

INDEX

Symbols

+ (addition) operator. *See* addition operator
+= (addition assignment) operator, 113
& (address) operator. *See* address operator
&& (AND) operator. *See* AND operator
= (assignment) operator. *See* assignment expressions
& (bitwise AND) operator, 67
&= (bitwise AND assignment) operator, 113
~ (bitwise complement) operator. *See* bitwise complement operator
| (bitwise OR) operator, 67
|= (bitwise OR assignment) operator, 113
?: (conditional) operator, 121–124. *See also* conditional expressions
-- (decrement) operator, 31–32, 33, 113
* (dereference) operator. *See* dereference operator
/ (division) operator. *See* division operator
/= (division assignment) operator, 113
== (equal to) operator. *See* equal to operator
> (greater than) operator. *See* greater than operator
>= (greater than or equal to) operator.
 See greater than or equal to operator
++ (increment) operator, 113
<< (left shift) operator, 67
<<= (left shift assignment) operator, 67
< (less than) operator. *See* less than operator
<= (less than or equal to) operator. *See* less than or equal to operator
* (multiplication) operator. *See* multiplication operator
*= (multiplication assignment) operator, 113
- (negation) operator. *See* negation operator
! (NOT) operator. *See* NOT operator
!= (not equal to) operator. *See* not equal to operator
|| (OR) operator. *See* OR operator
% (remainder) operator. *See* remainder operator
%= (remainder assignment) operator, 113
>> (right shift) operator, 67
>>= (right shift assignment) operator, 113
.
 (structure member) operator. *See* structure member operator

-> (structure pointer) operator. *See* structure pointer operator
[] (subscript) operator. *See* subscript operator
- (subtraction) operator. *See* subtraction operator
-= (subtraction assignment) operator, 113
^ (XOR) operator, 67
^= (XOR assignment) operator, 113

A

ABI (application binary interface), 184.
 See also System V x64 ABI
abstract declarators, 361–363
 abstract array declarators,
 395–396
abstract syntax tree (AST), 4, 10–14
 adding loop labels, 150
 adding type information, 252–253
 AST typing problem, 253
 additional resources, 21–22
 assembly, 18, 40
 constant representations in, 248,
 276, 306
 structure determines order of
 evaluation, 49
 TACKY, 36–37
add_edge function, 579–580
add instruction, 60, 62–63
 emitting, 66, 270
 fixing up, 64, 268
addition (+) operator, 47–50
 assembly for, 60, 62–63
 floating-point, 315
 parsing, 50–55
 pointer addition, 387–390
 TACKY for, 406–408
 type checking, 400, 472
 TACKY for, 58
 type checking, 254–255
addition assignment (+=) operator, 113
add_pseudoregisters, 632, 633, 637
AddPtr instruction, 406–408
 assembly for, 414–415
 omitting, 517
 structure member access with,
 514–517

- address (&) operator, 349, 353
 - assembly for, 376
 - constraints on, 364, 474
 - parsing, 354–355
 - TACKY for, 370–372, 374, 514, 516–517
 - type checking, 364–365
- address-taken analysis, 600–601
- AddrOf expressions, 354. *See also* address operator
 - array decay implemented with, 398–399, 409–410, 441
- addsd instruction, 311–312
 - fixing up, 337
- Advanced Compiler Design and Implementation* (Muchnick), 669
- aggregate types, 384
- Aho, Alfred V., 611, 670
- alias analysis, 601
 - additional resources, 611
- aliased variables, 599–602, 609, 637
- .align directive, 221, 238–239
- aligned_alloc function, 461
- allocated storage duration, 213, 461
- AllocateStack instruction, 40, 42, 44, 268
- AMD64. *See* x64 instruction set
- AND (&&) operator, 71–77
 - short-circuiting, 72
 - TACKY for, 75–77, 259
 - type checking, 255, 470
- and instruction, 323–325, 337, 341
- Appel, Andrew, 22, 670
- Apple Silicon, xxxv
- application binary interface (ABI), 184. *See also* System V x64 ABI
- arithmetic operations. *See also* names of individual operators
 - in assembly, 60–63
 - floating-point, 296
 - in assembly, 311–312, 315–316, 327–328
 - rounding behavior, 301
 - precedence value, 50, 55
 - type checking, 254–255, 476–477
 - usual arithmetic conversions, 254, 279–280, 308, 435
- ARM, xxvii, 672
- array declarators, 357–358
 - abstract, 361–362, 395–396
 - parsing, 394–396
- arrays, 384–399
 - alignment of, 415
 - assembly type of, 413
 - decay, 386–387
 - declaring, 384–385. *See also* array declarators
 - declarators
 - element comparison, 389–390
 - element type, 384
 - complete, 471–473
 - function declarations, array types in, 390–391
 - implicit conversion to pointers. *See* array-to-pointer decay
 - initializers, 385. *See also* compound initializers
 - string literals as, 425–426, 437–438, 440–441
 - memory layout, 385–386
 - multidimensional, 384–385, 386–389, 393
 - subscripting, 408–410
 - type checking, 398–399, 402–405, 471, 472–473
 - variable-length, 391
- array-to-pointer decay, 386–387
 - implemented with AddrOf, 398–399, 409–410, 441
 - sizeof operands not subject to, 462
- Arrow expression, 495. *See also* structure pointer operator
- ASCII, 10, 204, 426–427, 449–450
 - values of escape sequences, 429
- .ascii directive, 426–427, 449–450
- .asciz directive, 426–427, 449–450
- ASDL (Zephyr Abstract Syntax Description Language), 13–14, 22, 171
 - field names, 14
 - product types, 171
 - sum types, 171
- assembler directives, 5. *See also* entries for individual directives
- assemblers, xxviii, 5
 - GNU assembler (GAS), 268, 338
 - invoking, 7
 - LLVM assembler, 268, 338
- assembly code, xxvii, 4–7
 - arithmetic in, 60–63
 - floating-point, 311–312, 315–316, 327–328
 - AT&T vs. Intel syntax, 6, 244, 570
 - bitwise complement in, 26–27, 40–41
 - comments in, 20
 - comparisons in, 78–81, 82–83, 85–86
 - unsigned, 283–285, 287–288
 - floating-point, 317, 328
 - debugging, 675–697
 - with GDB, 677–687
 - with LLDB, 687–698
 - division in, 60–63
 - unsigned, 286, 288
 - floating-point, 310–336
 - function calls in, 161, 184–199, 312–315, 519–528
 - jumps in, 83–87

- linkage in, 168–169, 220–223
 - long integers in, 244–246, 261–264
 - negation in, 26–27, 40–41, 315–316, 327–328
 - storage duration in, 220–223
 - strings in, 426–429, 450
 - type conversions in, 244–245, 317–324
 - floating-point, 317–324, 328–329, 445
 - sign extension, 244–245, 263, 444
 - truncation, 245, 263, 444
 - zero extension, 286–288, 443–444
 - assembly generation, 4, 17–19
 - compiler passes in, 39
 - reference tables, 700–701, 704–711
 - assembly instructions, 5–6, 17–18. *See also names of individual instructions*
 - in assembly AST, 18
 - Streaming SIMD Extension, 310–312
 - suffixes, 6, 269, 311, 427, 443–444
 - assembly types, 261–262, 265–266
 - Byte, 443
 - ByteArray, 413
 - Double, 324
 - of eightbytes, 536–537
 - Longword, 261
 - Quadword, 261
 - suffixes for, 269–271, 340–341, 443
 - assignment expressions, 94–95
 - AST definition, 97–98
 - operator, 97, 101
 - precedence value, 103
 - parsing, 100–103
 - resolving variables in, 107
 - TACKY for, 110, 371–374, 516
 - type checking, 256, 368, 399
 - validating lvalues in, 107, 399
 - associativity, 50–51
 - AST. *See* abstract syntax tree
 - AT&T syntax, 6, 244, 570
 - automatic storage duration, 212–213, 217
 - arrays with, initializing, 440
 - automatic variables, 208
 - in the symbol table, 229–230
 - type checking, 233, 257
 - AVX instruction set extension, 318
- B**
- backend symbol table, 266–267
 - incomplete types in, 530, 546
 - register usage tracked in, 621–622, 635, 637, 647
 - return value passing information tracked in, 546, 550
 - top-level constants in, 327, 339, 446
 - backward analysis, 584, 604
 - iterative algorithm, 607–608
 - .balign directive, 238–289
 - Ballman, Aaron, 475
 - base pointer, 29
 - basic blocks, 576–578
 - creating, 578–579
 - empty, 583
 - unreachable, 581–582
 - basic source character set, 430
 - basic type, 356
 - Bendersky, Eli, 21, 68, 222, 611
 - binary expressions
 - AST definition, 48
 - formal grammar, 51, 52
 - parsing, 50–55
 - with precedence climbing, 51–55
 - with recursive descent, 50–51
 - operands, unsequenced, 58–59
 - TACKY for, 58
 - type checking, 254–255
 - binary fractions, 297
 - binary operators. *See* binary expressions *and names of individual operators*
 - bitwise AND (&) operator, 67
 - bitwise AND assignment (&=) operator, 113
 - bitwise complement (~) operator, 26
 - assembly for, 26–27, 40–41
 - parsing, 33–34
 - TACKY for, 36–37
 - token for, 31–32
 - type checking, 254, 308, 369, 435
 - bitwise OR (|) operator, 67
 - bitwise OR assignment (|=) operator, 113
 - blocks, 132, 135
 - compound statements as, 132
 - parsing, 136
 - resolving variables in, 136–139
 - block scope declarations, 208–217. *See also* scopes
 - invalid, 220
 - resolving identifiers in, 228–229
 - type checking, 232–233
 - Borgwardt, Michael, 343
 - break labels, 155–158
 - break statements, 146–148
 - annotating, 150, 151, 152–154
 - parsing, 149–150
 - TACKY for, 155–156
 - Briggs, Preston, 669
 - Briggs test, 656–659, 666–667
 - additional resources, 669–670
 - limits of, 661–663
 - .bss directive, 222
 - BSS section, 222
 - b suffix, 427, 443

build_graph function, 631–632
.byte directive, 450

C

caller-saved and callee-saved registers, 185, 648–649
 callee-saved registers in assembly AST, 620–621
 in graph coloring algorithm, 645–646
 saving and restoring, 187, 193–194, 196–197, 648–649
 tracking callee-saved register usage, 646–647
calling convention, 161, 184. *See also* System V x64 calling convention
call instruction, 186, 189–190
 emitting, 201–202
 generating, 198–199
calloc function, 461
case statements, 159
Cast expression, 248
 implicit type conversions represented by, 255
cast expressions. *See also* type conversions
 parsing, 247–249, 464–466
 pointer types as operands, 351–352
 TACKY for, 259–260, 281–283, 309–310, 375, 440, 479
 type checking, 254, 369, 402, 471, 505
 to void, 459, 471, 479
cdq instruction, 61–63, 262
 emitting, 66, 269
Chaitin, Gregory, 669
Chaitin-Briggs algorithm, 669–670
character constants, 424
 lexing, 429–431
 parsing, 433
 type of, 424
 UTF-8, 424
character types, 423–424
 assembly type, 443
 char, 423–424
 integer promotions, 424, 435
 signed char, 423–424
 specifiers, parsing, 433
 static initializers for, 436
 type conversions
 assembly for, 443–445
 TACKY for, 440
 unsigned char, 423–424
 wide, 424
char keyword, 429
char type, 423–424
 signedness, 424
 static initializer for, 436–438
Chu, Andy, 68
Ciechanowski, Bartosz, 345

Clang, xxxiv–xxxv, 4–5
 floating-point support, 296–297, 317–318, 344
 installing, xxxiv
 invoking with gcc command, xxxv, 4
 language extensions, 395, 401, 471
 System V ABI violation, 444–445
 void, treatment of, 474–475
classify_parameters function, 329–330, 534–536
classify_return_value function, 532–533, 537–538
classify_structure function, 533–534
cmp instruction, 79–80, 85–86, 262
 emitting, 90, 270
 fixing up, 88, 268
coalesce function, 665–666
 in build-coalesce loop, 663
code emission, 4, 19–20. *See also entries for individual instructions and language constructs*
 floating-point constants, 338–339
 function names, 201
 function prologue and epilogue, 43–44
 instruction size suffixes, 269–271, 340–341, 443
 local labels, 89, 339, 450
 non-executable stack note, 19
 @PLT suffix, 201–202
 reference tables
 Part I, 702–704
 Part II, 711–715
 Part III, 716–724
 register aliases, 88, 90, 203
 string literals, 449–450
color_graph function, 644–646
comisd instruction, 317, 324, 328
 emitting, 341
 fixing up, 337
common real type, 254–255, 279–280, 308, 435
comparisons, 78–83. *See also* pointer comparisons; relational operators
 floating-point, 317, 328
 unsigned, 283–286
compiler, xxvii
 stages, 3–4
compiler driver, xxviii, 7–8
 command line options, 8
 -c, 169–170
 --codegen, 8, 43
 --eliminate-dead-stores, 570
 --eliminate-unreachable-code, 569
 --fold-constants, 569
 -l, 301

- lex, 8
- optimize, 570
- parse, 8
- propagate-copies, 569
- S, 569
- tacky, 38
- validate, 109
 - generating assembly files, 569
 - generating object files, 169
 - linking shared libraries, 301
- Compiler Explorer (Godbolt), xxxvi
- Compilers: Principles, Techniques, and Tools*,
2nd edition (Aho et al.),
611, 670
- complete types, 461–462
 - required, 471–473, 477–478, 488, 491
 - structure types, 486–487
- compound assignment operators, 113–114
- compound initializers, 385
 - assembly for, 413, 418–419
 - AST definition, 393
 - not implemented, 391–392
 - parsing, 396
 - static, 404–405, 509–511
 - for structures, 492
 - TACKY for, 406, 410–411, 517–518
 - type checking, 403–405, 509–511
- compound literals, 391
- compound statements, 131
 - as blocks, 132
 - parsing, 135–136
 - resolving variables in, 136–139
 - scope determined by, 131–134
 - TACKY for, 140
- concrete syntax tree, 14
- conditional (?) operator, 121–124
- conditional expressions, 117
 - delimiter tokens, 118
 - parsing, 121–125
 - resolving variables in, 125–126
 - TACKY for, 127, 479–480
 - type checking, 256, 368, 467, 470,
476, 508
 - void operands, 459, 476
- conditional jump instructions. *See* jump
instructions (assembly);
jump instructions
(TACKY)
- conditional set instructions, 82–83
 - emitting, 88–90
 - generating, 85–86
 - SetCC, 85–86
- condition codes, 85–86, 285, 287–288
 - suffixes for, 90, 291
- conservative_coalesceable function,
666–667
- conservative coalescing, 653, 656, 670
- constant folding, 561, 573–576
 - combining with other
optimizations, 569
- constant propagation, 563
- constant strings, 425–426
 - in assembly, 428, 446
 - emitting, 449–451
 - in the symbol table, 437–439, 441
 - in TACKY, 441–442
 - type checking, 436, 437–439
- constant tokens, 8–10
 - character, 429–431
 - floating-point, 302–303
 - rounding, 300
 - long integer, 247
 - regular expressions, 304
 - unsigned integer, 275
 - unsigned long integer, 275
- continue labels, 155–158
- continue statements, 146–150
 - annotating, 150, 151, 152–154
 - parsing, 149–150
 - TACKY for, 155–156
- control-flow graphs, 570, 576–581
- control-flow protection, 5
- controlling expression, 118–119
 - loops, 144–145
 - type checking, 352, 470
- control structures, 117
- conversion ranks, 279
- convert_by_assignment function, 368, 469,
504–505
- convert_function_call function, 197–199,
263, 331–333, 538–541
- convert_unop function, 37–38
- convert_val function, 198
- Cooper, Keith, 669–670
- copy_bytes_from_reg function,
543–544
- copy_bytes_to_reg function, 541–543
- CopyFromOffset instruction, 512–513
 - assembly for, 532, 548
 - structure member access with,
513–514, 516
- Copy instruction, 75–77, 110
 - assembly for, 86
 - with non-scalar operands, 531
 - type conversions with, 282, 574
- copy propagation, 563–564, 585–602.
 - See also* reaching copies
analysis
 - additional resources, 611
 - combining with other
optimizations, 569
 - with Part II TACKY programs,
599–602
 - rewriting instructions, 598–599, 602

- CopyToOffset instruction, 406–407
 - assembly for, 414
 - with non-scalar operands, 531–532
 - initializing aggregate objects with, 410–411, 440–441, 517–518
 - structure member access with, 514, 516
- Cordes, Peter, 445
- cqo instruction, 269
- C standard, xxxvi
 - & (address) operator applied to dereferenced pointer, 353
 - array decay, 386–387
 - basic source character set, 430
 - C17, xxxvi–xxxvii, 164
 - C23 standard, xxxvi–xxxvii
 - checked integer arithmetic, 82
 - decimal floating-point types, 300
 - empty initializers, 519
 - empty parameter lists, 164
 - memset_explicit, 565
 - old-style function definitions removed, 164
 - u8 character constants, 424
 - declarators, 358
 - escape sequences, 429
 - evaluation order, 59
 - short-circuiting operators, 72
 - floating-point types, 296
 - for loops, missing controlling expression in, 158
 - implementation-defined behavior, 245
 - incomplete types, 461
 - linkage, 167–168, 210–212
 - lvalues, 348
 - observable behavior, 560
 - preprocessing tokens, 303
 - return statements, missing, 111–112
 - storage duration, 212
 - strict aliasing rules, 352
 - structure member declarations, 488–489
 - temporary lifetimes, 508
 - type conversions, 244, 274
 - types of integer constants, 278
 - type specifiers, 278
 - undefined behavior, 80, 91, 107, 112
 - unsigned wraparound, 285
 - usual arithmetic conversions, 254, 279–280
 - void, 458, 473–475
- .cstring directive, 428, 450
- Cuoq, Pascal, 344
- cvtsi2sd instruction, 320–321, 324, 329
 - in character type conversions, 445
 - fixing up, 337
- cvtttsd2si instruction, 317–318, 324
 - in character type conversions, 445
 - fixing up, 337
- ## D
- dangling else ambiguity, 120–121
- data-flow analysis, 563, 584–585
 - additional resources, 611
 - liveness analysis, 604–609
 - of assembly programs, 633–636
 - reaching copies analysis, 589–598
- Data operand, 236–238
 - for constants, 326, 339
 - offset, 529, 550, 551
- data section, 221
- Dawson, Bruce, 344
- dead store elimination, 564–565, 603–609
 - combining with other optimizations, 569
 - liveness analysis, 604–609
 - iterative algorithm, 607–608
 - meet operator, 606–607
 - transfer function, 605–606
 - with Part II TACKY programs, 608–609
 - security impact, 566
- DeallocateStack instruction, 194–195, 198–199, 202, 264
- debuggers, xxxiv, 675–698
 - GDB (GNU debugger), xxxiv–xxxv, 677–687
 - LLDB (LLVM debugger), xxxv, 687–698
- declarations, 94, 162–163, 208–220. *See also* function declarations; variable declarations
 - vs. definitions, 214–216
 - hidden, 133
 - linkage, 166–168, 209–212
 - scope, 131–134, 208–209
 - vs. statements, 98–99
 - structure type, 486–491
- declarators, 356–361
 - abstract, 361–363, 395–396
 - array, 357–358, 394–396
 - in C standard, 358
 - function, 357
 - parsing, 358–361, 362–363, 394–396
 - pointer, 356, 361
- decrement (--) operator, 31–32, 33, 113
- default statements, 159
- degree, 638
- degree < *k* rule, 638
- dereference (*) operator, 349–350
 - & operator applied to result, 353
 - parsing, 354–355

- pointers to void as operands, 473–475
- TACKY for, 371–374
- type checking, 364–365
- DereferencedPointer construct, 372–374, 408, 410, 515–517
- derived types, 354
- disjoint-set data structures, 663–664
- div instruction, 286, 287–288
 - fixing up, 290
- division (/) operator, 47–48
 - assembly for, 60–63
 - floating-point, 315, 327
 - unsigned, 286, 288
 - parsing, 50–55
 - TACKY for, 58
 - type checking, 254–255, 369
- division assignment (/=) operator, 113
- divsd instruction, 315
 - DivDouble, 324–325
 - fixing up, 337
- do statements, 144, 148–151, 152–155, 156
- Dot operator, 495. *See also* structure member operator
- .double directive, 312, 338–339
- double extended precision floating-point format, 299
- double-precision floating-point format, 297–299
- double rounding error, 306
 - additional resources, 344
 - type conversion with, 320–323
- DoubleToInt instruction, 309–310
 - assembly for, 317–318
- DoubleToUInt instruction, 309–310
 - assembly for, 318–320
- double type, 295–301. *See also* floating-point constants; floating-point operations
 - alignment, 336
 - assembly type, 324
 - conversions. *See* conversions to and from double *under* integer types; double *under* type conversions
 - in function calls, 312–315, 329–333
 - representation, 297–299
 - precision, 301
 - rounding, 299–301
 - size, 336
 - specifier, 302, 305, 306–307
 - static initializers for, 308–309, 340
 - type checking, 308–309
- Drysdale, David, 21
- D’Silva, Vijay, 611
- dynamic linkers, 202

E

- EAX register, 5–6, 40–41, 60–62, 185, 193, 525
- EBNF. *See* Extended Backus-Naur Form notation
- EDX register, 60–64, 185, 525
- effective type, 352
- Elements of Computing Systems, The* (Nisan and Schocken), 45
- ELF (Executable and Linkable Format), 201
- else clause, 118–121, 126–127
 - dangling else ambiguity, 120–121
- Engineering a Compiler*, 2nd edition (Cooper and Torczon), 669–670
- equal to (==) operator, 71–74
 - assembly for, 85–87
 - floating-point, 317, 328
 - pointer comparisons, 352
 - TACKY for, 75–76, 77
 - type checking, 254–255, 366–367, 476–477
- escape sequences, 429–431
 - in assembly, 449–450
- Executable and Linkable Format (ELF), 201
- executable stacks, 19
 - additional resources, 22
- expect function, 16
- expressions, 14
 - converting to TACKY, 38
 - full, 374
 - lvalue vs. non-lvalue, 348
 - parsing, 34. *See also* precedence climbing
 - resolving variables in, 107
 - type checking, 251–256
 - result types, 251
 - void, 459–460
- expression statements, 95, 98, 110
- Extended Backus-Naur Form (EBNF)
 - notation, 15
 - optional sequences, 101
 - repeated sequences, 100
 - at least once, 225
- external linkage, 167–168, 209–211
- external variables, 208
 - resolving, 227–229
- extern specifier, 207, 208, 210–212, 213, 214–217
 - on declarations with incomplete types, 474, 505
 - in identifier resolution, 228–229
 - parsing, 225–226
 - in the type checker, 230–233
- extra credit features, xxxii–xxxiii
 - bitwise operators, 67
 - case statements, 159

- extra credit features (*continued*)
 - compound assignment operators, 113–114
 - decrement (--) operator, 113
 - default statements, 159
 - goto statements, 128
 - increment (++) operator, 113
 - labeled statements, 128
 - NaN, 342–343
 - switch statements, 159
 - union types, 552–553

F

- fetch-execute cycle, 84
- file scope, 207–208
- file scope variable declarations, 208–217
 - resolving identifiers in, 227–228
 - type checking, 231–232
- Finley, Thomas, 45
- floating-point constants
 - assembly for, 311–312
 - emitting, 338–339
 - generating, 324–327
 - local labels, 312, 326–327, 339
 - AST representation, 305–306
 - hexadecimal, 302, 338–339, 345
 - lexing, 302–304
 - rounding decimal constants to, 300, 306
- floating-point formats, 296–299
 - decimal, 300
 - double extended precision, 299
 - double-precision, 297–298
 - IEEE 754, 296–299
 - single-precision, 299
- The Floating-Point Guide (website), 343
- floating-point instructions. *See* Streaming SIMD Extension instructions
- floating-point operations
 - arithmetic operations, 296
 - in assembly, 311–312, 315–316, 327–328
 - rounding behavior, 301
 - comparisons
 - in assembly, 317, 328
 - with NaN, 299, 317, 342
 - with negative zero, 298, 317
 - with Streaming SIMD Extension instructions, 310–312
 - type conversions
 - in assembly, 317–324, 328–329, 445
 - rounding behavior, 300–301
 - in TACKY, 309–310, 440
- floating-point registers. *See* XMM registers
- floating-point values
 - assembly type, 324

- in function calls, 312–315, 329–333
- representation, 297–299
 - gaps between, 300–301, 322, 344
 - normalized floating-point numbers, 298
 - precision, 301
 - special values, 298–299
 - infinity, 298
 - NaN, 299, 342–343
 - negative zero, 298
 - subnormal numbers, 298
- float type, 295, 299
- Fog, Agner, 553–554
- formal grammar, 14–15
 - ambiguity, 50, 120
 - for binary expressions, 51, 52
 - left recursion, 50
 - for unary expressions, 33, 397, 465
- for statements, 144–145, 148–151, 152, 154, 157–158
 - headers, restrictions on, 172, 220
 - missing controlling expression in, 158
- forward data-flow analysis, 584
- free function, 460–461
- Friedl, Steve, 358
- frontend symbol table, 266. *See also* symbol table internal to compiler
- full expressions, 374
- FunCall instruction, 182–183
 - assembly for. *See* in assembly *under* function calls
 - in liveness analysis, 605–606
 - optional destination, 479, 482
 - in reaching copies analysis, 591–592, 601–602
- function calls, 165
 - arguments, 165
 - in assembly, 161, 184–194, 197–199
 - with floating-point values, 312–315, 329–333
 - with quadword arguments, 263
 - with structures, 519–528, 532–544
 - with void return type, 482
 - AST definition, 171
 - parsing, 172–173
 - resolving identifiers in, 175–176
 - TACKY for, 182–183, 479
 - type checking, 179, 181–182, 256
- function declarations, 162–163
 - array types in, 390–391
 - AST definition, 171, 224, 247–248
 - in identifier resolution, 174, 176–178
 - incomplete types in, 505

- linkage, 166–169, 209–212
- parsing, 172–173, 226–227
- type checking, 179–181, 230–231, 257, 402–403
- with void parameters, 466
- function definitions, 162–163
 - in assembly, 195
 - accessing function parameters, 195–197
 - allocating stack space, 200
 - converting to TACKY, 110–111, 182–183
 - nested, 163
 - old-style, 164
- function pointers, 164, 359–361, 364
- function prologue and epilogue, 26–27, 29–31
 - emitting, 43–44
- functions, 161–169
 - arguments, 165
 - calling convention, 161, 184–194, 312–315, 519–528
 - declarators, 357
 - parameters, 162–163, 165, 177
 - types, 178–179, 247–248
 - variadic, 191
 - with void return types, 458, 469–470, 479, 482

G

- GAS (GNU assembler), 268, 338
- GCC, xxxiv–xxxv, 4–5
 - floating-point support, 296–297, 317–318, 344
 - implementation-defined type
 - conversion in, 245
 - installing, xxxiv–xxxv
 - language extensions, 395, 401, 471
 - narrow arguments, treatment of, 445
 - optimizations, 27, 558–559
 - UndefinedBehaviorSanitizer, 672
 - void, treatment of, 474–475
- gcc command, 4–5
 - invoking Clang with, xxxv, 4
- GDB (GNU debugger)
 - debugging assembly code, 677–687
 - installing, xxxiv–xxxv
- general-purpose registers, 311
- George, Lal, 670
- George test, 659–663
 - additional resources, 670
 - limits of, 661–663
- GetAddress instruction, 370–372, 374
 - alias analysis and, 599–601
 - assembly for, 376
- get_common_pointer_type function, 366–368, 468

- get_common_type function, 254–255, 280, 308, 435
- Ghuloum, Abdulaziz, xxvi
- Gibbons, Phillip, 611, 670
- global offset table (GOT), 223
- global symbol, 5, 168–169
- .globl directive, 5, 20, 168–169, 221, 238
- GNU assembler (GAS), 268, 338
- Goldberg, David, 343
- goto statements, 128
- graph coloring, 622–646
 - algorithm, 638–646
 - degree $< k$ rule, 638
 - optimistic coloring, 669
 - spilling registers, 627–630, 642–644, 646
- greater than ($>$) operator, 71–74
 - assembly for, 85–87
 - floating-point, 317, 328
 - unsigned, 287–288
 - pointer comparisons, 389–390
 - TACKY for, 75–76, 77
 - type checking, 254–255, 401
- greater than or equal to ($>=$) operator, 71–74
 - assembly for, 85–87
 - floating-point, 317, 328
 - unsigned, 287–288
 - pointer comparisons, 389–390
 - TACKY for, 75–76, 77
 - type checking, 254–255, 401

H

- Hailperin, Max, 670
- “Hello, World!” program, 204, 451–453
- hexadecimal floating-point constant, 302, 338–339, 345
- Hilfinger, Paul, 611
- Hyde, Randall, 199

I

- identifier resolution, 174–178, 227–229, 364.
 - See also* variable resolution
 - renamed from variable resolution, 174
 - structure tags, 498–500
- identifiers, 8
 - autogenerated, 37–38, 105–106
 - lexing, 8–10
 - linkage of, 167–169, 209–212
 - scope of, 131–134, 208–209
 - structure tags, 486–488, 489–490
 - in the symbol table, 179–181, 229–233, 257–258
 - type of, 178–179
- idiv instruction, 60–65, 262
 - emitting, 66, 270

- IEEE 754 standard, 296–299
 - additional resources, 343–344
 - double-precision format, 297–299
 - floating-point formats, 296–299
 - rounding modes, 299
 - if statements, 117–121
 - AST definition, 118–119
 - parsing, 118–121
 - dangling else ambiguity, 120–121
 - resolving variables in, 125–126
 - TACKY for, 126–127
 - immediate values, 18, 268
 - as function arguments, 198–199
 - size inferred, 266
 - Imm operand, 18–20
 - implementation-defined behavior, 245–246
 - char signedness, 424
 - ptrdiff_t, 400
 - rounding behavior, 307
 - size_t, 460
 - source character set, 430
 - type conversions, 245, 352
 - imul instruction, 60, 62–63
 - emitting, 66, 270
 - fixing up, 64–65, 268
 - incomplete types, 461–462
 - in backend symbol table, 530, 546
 - in function declarations, 505
 - pointers to, 461–462, 471–472, 473, 505
 - structure types, 486–487, 505–506
 - type checking, 471–473, 505–506
 - increment (++) operator, 113
 - indeterminately sequenced evaluations, 58–59, 82
 - indexed addressing, 412
 - Indexed operand, 412–415
 - emitting, 419
 - initializers, 94. *See also* compound
 - initializers; static initializers
 - array, 385, 425, 440
 - string literals as, 425–426, 437–438, 440–441
 - invalid, 220
 - resolving identifiers in, 105–106
 - structure type, 492
 - TACKY for, 110, 440–441
 - using variables in their own, 106–107
 - instruction fix-up pass, 42–43
 - scratch registers, 42, 64–65, 325, 337
 - instruction pointer (IP), 84. *See also* RIP
 - register
 - instruction register, 84
 - INTEGER class, 519
 - integer constants, 6, 8. *See also* character
 - constants
 - parsing, 250–251, 278
 - regular expressions for, 304
 - representation in the abstract syntax tree, 248, 276
 - tokens for, 8, 247, 275, 304
 - integer overflow, 78–82
 - integer promotions, 424, 435
 - integer types
 - common real type, 254–255, 279–280
 - conversions between, 244–245, 274–275, 279–280
 - in assembly, 244–245, 263, 286–288, 443–444
 - conversion rank, 279
 - in TACKY, 259–260, 281–283
 - conversions to and from double
 - in assembly, 317–324, 328–329, 445
 - rounding behavior, 300–301
 - in TACKY, 309–310, 440
 - parsing specifiers, 249–250, 277–278
 - Intel 64 Software Developer’s Manual, xxxvi, 344
 - Intel syntax, 6
 - interactive devices, 560
 - intermediate representations (IRs), 35–36
 - control-flow graphs, 570, 576–581
 - internal linkage, 209–212
 - interprocedural optimizations, 570
 - intraprocedural optimizations, 570
 - IntToDouble instruction, 309–310
 - assembly for, 320
 - int type
 - alignment of, 246
 - size of, 244
 - static initializer for, 257
 - IP (instruction pointer), 84
 - iterated register coalescing, 663
 - iterative algorithms, 585
 - copy propagation, 593–599
 - dead store elimination, 607–608
- ## J
- je instruction, 84–85
 - JmpCC instruction, 85–86, 89
 - jmp instruction, 83–84, 85
 - jne instruction, 85
 - Jones, Joel, 21
 - JumpIfNotZero instruction, 75–76
 - JumpIfZero instruction, 75–76
 - Jump instruction, 75–76
 - jump instructions (assembly), 83–85
 - in assembly generation, 85–87
 - conditional, 84–85
 - emitting, 89
 - je, 84–85
 - jmp, 83–84, 85

- local variables (*continued*)
 - on the stack, 29
 - storage duration, 214
 - undefined behavior, 96
 - logical operators, 71. *See also names of individual operators*
 - parsing, 73–75
 - precedence values, 74
 - short-circuiting, 72
 - TACKY for, 75–77, 259
 - tokens for, 72
 - type checking, 255, 369, 470
 - .long directive, 221
 - long double type, 295, 299
 - long integers, 243. *See also long type*
 - in assembly, 244–246, 261–264
 - assembly type, 261
 - unsigned long type, 273–281
 - long keyword, 247
 - long type, 243
 - alignment, 246
 - constants of, 247–248, 250–251, 254
 - conversions, 244–245, 274–275
 - size, 244
 - static initializer, 257–258
 - Longword assembly type, 261–262
 - longwords, 6, 244, 267, 270
 - l suffix, 60
 - loops, 144–148
 - analysis, 638, 670
 - do, 144–146, 148–151, 154, 156
 - effect on spill cost, 638
 - enclosing loops, 146, 151, 153
 - for, 144–145, 148–151, 152, 154, 157–158
 - labeling, 150, 152–155
 - resolving variables in, 151–152
 - TACKY for, 155–158
 - while, 144, 148–150, 151–155, 157
 - .L prefix, 89. *See also local labels*
 - lvalues, 95, 348, 349–350
 - conversion, 348, 350
 - string literals, 425, 436
 - structure members, 491, 507–508
 - in TACKY, 371–374, 515–517
 - validating, 107, 364, 365, 399, 436, 507–508
 - and void, 474–475
- M**
- machine-dependent optimizations, 558
 - machine-independent optimizations, 557–558
 - constant folding, 561, 573–576
 - copy propagation, 563–564, 585–602
 - dead store elimination, 564–565, 603–609
 - unreachable code elimination, 561–562, 581–584
 - machine instruction, 5–6
 - Mach-O file format, 201
 - macOS, xxxiv
 - local label prefix, 89
 - platform-specific directives, 221, 238–239, 312, 339, 428, 450
 - prefix for user-defined names, 19, 201, 238
 - setup instructions, xxxiv–xxxv
 - main function, 4, 6, 169
 - make_tacky_variable function, 261
 - make_temporary function, 37–38
 - malloc function, 460
 - mantissa, 297
 - meet operator, 585
 - liveness analysis, 606–607
 - of assembly programs, 633–634
 - reaching copies analysis, 592–593
 - member access operators. *See* structure member operator; structure pointer operator
 - MEMORY class, 519
 - memory management functions, 457–458, 460–461
 - aligned_alloc, 461
 - calloc, 461
 - free, 460–461
 - malloc, 460
 - realloc, 461
 - Memory operands, 375–379
 - mov instruction, 5–6, 18, 261–262
 - emitting, 20
 - fixing up, 42, 268, 270
 - movsd instruction, 311–312
 - movsx instruction, 244–245, 261, 263, 444
 - emitting, 269, 450–451
 - fixing up, 267
 - sign extension with, 263
 - MovZeroExtend instruction, 287–289, 443–444
 - in conversions to double, 329, 445
 - emitting, 450–451
 - fixing up, 290, 449
 - movz instruction, 443, 449
 - Muchnick, Steven, 669
 - mulsd instruction, 315
 - emitting, 341–342
 - fixing up, 337
 - multidimensional arrays, 384–385, 386–389, 393
 - multiplication (*) operator, 47–48
 - assembly for, 60, 62–63
 - floating-point, 315, 327

- parsing, 50–55
 - TACKY for, 58
- type checking, 254–255, 369

multiplication assignment (`*`) operator, 113

Myers, Joseph, 218

N

NaNs (not-a-number), 299, 342–343

- comparing, 299, 317, 342
- extra credit, 342
- quiet, 299
- signaling, 299

negation (`-`) operator, 26

- assembly for, 26–27, 40–41
 - floating-point, 315–316, 327–328
- parsing, 33–34
- TACKY for, 36–38
- token for, 31–32
- type checking, 254, 369, 435

negative infinity, 298

negative zero, 298, 317, 326, 340

neg instruction, 26–27, 40–41, 44

- emitting, 44, 270

nested function definitions, 163

Nisan, Noam, 45

non-scalar types, 470–471

non-terminal symbols, 15

NOT (`!`) operator, 71–74

- assembly for, 86, 328
- TACKY for, 75–76, 77
- type checking, 254, 369, 470

not equal to (`!=`) operator, 71–74

- assembly for, 85–87
 - floating-point, 317, 328
- pointer comparisons, 352
- TACKY for, 75–76, 77
- type checking, 254–255, 366–367

not instruction, 26–27, 40–41

- emitting, 44, 270

null pointers, 351–352

- comparisons, 352
- constants, 351, 366–368, 401
 - as static initializers, 369

null statements, 98, 110

O

object code, xxviii

object files, xxviii, 5

- generating, 169–170
- sections of, 5
 - BSS, 222, 340, 418
 - data, 221–222
 - read-only data, 311–312, 339
 - text, 5

objects, 348

- lifetime of, 212–213, 461, 508

observable behavior, 558–560

OF. *See* overflow flag

optimistic coloring, 669

optimization pipeline, 570–573, 600–601

optimizations. *See also* machine-independent optimizations *and* entries for individual optimizations

- constant folding, 561, 573–576
- copy propagation, 563–564, 585–602
- dead store elimination, 564–565, 603–609
- interprocedural, 570
- intraprocedural, 570
- machine-dependent, 558
- safety of, 558
- security impact, 564–565
- unreachable code elimination, 561–562, 581–584

optimize function, 570–573, 601

- termination, 572–573

OR (`||`) operator, 71–77

- short-circuiting, 72
- TACKY for, 75–77, 259
- type checking, 255, 470

or instruction, 323–325, 337, 341

overflow, 78–82

overflow flag (OF), 78–80, 83

- not applicable, 284–285, 317

P

packed operands, 310, 316

parameter-passing registers, 185, 312

parameters, 162–163, 165, 177, 195–197

parity flag (PF), 342

parse_exp function, 16, 34, 51–57, 101–102, 124

parser generators, 11

parsers, 4, 10–17. *See also* recursive descent parsing

- handwritten, 11
- precedence climbing, 51–57
- predictive, 16

parse_type function, 249–250, 277, 307, 433, 466

pattern matching, xxxiii–xxxiv

Payer, Mathias, 611

PF (parity flag), 342

phase ordering problem, 573

PlainOperand construct, 372–374

PLT (procedure linkage table), 201–202

pointer analysis, 601

pointer arithmetic, 387–390

- addition, 387–390
 - assembly for, 414–415
- relationship to subscript operator, 387–389
- subtraction, 388–390

- pointer arithmetic (*continued*)
 - TACKY for, 406–408
 - type checking, 400–401, 472
 - undefined behavior, 388, 390
 - pointer comparisons, 352–353, 389–390
 - assembly for, 377, 415
 - type checking, 366–367, 401
 - TACKY for, 375, 408
 - pointers, 347, 349–353. *See also* null pointers
 - pointers
 - conversions to and from, 351–352, 460
 - pointers to void, 467–469
 - TACKY for, 375
 - type checking, 367–369, 467–469
 - declarators, 356, 361
 - dereferencing, 349–350
 - to incomplete types, 461–462, 471–472, 473, 505
 - operations on, 349–353
 - PointerInit, 437, 450
 - referenced types, 354
 - static initializers for, 369–370, 428–429, 437, 438–439
 - type checking, 364–370, 400–402, 467–469, 471–472
 - types
 - AST definition, 354
 - parsing, 356–364
 - pop instruction, 27–28, 30–31, 620–621, 648
 - emitting, 44, 649
 - positive infinity, 298
 - postfix operators, 113, 396–397, 498
 - postorder traversals, 49
 - precedence climbing, 47, 51–57
 - additional resources, 68
 - combined with recursive descent parsing, 52–53
 - example of, 55–57
 - pseudocode for, 54
 - with assignment operator, 102
 - with conditional operator, 124
 - right-associative operators, 101–102
 - precedence values
 - arithmetic operators, 55
 - assignment operator, 103
 - conditional operator, 123
 - logical operators, 74
 - relational operators, 74
 - precoloring register interference graphs, 625
 - predictive parsers, 16
 - prefix operators, 113, 396
 - preprocessor, xxviii, 7
 - pretty-printer, 17
 - procedure linkage table (PLT), 201–202
 - production rule, 15
 - PseudoMem operand, 412–414
 - replacing, 417–418
 - Pseudo operand, 40–42
 - pseudoregisters, 40–41
 - replacing, 42, 237, 267
 - push instruction, 27–30, 194–195
 - emitting, 43, 203
 - fixing up, 378–379
 - passing arguments with, 188–189, 198–199, 263, 332
 - putchar function, 204
 - puts function, 451–453
 - Python, xxxiv
- ## Q
- .quad directive, 246, 270, 428, 450
 - Quadword assembly type, 261–262
 - quadwords, 6
 - arguments, 263
 - instructions, 244, 261–262
 - suffix, 6, 269
 - pseudoregisters, 267
 - static, 246
- ## R
- RAX register, 5–6, 40–41, 60–62, 185, 193, 525
 - RBP register, 29–30, 375
 - RDX register, 60–64, 185, 525
 - reaching copies, 589
 - reaching copies analysis, 584, 589–599
 - iterative algorithm, 593–599
 - meet operator, 592–593
 - transfer function, 589–592, 601–602
 - read-only data section, 311–312, 339
 - realloc function, 461
 - recursive descent parsing, 15–17
 - with backtracking, 17
 - of binary operations, 50–51
 - combined with precedence climbing, 52–53
 - dangling else ambiguity handled, 120–121
 - of declarators, 359
 - precedence and associativity, issues with, 50–51
 - Regan, Rick, 344–345
 - Regehr, John, 91
 - register allocation, 613–619. *See also* spilling
 - additional resources, 669–670
 - graph coloring, 622–646
 - algorithm, 638–646
 - degree < *k* rule, 638
 - handling multiple types, 631, 637
 - register coalescing, 618–619, 651–668
 - iterated, 663
 - top-level algorithm, 630

- register coalescing, 614, 618–619, 651–653, 663–667
 - conservative coalescing, 653, 656, 670
 - Briggs test, 657–659, 661–663, 666–667, 669–670
 - George test, 659–663, 666–667, 670
 - iterated, 663
 - updating the graph while, 653–656, 663, 666
 - register interference graphs, 622–626
 - building, 631–637
 - coloring, 622–625, 638–646
 - precoloring, 625
 - detecting interference, 623–624, 626–627
 - updating, 653–656, 666
 - registers, 5–6
 - aliases, 40
 - assembly AST definition, 18, 40, 62, 620–621
 - parameter-passing registers, 195
 - RBP register, 375
 - RSP register, 264
 - XMM registers, 325
 - caller-saved and callee-saved, 185, 620–621, 645–646, 648–649
 - general-purpose, 311
 - instruction, 84
 - parameter-passing, 185, 195, 312
 - XMM, 311–312, 316, 325
 - Reg operand, 40
 - relational operators, 71. *See also names of individual operators*
 - assembly for, 85–88
 - floating-point, 317, 328
 - unsigned, 285, 287–288
 - parsing, 73–75
 - precedence values, 74
 - pointer operands, 352, 389–390
 - TACKY for, 75–76
 - tokens for, 72
 - type checking, 254–255, 366–367, 401, 476–477
 - remainder (%) operator, 47–48
 - assembly for, 60–63
 - unsigned, 288
 - parsing, 50–55
 - TACKY for, 58
 - type checking, 254–255, 308, 369
 - remainder assignment (%=) operator, 113
 - replacing pseudoregisters, 42
 - with different types, 267
 - PseudoMem operands, 417–418
 - with static storage duration, 237
 - ret instruction, 6, 18
 - emitting, 20, 44
 - return statements, 4
 - assembly for, 18, 333, 482, 545–546
 - AST definition, 13–14
 - missing, 111–113, 458
 - parsing, 14–17
 - without return values, 458, 469–470, 479, 482
 - TACKY for, 36–38, 479
 - type checking, 256–257, 469–470
 - return values, 4–6, 14
 - absent, 458, 469–470, 479, 482
 - classifying, 537–538. *See also* classify_return_value function
 - of floating-point type, 312–313, 333
 - of structure type, 525–528, 537–541, 545–546
 - rewrite_coalesced function, 667–668
 - RFLAGS register, 78
 - right-associative operators, 50, 101–102, 123–124
 - right shift (>>) operator, 67
 - right shift assignment (>>=) operator, 113
 - RIP register, 83–84, 189–190, 222
 - RIP-relative addressing, 222, 223, 311, 376, 529
 - Data operand, 236–238
 - Ritchie, Dennis, 390
 - .rodata directive, 311–312, 339, 428, 450
 - Rosetta 2, xxxv
 - rounding modes, 299, 320
 - round-to-nearest, ties-to-even, 299, 321, 575
 - rounding to odd, 322–324
 - RSP register, 27–30, 43–44, 185, 264
- ## S
- safety of optimizations, 558
 - scalar types, 384, 470–471
 - Schocken, Shimon, 45
 - scopes, 131–134, 208–209
 - block scope, 208
 - compound statements determine, 131–134
 - file scope, 207–208
 - vs. storage duration, 213
 - .section .rodata directive, 311–312
 - semantic analysis, 93, 103–104
 - identifier resolution (aka variable resolution), 104–109, 174–178, 227–229
 - loop labeling, 150, 152–155
 - type checking, 178–182, 251–258
 - Serra, Christopher, 22
 - SetCC instruction, 85–87
 - emitting, 89–90
 - set_up_parameters function, 544–545
 - SF (sign flag), 78–80, 83

- shift left (shl) instruction, 529–530, 541–543, 551
- shift right (shr) instruction, 320–321, 323–325, 529
 - two-operand form, 529, 543, 551
- short-circuiting operators, 72, 76–77
- signed char type, 423–424
- signed integers, 243
 - overflow, 78–82
 - representation, 26, 61, 244
 - type conversions, 244–246, 274–275
- signed keyword, 275
- SignExtend instruction, 259–260, 263, 282–283
- sign extension, 61, 244–245, 275
 - in assembly, 263, 444
 - in TACKY, 259–260, 282–283
- sign flag (SF), 78–80, 83
- significant degree, 638
- single-precision format, 299
- sizeof operator, 458, 462–466, 471, 477–478, 480–481
- Song, Dawn, 611
- source character set, 430
- source file, xxviii, 7–8, 208
- special characters, 429, 450
- special sequences (EBNF), 15
- spilling, 616, 627–630, 642–644, 646
 - candidates for, 642
 - spill code, 616, 620
 - spill cost, 630–631, 638, 642, 644–645
- SSA (static single assignment) form, 672
- SSE. *See* Streaming SIMD Extension instructions
- SSE class, 519
- stack, 19, 27–31
 - alignment, 185, 197–198, 648–649
 - executable, 19
 - additional resources, 22
 - frames, 29–31
 - allocating, 42, 197–199, 200–201
 - pointer, 27
- stack frames, 29–31
- Stack operand, 40, 42, 44
 - replaced with Memory operand, 375
- StaticConstant construct (assembly), 324, 326, 336, 446
 - emitting, 340
- StaticConstant construct (TACKY), 442, 446
- static initializers, 213–214. *See also* ZeroInit construct
 - in assembly, 221–222, 238–239
 - for characters, 436
 - compound, 404–405
 - in assembly, 418–419
 - for structures, 509–511
 - for double type, 308–309, 340
 - for long integers, 246, 257–258, 270
 - for pointers, 369–370, 428–429, 437, 438–439
 - null pointers as, 369
 - strings as, 437–439
 - in the symbol table, 257
 - type checking, 257–258
 - for unsigned integers, 280–281
- static single assignment (SSA) form, 672
- static specifier, 208, 209–211, 213, 216–217, 230–233
- static storage duration, 213–214. *See also* static variables
 - replacing pseudoregisters with, 237
- StaticVariable construct (assembly), 235–236, 263–264, 413
 - emitting, 238–239
- StaticVariable construct (TACKY), 234–235, 258–259, 406
- static variables, 213–214
 - assembly for, 221–222, 235–239, 246
 - initializing, 213–214. *See also* static initializers
 - in TACKY, 234–235
 - type checking, 229–230, 231–233
- status flags, 78–80
 - carry, 284–285, 317
 - overflow, 78–80, 83
 - parity, 342
 - sign, 78–80, 83
 - zero, 78–80, 83
- Sterbenz lemma, 319
- storage-class specifiers, 207–208, 223
 - effects, 209–217
 - parsing, 225–226
- storage duration, 207, 212–213
 - allocated, 213, 461
 - in assembly, 221–222
 - automatic, 212–213, 217
 - vs. scope, 213
 - static, 213–214, 237
 - in the symbol table, 229–230
 - thread, 213
- Store instruction, 370–374
 - and liveness analysis, 609
 - and reaching copies analysis, 599–600, 601–602
- Streaming SIMD Extension (SSE) instructions, 310–312
 - arithmetic, 315–316
 - comparisons, 317
 - type conversions, 317, 320
- strict aliasing rules, 352
- string literals, 425–426
 - as array initializers, 425, 426
 - TACKY for, 440–441
 - type checking, 437–438

- in assembly, 426–429
 - AST definition, 324
 - emitting, 449–450
 - designating constant strings, 425–426
 - TACKY for, 441–442
 - type checking, 436, 438–439
 - lexing, 429–431
 - lvalues, 425, 436
 - parsing, 433
- struct keyword, 494
- structure member (.) operator, 491, 495
 - parsing, 497–498
 - TACKY for, 513–517
 - token for, 494
 - type checking, 506–508
- structure pointer (->) operator, 491, 495
 - parsing, 497–498
 - TACKY for, 514–515, 517
 - token for, 494
 - type checking, 506–507
- structure tags, 486–488, 489–490
 - resolving, 498–500
- structure types
 - classifying, 519–522, 533–534
 - complete, 486–487, 503
 - copying, 531–532
 - declarations, 486–491
 - definitions, 486
 - in the type table, 501–502
 - validating, 501
 - in function calls, 519–528, 532–546
 - incomplete, 486–487, 490, 503, 505–506
 - initializers, 492
 - TACKY for, 517–518
 - type checking, 509–511
 - layout in memory, 492–494
 - not implemented, 490–491
 - operations on, 491–492. *See also*
 - structure member operator;
 - structure point operator
 - padding, 493, 510–511, 518–519
 - return values of, 525–528, 545–546
 - specifiers, 498
 - tags, 486–488, 489–490
 - resolving, 498–500
 - type checking, 500–511
- sub instruction, 29–30, 60, 62–63
 - emitting, 66, 270
 - fixing up, 64, 268
- subnormal numbers, 298
- SubObject construct, 515–517
- subscript ([]) operator, 389
 - AST definition, 393
 - generation, 408
 - parsing, 396–397
 - TACKY for, 408–410
 - type checking, 399, 401–402, 471–472
- subsd instruction, 315
 - fixing up, 337
- subtraction (-) operator, 47–48
 - assembly for, 60, 62–63
 - floating-point, 315, 327
 - parsing, 50–55
 - pointer subtraction, 388–390
 - TACKY for, 406–408
 - type checking, 400–401, 472
 - TACKY for, 58
 - type checking, 254–255
- subtraction assignment (-=) operator, 113
- switch statements, 159
- symbols (assembly), 5
 - global vs. local, 168–169
 - symbol tables, 5, 89
- symbol table internal to compiler, 174–175, 179–181. *See also* backend
 - symbol table
 - generating TACKY top-level definitions
 - from, 234–235, 442
 - identifier attributes in, 229–233, 257–258, 438
 - renamed to frontend symbol table, 266
 - temporary variables in, 260–261
 - tentative definitions in, 229–230, 235
- symbol tables in object files, 5, 89
- System V x64 ABI, xxxvi, 184. *See also*
 - System V x64 calling convention
 - arrays, alignment of, 415
 - char, signedness of, 424
 - Clang violation of, 444–445
 - floating-point format, 296, 297
 - int and long
 - alignment of, 246
 - size of, 244
 - size_t, definition of, 460
 - structures, size and alignment of, 493
 - System V x64 calling convention, 184–194
 - additional resources, 344, 553–554
 - classifying values, 519
 - floating-point values in, 312–315
 - narrow arguments in, 444–445
 - structures in, 519–528

T

- TAC (three-address code), 35–36
- TACKY, 36–38
 - Constant operands, 36
 - generation, 37–38
 - address (&) operator, 370–372, 374, 514, 516–517
 - assignment expressions, 110, 371–374, 516
 - binary expressions, 58

- TACKY (*continued*)
 - break and continue statements, 155–156
 - cast expressions, 259–260, 281–283, 309–310, 375, 440, 479
 - compound initializers, 406, 410–411, 517–518
 - compound statements, 140
 - conditional expressions, 127, 479–480
 - dereference (*) operator, 371–374
 - function calls, 182–183, 479
 - function definitions, 182–183
 - if statements, 126–127
 - loops, 155–158
 - pointer arithmetic, 406–408
 - return statements, 36–38, 479
 - short-circuiting operators, 76–77
 - sizeof operator, 480–481
 - static variables, 234–235
 - structure member access operators, 513–517
 - subscript ([]) operator, 408
 - unary expressions, 37–38
- instructions, 42–43
- lvalue conversion in, 371–374, 515–517
- temporary variables, 36–38, 260–261
- top-level constants, 442
- Var operands, 36
- Taylor, Ian Lance, 21, 22
- temporary lifetimes, 508
- temporary variables, 36–38
 - naming, 38
 - on the stack, 29
 - in the symbol table, 260–261
- tentative definitions, 215–216
 - converting to TACKY, 235
 - in the symbol table, 229–230, 235, 411
 - type checking, 231–232
 - undefined behavior, 219–220
- terminal symbols, 15
- ternary operators, 121. *See also* conditional expressions
- .text directive, 238
- text section, 5, 283
- thread storage duration, 213
- three-address code (TAC), 35–36. *See also* TACKY
- tokens, 3, 8–10
 - constants, 8–10
 - character, 429
 - floating-point, 302–304
 - integer, 8, 247, 275, 304
 - identifiers, 8–10
 - string literals, 429
- Torczon, Linda, 669–670
- transfer functions, 584–585
 - liveness analysis, 605–606
 - for assembly code, 634–636
 - with Part II types, 608–609
 - reaching copies analysis, 589–592
 - with Part II types, 601–602
- translation units, 167, 208
- Truncate instruction, 259–260, 263, 282
- two's complement, 26, 45, 78, 274
- type checking, 178–182, 251–258
 - arrays, 398–399, 402–405, 471, 472–473
 - compound initializers, 403–405, 509–511
 - declarations, 179–181, 230–233, 257–258, 402–403
- double, 308–309
- expressions, 253–256
 - arithmetic operators, 254–255, 369, 435, 476–477
 - assignment, 256, 368, 399
 - bitwise complement (~) operator, 308, 369, 435
 - cast, 254, 369, 402, 471, 505
 - conditional, 256, 368, 467, 470, 476, 508
 - logical operators, 254–255, 470
 - pointer arithmetic, 400–401, 472
 - relational operators, 254–255, 366–367, 401, 476–477
 - remainder (%) operator, 254–255, 308, 369
 - sizeof operator, 477–478
 - structure member access operators, 506–507
 - subscript ([]) operator, 399, 401–402, 471–472
- file scope variable declarations, 231–232
- function calls, 179, 181–182, 256
- incomplete types, 471–473, 505–506
- pointers, 364–370, 400–402, 467–469, 471–472
- return statements, 256–257, 469–470
- string literals, 436–439
- structure types, 500–511
- type errors, 174
- type conversions
 - in assembly, 244–245, 317–324
 - floating-point, 317–324, 328–329
 - sign extension, 244–245, 444
 - truncation, 245, 263, 444
 - zero extension, 286–288, 443–444
 - character, 443–445
 - double, 317–324
 - to and from character types, 445

- rounding behavior, 300–301
 - undefined, 308, 371
- implementation-defined, 245, 352
- implicit, 254–255, 279, 351, 467–469
 - as if by assignment, 368, 468–469, 504–505
- Cast expression representing, 255
 - usual arithmetic conversions, 254–255, 279–280, 308, 435
- integer, 244–245, 274–275
- pointer, 351–352, 460
- in TACKY, 259–260, 281–283, 309–310
 - Copy, 259–260
 - to and from double, 309–310
 - SignExtend, 259–260
 - Truncate, 259–260
 - ZeroExtend, 281–283
- typedef declarations, 108–109
- type errors, 174. *See also* type checking
- type names, 361–363, 462, 465–466
- types, 178–179. *See also* character types; integer types; void type
 - aggregate, 384
 - arithmetic, 347, 476–477
 - array, 384–392
 - derived, 354
 - on exp nodes, 252–253
 - floating-point, 295–299
 - function, 178–179, 247–248
 - incomplete, 461–462
 - non-scalar, 470–471
 - pointer, 347, 349–353
 - scalar, 384, 470–471
- type specifiers
 - char, 429
 - character, 433
 - double, 302, 306–307
 - int, 8
 - integer, 249–250, 277–278
 - long, 247
 - signed, 275
 - structures, 498
 - unsigned, 275
 - void, 8
- type table, 500–502, 503–504, 506–507, 509–511, 515, 517–518

U

- UIntToDouble instruction, 309–310
 - assembly for, 320–324
- Ullman, Jeffrey, 611
- unary expressions, 25–27, 31–38
 - AST definition, 33
 - parsing, 33–34
 - formal grammar, 33, 397, 465
 - TACKY for, 36–38
 - type checking, 254

- unary operators. *See* unary expressions *and* names of individual operators
- unconditional jump instructions. *See* jump instructions (assembly); jump instructions (TACKY)
- undeclared variables, 104, 107, 134
- undefined behavior, 80–82
 - additional resources, 91
 - conflicting linkage, 218–219
 - handling safely, 672
 - integer overflow, 80–82
 - missing return statement, 111–112
 - modifying objects, 425–426, 508
 - out-of-range type conversions, 308, 317
 - pointer arithmetic, 388, 390
 - pointer dereferences, 351–352
 - tentative definitions, 219–220
 - variable accesses, 96, 106–107
- UndefinedBehaviorSanitizer, 672
- union types, 552–553
- universal character names, 10
- unreachable code elimination, 561–562, 581–584
 - combining with other optimizations, 569
- unsequenced evaluations, 58–59, 82
- unsigned char type, 424
- unsigned integers, 273–289
 - in assembly, 283–289
 - assembly type, 287
 - unsigned comparisons, 283–285, 287–288
 - unsigned division, 286, 288
 - constants, 275–278
 - regular expression for, 304
 - static initializers for, 280–281
 - type conversions, 274–275, 279–280, 282–283
 - unsigned int type, 273–281
 - unsigned long type, 273–281
 - wraparound, 79, 285–286, 575
- unsigned keyword, 275
- usual arithmetic conversions, 254–255, 279–280, 308, 435

V

- values, 348
- variable declarations, 94–95, 208–220
 - of array type, 384–385
 - AST definition, 98, 171
 - linkage, 209–212
 - parsing, 100–101, 224–227
 - resolving identifiers in, 105–106, 138–139, 227–229
- scopes, 131–134, 208–209
 - block scope, 208
 - file scope, 208

- variable declarations (*continued*)
 - storage duration, 212–214
 - type checking, 179–180, 231–233, 257–258
- variable resolution, 104–108, 136–139, 227–229
 - conditional expressions, 125–126
 - if statements, 125–126
 - loops, 151–152
 - multiple scopes, 136–139
 - renamed identifier resolution, 174
- variables, 93–97, 208–222. *See also*
 - static variables
 - aliased, 599–602, 609, 637
 - automatic, 208
 - external, 208, 227–229
 - live, 603
 - local, 93–95
 - resolving, 104–107, 227–229
 - scopes, 131–134, 208–209
 - in TACKY, 36–38, 110
 - temporary, 36–38, 260–261
 - type checking, 181, 253
 - variable resolution, 107
 - variadic functions, 191
 - void expressions, 459
 - void keyword, 8–9
 - as parameter list, 162, 459, 466–467
 - void type, 458–460
 - casts to, 459, 471, 479
 - conditional expressions with, 459, 476, 479–480
 - in C standard, 458, 474–475
 - functions returning, 458, 469–470, 479, 482
 - pointers to, 460–461, 475
 - conversions to and from, 467–469
 - restrictions on, 473–476
 - when valid, 473–475
 - volatile objects, 560

W

- Wang, Daniel, 22
- while statements, 144, 148–150, 151–155, 157
- whitespace, 9–10
- wide character types, 424
- Windows Subsystem for Linux (WSL), xxxiv
- w suffix, 28

X

- x64 instruction set, xxvii. *See also* assembly code and names of individual instructions
 - AT&T vs. Intel syntax, 6, 244, 570
 - documentation, xxxvi
 - Streaming SIMD Extension
 - instructions, 310–312
- x64 processor, xxxiv
 - little-endian, 86
 - memory address size, 28
- x86-64. *See* x64 instruction set; x64 processor
- Xcode, xxxiv–xxxv
- XMM registers, 311–312, 325
 - allocating, 631
 - building register interference graph, 637
 - in function calls, 312–315, 329–333, 519, 532–541, 545–546
 - zeroing, 316
- XOR (^) operator, 67
- XOR assignment (^=) operator, 113
- xorpd instruction, 316, 324–325, 328
 - emitting, 341
 - fixing up, 337

Y

- Yang, Edward, 253
- Yang, Zhaomo, 611

Z

- Zephyr Abstract Syntax Description Language. *See* ASDL
- .zero directive, 222
- ZeroExtend instruction, 281–283, 288
- zero extension, 274, 281–282, 286–288, 443–444
- zero flag (ZF), 78–80, 83
 - comisd, set by, 317
 - in unsigned comparisons, 284–285
- ZeroInit construct, 405
 - emitting, 418–419
 - initializing padding, 510–511
 - initializing scalar variables, 405
 - initializing tentatively defined arrays, 411