specification and, 19
 - Boolean logic and, 7–28
 - built-in, 287–288, 293, 296, 304–305
 - buses and, 286–287
 - clocks and, 289–291
 - combinational, 41, 46–47

- connections and, 285–286
- cost and, 14–15
- description of, 11
- efficiency and, 288
- feedback loops and, 291–292
- Hack platform and, 85–91
- hardware simulator and, 283–284
- HDL and, 14–17, 281–296
- incrementer, 33–39
- maintaining state and, 41–42
- pins and, 284–286
- RAM, 86
- ROM, 85
- sequential, 41–55, 289–292
- simulators and, 299–306
- testing and, 297–313
- visualized operations for, 288, 292–296
- Clocks, 41, 48, 289–290
 - feedback loops and, 291–292
 - memory and, 42, 52–54
- Code generation
 - commands translation and, 231–232
 - data translation and, 224–231
 - operating systems and, 272 (*see also* Operating systems)
 - registers and, 223–224
 - syntax analysis and, 237–241
 - virtual machines and, 224
- Combinational logic. *See* Boolean arithmetic
- Commands translation, 231–232
- Common Language Runtime (CLR), 123, 146–147
- Communications, 279
- Compare file, 18
- Compilers, ix–x, 2, 5–6, 17, 103, 112
 - abstraction and, 175–179
 - analysis-synthesis paradigm and, 223
 - code generation and, 223–246
 - description of, 199–201
 - grammars and, 203, 206–207
 - Hack and, 133–134 (*see also* Hack)
 - high-level language and, 146–147
 - Jack and, 133–134, 174, 193–195 (*see also* Jack)
 - lexical analysis and, 202, 208
 - mapping and, 137–141
 - memory allocation and, 234
 - nested subroutine calling and, 153
 - parsing and, 203–207
 - p-code and, 123, 146
 - semantics and, 199
 - syntax analysis and, 199–221, 237–241
 - VM and, 122–127, 161–168, 233–235 (*see also* Virtual Machine)
 - XML and, 199–201, 211–218, 221
- Complex Instruction Set Computing (CISC), 98
- Composite gates, 11–13
- Compute instruction (*C*-instruction), 66–69, 86, 108–110, 115
- Computers. *See also* Architecture
 - ALU and, 29 (*see also* Arithmetic Logic Unit)
 - Boolean abstraction and, 11
 - bootstrap code and, 165
 - CPU and, 29 (*see also* Central Processing Unit)
 - dedicated, 97
 - emulators and, 121–122
 - general-purpose, 97–98
 - HDL and, 14–17 (*see also* Hardware Description Language)
 - machine language and, 57–77
 - memory and, 81–82
 - program flow and, 153–159
 - stored program concept and, 79–80
- Conditional execution, 62
- Conditional jump, 62
- Constants, 181–182
- Control logic, 94–95
- Control unit, 82–83
- Converters. *See* Not function
- Counters, x, 45, 47–48, 50, 52, 84, 95
- CPU. *See* Central Processing Unit
- Cycles, 42
- Data flip-flop (DFF)
 - clocked chips and, 290–291
 - implementation of, 50–51
 - sequential logic and, 42–48

- Data races, 46
- Debugging, 75
- Decoding, 94–96
- Defragmentation, 254–256
- Demultiplexors, 21, 24–26
- Design. *See also* Architecture
 - alternative elements for, 277
 - Boolean logic and, 7–28
 - bottom-up, 3–4
 - cost and, 14–15
 - digital, 27
 - gate logic and, 11–13
 - HDL and, 14–17 (*see also* Hardware Description Language)
 - modifications and, 277–279
 - standards and, 84
 - testing and, 297–313
 - top-down, 3
- Device driver, 256–257
- Direct addressing, 60–61
- Division, 250–251
- DOS, 272
- Emulators, 121–122
 - Hack and, 76–77
 - testing and, 297, 306–313
- Equivalence function, 10
- Execute cycle, 86, 98
- Expression evaluation, 187–188, 231–232
- Feedback loops, 46, 52–53, 291–292
- Fetching, 86, 95–96, 98
- File formats, 107–110
- First-fit, 254
- Flip-flops, x, 41, 52–54, 287–288
 - clocked chips and, 290–291
 - data, 42–51, 290–291
 - implementation of, 50–51
 - memory and, 42
- Flow control, 231–232
- Formal languages, 201–202
- Fragmentation, 254, 256
- FreeList, 254, 256
- Full-adder chip, 32–33, 38
- Functions. *See also* Boolean logic
 - And, 8–9, 20–23, 26
 - assembly language symbols and, 164
 - bootstrap code and, 165
 - calling commands and, 159–164
 - compilers and, 233–235 (*see also* Compilers)
 - Jack and, 174–175, 190–193 (*see also* Jack)
 - Nand, 2, 7, 10, 19, 27
 - Nor, 2, 10
 - Not, 8–9, 26
 - Or, 8–9, 20–26
 - subroutines, 62, 112, 153–161, 181–190, 195, 209, 234–235
 - symbolic names and, 160
 - testing and, 297–313
 - VM-Hack mapping and, 139–143, 161–168
 - Xor, 10, 20–23, 26
- Gates, ix–x, xvi, 4, 6
 - adder, 29–39
 - And, 8–9, 20–23, 26
 - API specification and, 19
 - Boolean arithmetic and, 29–40
 - Boolean logic and, 7–28
 - built-in chips and, 287–288
 - buses and, 21–22, 286–287
 - composite, 11–13
 - construction of, 13–14
 - demultiplexors and, 21
 - flip-flops and, 41–54, 287–291
 - HDL and, 14–17 (*see also* Hardware Description Language)
 - interfaces and, 12–13
 - memory and, 42–47
 - multi-bit versions of, 21–25
 - Nand, 2, 7, 10, 19, 27
 - Nor, 2, 10
 - Not, 8–9, 26
 - Or, 8–9, 20–26
 - primitive, 11–13, 25–26
 - sequential, 41–55
 - specification, 17–25
 - switching devices and, 2
 - Xor, 10, 20–23, 26

- Goto operation, 95, 155
- Grammars
 - Jack and, 203, 207–215
 - parsing and, 203–207
 - syntax analyzer and, 207–213
- Graphical User Interface (GUI), 247, 283, 288–290
 - testing and, 297–313
 - visualized chip operations and, 292–296
- Graphics, 98
 - character output, 259–261
 - circle drawing, 259
 - keyboard handling and, 261–263
 - line drawing, 257–258
 - multiplication and, 258–259
 - pixel drawing, 257
- GUI. *See* Graphical User Interface
- Hack, 5, 35, 79
 - address instruction format and, 64–65, 85–86
 - assembler, 75–76, 103–120
 - built-in chips and, 293, 296
 - case conventions and, 108
 - case sensitivity, 75
 - C-instruction, 66–69
 - CPU and, 62–63, 76–77, 85–96
 - destination specification and, 66–68
 - file formats and, 71–72, 107–110
 - graphics card and, 98
 - input/output (I/O) handling and, 70–71, 98
 - instructions and, 108–110
 - Internet and, 279
 - jump specification, 68–69
 - memory and, 63, 87–91, 96
 - modifications and, 278–279
 - platform description, 62–64, 85–98
 - symbols, 69–70
 - syntax, 71–73, 107–110
 - VM mapping and, 139–143, 161–168
- Half-adder chip, 32–33, 38
- Hardware, ix–x, 4–6. *See also* Input/output architecture of, 2, 79–101
 - Boolean logic and, 8–28
 - chips and, 85, 293–296 (*see also* Chips; Gates)
 - keyboard, 71
 - machine language and, 57–77
 - memory and, 81–82
 - modifications and, 278–279
 - operating systems and, 247–276
 - RAM, 42–47
 - screen, 70
 - sequential chips and, 41–55
 - simulators and, 299–306
 - stored program concept and, 79–80
- Hardware Description Language (HDL), x–xiii, 5–6, 93, 278
 - API notation and, 282
 - case sensitivity and, 283
 - chip logic and, 17–25, 281–296
 - compare file, 18
 - description of, 281
 - efficiency and, 288
 - hardware simulator and, 14, 17–25, 283–284
 - header section, 15
 - identifier naming and, 283
 - interfaces and, 15–16
 - logic building and, 39
 - parts section, 15
 - statement representation, 15
 - technical references for, 281–296
 - testing and, 16–17
 - visualized chip operations and, 292–296
- Hardware simulator, 14, 283–284
 - chip specifications and, 17–25
- Hash tables, 115, 226
- HDL. *See* Hardware Description Language
- Heap, 132–133
- High-level language, 4–6
 - Jack, 173 (*see also* Jack)
 - operating systems and, 248
 - program flow and, 153–159
 - subroutines and, 62, 112, 153–161, 181–190, 195, 209, 234–235
 - VM-Hack mapping and, 139–143, 161–168

- If-goto destination, 155
- If-x-then-y function, 10
- Immediate addressing, 61
- Incrementer chip, 33–39
- Indirect addressing, 61
- Inheritance, 195–196, 241–242
- Input/output (I/O), x
 - characters and, 259–263
 - device driver, 256–257
 - graphics, 257–263
 - Hack and, 62–77, 70–71, 98
 - keyboards, 261–263, 266
 - operating systems and, 256–270
 - screens, 265–266
 - standards and, 84
- Instructions, 116
 - addresses, 64–65, 108–110 (*see also* Addresses)
 - assembler and, 103–120
 - CISC, 98
 - compilers and, 122–127 (*see also* Compilers)
 - compute, 66–69, 108–110
 - decoding, 94–96
 - execution, 94–96
 - fetching, 86, 95–96, 98
 - labels and, 105
 - macros and, 117
 - memory and, 63, 82
 - RISC, 98
 - stack processing and, 130 (*see also* Stack processing)
 - subroutines and, 62, 112, 153–161, 181–190, 195, 209, 234–235
 - symbolic vs. binary, 104
 - variables and, 105
- Interfaces, 282, 284
 - HDL and, 15–16
 - logic gates and, 12
- Intermediate language (IL), 123
- Internal pins, 15–16
- Jack, 1, 4–5, 147, 165, 169, 197
 - abstract data types and, 175–179
 - API notation and, 175–176, 200, 215, 224
 - applications writing, 193–195
 - array handling, 175, 184–185, 191, 265, 269
 - binary code and, 174
 - classes and, 175–183, 187–193, 208, 248, 263–273
 - code generation and, 223–246
 - constants, 181–182
 - constructor for, 234–235
 - data types and, 183–185
 - evaluation order, 188
 - expression evaluation and, 187–188, 231–232
 - flow control and, 231–232
 - generic statements, 187
 - grammar and, 203, 207–215
 - identifiers, 181–182
 - inheritance and, 195–196, 241–242
 - I/O and, 191–193, 209–215, 265–266, 269–270
 - Java and, 174, 183, 196
 - keyboards and, 192–193, 266, 270
 - lexical analysis and, 202, 208
 - linked list implementation, 179–180
 - Main.main** function, 174–175
 - memory and, 193, 266–267, 270–271
 - modifications and, 277–278
 - as object-based language, 173, 195–196, 199
 - object handling and, 189–190, 228–231
 - operating system, 195, 197, 235, 253, 257–273
 - operator priority, 188
 - parsing and, 200, 217, 221
 - program elements in, 133–134
 - rational numbers and, 175–179
 - reserved words, 181–182
 - screens and, 192, 265–266, 269
 - simplicity of, 174
 - standard library of, 174, 190–193, 196, 263
 - strings and, 191, 264–265, 268–269
 - subroutines and, 181–190, 195, 209, 234–235
 - symbols and, 181–182, 238–239
 - syntax and, 181–182, 187, 207–221, 237–241
 - tokenizing and, 181, 202, 205, 208, 214–215, 219–221

- type conversions, 183, 241
- variables and, 181–187
- VM code and, 174, 233–235, 240
- void methods and, 235
- white space, 181–182
- XML and, 199–201, 211–218, 221
- Java, 17, 247, 253, 277
 - assembler and, 112
 - built-in chips and, 293, 296
 - Jack and, 174, 183, 196
 - stack arithmetic and, 122, 134
 - standard libraries, 147
 - VM and, 122, 134, 169
- Java Runtime Environment, 123, 146
- Java Virtual Machine (JVM), 121, 123, 146
- Jump, 109–110, 114
 - nested subroutine calling and, 153–159
 - specification, 61–62, 68–69, 96
- Keyboard input, 71, 84, 86, 89, 96
 - Jack and, 192–193, 266, 270
 - operating systems and, 266, 270
 - string reading and, 262–263
 - text handling and, 261–263
 - visualized chip operations and, 292–293
- Labels, 70, 105, 110, 116, 155, 159
- Last-in-first-out (LIFO) storage model, 124, 157
- Least significant bits (LSB), 30
- Lexical analysis, 202, 208
 - XML and, 199–201, 211–218, 221
- Lexical analysis (LEX) tool, 217
- Line drawing, 257–259
- Linked list, 179–180
- Linux, xiii, 277
- Load command, 60
- Logic
 - Boolean, 7–28 (*see also* Boolean logic)
 - control logic and, 94–95
 - decoding, 94–96
 - fetching, 95–96
 - HDL and, 281–296
 - instruction execution, 94–96
 - jumps, 61–62, 68–69, 96
 - stack processing and, 130 (*see also* Stack processing)
 - stored program concept and, 79–80
- Machine language, x
 - abstraction and, 81
 - addressing and, 60–61, 63
 - assembler and, 103–120
 - binary codes and, 59–60
 - commands and, 60–62
 - compilers and, 122–127 (*see also* Compilers)
 - conditional execution, 62
 - Hack, 62–77
 - instruction memory and, 82
 - labels and, 105
 - memory and, 58–62
 - mnemonic symbols, 59
 - processor and, 59
 - program size limits and, 106
 - registers and, 59
 - repetition and, 61–62
 - subroutine calling, 62
 - symbolic vs. binary, 104
 - syntax and, 60–62, 71–73, 104
 - testing and, 306–309
 - unconditional jump, 62
 - variables and, 105
 - VM and, 122–127 (*see also* Virtual Machine)
- Macro commands, 117
- Mapping
 - I/O operations and, 84–91
 - keyboard handling and, 262–263
 - memory segments and, 142–143
 - VM-to-Hack, 139–143, 161–168
 - VM-to-Jack, 233–235
- Memory, 2
 - addresses and, 45, 91 (*see also* Addresses)
 - allocation and, 253–254
 - arrays and, 227–228
 - clocks and, 42, 52–54
 - compilers and, 234
 - dynamic allocation and, 252–253
 - flip-flops and, 42–54

Memory (cont.)

- fragmentation and, 254, 256
- graphics and, 257–263
- Hack and, 63, 87–91, 96
- implementation and, 50–52
- improved allocation and, 254–256
- instruction, 63, 82
- Jack and, 193, 266–267, 270–271
- machine language and, 58–62
- mapped input/output (I/O) and, 84–91
- object handling and, 228–231
- operating systems and, 247, 252–256, 266–267, 270–271
- RAM, 42–45, 49–50 (*see also* Random access memory)
- registers and, 42–49
- stored program concept and, 79–80
- subroutines and, 62, 112, 153–161, 181–190, 195, 209, 234–235
- testing and, 310–311
- variable locations and, 106
- virtual segment mapping and, 142–143
- visualized chip operations and, 292
- VM and, 127–133
- von Neumann architecture and, 81

Mnemonics, 59, 108, 114

Multi-bit bus, 286–287

Multiplexors, x, 20–26

Multiplication, 249–250, 258–259

Multitasking, 247

Nand function, 2, 7, 10, 19, 27

Negative numbers, 31–32

Nested subroutine calling, 153–159

.NET infrastructure, 122, 123, 146–147

Network interface cards, 84

Newton-Raphson method, 251

Non-terminals, 203, 211

Nor function, 2, 10

Not function, 8–9, 26

Number base, 30

Object types, 183–184

Operating systems, ix–x, 4

- API notation and, 263, 267

- arrays and, 256, 265, 269
- classes and, 264–271
- description of, 247
- graphics and, 257–263
- hardware/software gaps and, 247
- initialization and, 267
- input/output (I/O) management, 256–266, 269–270
- Jack and, 195, 235, 263–273 (*see also* Jack)
- mathematical operations and, 248–252, 264, 268
- memory and, 247, 252–256, 266–267, 270–271
- program size limits and, 106
- screens and, 265–266, 269–270
- strings and, 252, 256, 264–265, 268–269
- Sys and, 267, 271

Operator priority, 188

Or function, 8–9, 20

- implementation of, 26
- multi-bit versions of, 21–23
- multi-way versions and, 23–25

Overflow, 30

Parsing, 2, 17, 60, 116

- assembler and, 112–114
- compilers and, 217 (*see also* Compilers)
- expression evaluation and, 187–188, 231–232
- grammar and, 203–207
- Jack and, 200, 203–207, 217, 221
- programming and, 107
- recursive descent, 204–206
- symbol-less, 114–115
- VM and, 144–146, 168

Pascal, 123, 146

P-code, 123, 146

Pins, 11, 15–16, 284–286, 290, 300

Pixel drawing, 257

Pointers, 69–70, 124, 131, 142, 161

Pop operation, 124, 130–132

Positive numbers, 31–32

Postfix notation, 231–232

Primitive gates, 11–13, 25–26

Program counter, 45, 84, 95

- Program flow
 - assembly language symbols and, 164
 - bootstrap code and, 165
 - calling protocol and, 160–161
 - LIFO model and, 157
 - nested subroutine calling and, 153–159
 - VM, 129–130, 133–134, 153–168
- Push operation, 124, 130–132
- Radix complement method, 31
- RAM. *See* Random access memory
- Random access memory (RAM), x, 6, 278–279
 - clocked chips and, 290–291
 - Hack platform and, 86, 96, 139–143, 161–168
 - implementation of, 52
 - memory management and, 253
 - operating systems and, 270–271
 - registers and, 49–50
 - sequential logic and, 42–47
 - testing and, 304–308, 311–312
 - VM and, 137–143, 161–168
- Rational numbers, 175–179
- Read-only memory (ROM) chips, 6, 85–86, 91, 278–279
- Read/write operations
 - memory and, 42–47
 - registers and, 48–49
- Recursive descent parsing, 204–206
- Reduced Instruction Set Computing (RISC), 98
- Registers, x, 2
 - addresses and, 45, 83–86
 - API specification and, 48–49
 - architecture of, 83–84
 - CPU and, 82
 - Hack and, 63–64, 69
 - implementation of, 52
 - machine language and, 59
 - memory and, 42–49
 - RAM and, 49–50
 - read-write operations and, 48–49
 - testing and, 304–305
 - virtual, 69
 - visualized chip operations and, 292
- Reserved words, 181–182
- Return address, 158
- Right Polish Notation (RPN), 231–232
- Rogers, Carl, 1
- RPN. *See* Right Polish Notation
- Screen output, 70, 84, 86, 89, 96
 - characters and, 259–263
 - graphics and, 257–263
 - Jack and, 192, 265–266, 269
 - operating systems and, 265–266, 269–270
 - resolution and, 257–258
 - visualized chip operations and, 292–296
- Segment index, 131–132, 135
- Selectors, 20
- Semantics, 199. *See also* Symbols; Syntax
 - data translation and, 224–231
- Sequential logic, 6
 - chip hierarchy and, 47–50
 - clocks and, 289–291
 - feedback loops and, 291–292
 - flip-flops and, 41–54
 - memory and, 42–47
 - time and, 45–47
- Signed binary numbers, 31–32
- Simulators, 101
 - testing and, 297, 299–306
- Square root function, 251
- Stack pointer, 124, 131, 142, 161
- Stack processing, 122
 - arithmetic and, 126–130
 - bootstrap code and, 165
 - heap structure and, 132–133
 - LIFO model and, 124, 157
 - memory and, 130–133
 - model of, 124–127
 - nested subroutine calling and, 153–159
 - pop operation, 124, 130–132
 - push operation, 124, 130–132
 - subroutines and, 153–159
 - VM-Hack mapping and, 139–143, 161–168
- Standard language library, 4

- Standard mapping, 141
- Store command, 60
- Stored program concept, 79–80
- Strings, 184
 - Jack and, 191, 264–265, 268–269
 - keyboard handling and, 262–263
 - operating systems and, 252, 256, 264–265, 268–269
- Subroutines, 62, 112
 - calling protocol and, 160–161
 - code generation and, 234–235
 - functional commands and, 153–159
 - Jack and, 181–190, 195, 209, 234–235
 - LIFO model and, 157
 - void, 235
- Switching technology, 2, 11
- Symbols
 - assembler and, 60, 104–106, 110–111, 114–116, 143, 164
 - function calling and, 160
 - Jack and, 181–182
 - labels, 70, 105, 110, 116, 155, 159
 - machine language and, 59–60, 69–70, 104
 - mnemonic, 59
 - resolution and, 105–106
 - variables and, 105
- Symbol tables, 103, 105, 115–116, 243
 - data translation and, 225–226
 - Jack and, 238–239
- Syntax, 5, 104
 - expression evaluation and, 187–188, 231–232
 - formal languages and, 201–202
 - non-terminals and, 203, 211
 - RPN, 231–232
 - semantics and, 199
 - terminals and, 203, 211
 - testing and, 301–304
 - XML and, 199–201, 211–218, 221
- Taylor series, 251
- Terminals, 203, 211
- Testing
 - chips, 299–306
 - emulators and, 297, 306–313
 - GUI and, 297–298
 - machine language and, 306–309
 - script commands and, 301–304
 - simulators and, 297, 299–306
 - test scripts, 16–17, 103–104
 - VM and, 310–313
- Text files, 2
- Time
 - clocks, 41–54, 289–292
 - counters, x, 45, 47–48, 50, 52, 84
 - sequential logic, 6, 42–54, 289–292
 - testing and, 297–313
- Tokens, 181
 - Jack tokenizing, 202, 205, 208, 214–215, 219–221, 237–241
 - syntax analyzer and, 207–213
- Transistors, 2, 11
- Translator program, 163–164
- Truth tables, 8–9
- Turing, Alan, 122
- Turing machine, 80–81
- Two-input Boolean functions, 9–10
- 2's complement method, 30
- Unconditional jump, 62
- Unix, 272
- Variables, 105, 116
 - argument, 234
 - fields, 183, 185, 226, 227
 - Jack and, 181–187
 - local, 183, 185–186, 227, 234, 253
 - parameter, 183, 185–186
 - scope and, 1, 225–226
 - static, 183, 185, 226–227, 234, 253
- Virtual Hardware Description Language (VHDL), 14
- Virtual Machine (VM)
 - advantages of, 121, 123–124
 - arithmetic and, 126–130, 135
 - array handling and, 137
 - bootstrap code and, 165
 - class and, 129
 - compilers and, 122–127
 - design suggestions for, 143

- emulators and, 121–122, 150–151
- examples of, 135–139
- functions and, 127, 129–130, 133, 135–139
- Hack mapping and, 139–143, 161–168
- heap structure and, 132–133
- high-level language and, 146–147, 153–154
- implementation, 55, 103, 112
- Jack and, 174, 233–235, 240
- language for, 122
- memory and, 127, 129–133
- modifications and, 277–279
- modularity and, 123–124
- nested subroutine calling and, 153–159
- object handling and, 137–139
- program flow and, 129–130, 133–134, 153–168
- stack processing and, 124–127
- subroutines and, 154–159
- symbols and, 143
- syntax and, 123
- testing and, 310–313
- translator, 121
- Virtual memory segments, 131
- Visual Basic, 147
- VM. *See* Virtual Machine
- Void methods, 235
- von Neumann architecture, 62, 79–81, 85

- White space, 108, 113, 181–182
- Windows, xiii, 277
- Working stack, 161

- XML, 199–201, 211–218, 221
- Xor function, 10, 20
 - implementation of, 26
 - multi-bit versions of, 21–23

- Yet Another Compiler Compiler (YACC), 217