Index

abstract data type, 5
abstract syntax, see syntax, abstract
access link, see static link
activation record, 6, 125–134
Ada, 350, 351, 371, 380
addressing mode, 199, 203
ADT, see abstract data type
Aiken-Nicolau algorithm, 478–482, 493
alias analysis, 390, 402–407, 426
in coalescing register allocation, 250
alignment, see cache alignment
alloca, 213
allocation of activation records, 125, 127, 171
of arrays and records, 167
of heap data, 291
register, see register allocation
alphabet, 18
ambiguous grammar, see grammar analysis
dataflow, see dataflow analysis
liveness, see liveness
antidependence, see dependence, write-after-read
approximation
dataflow analysis, 224, 227, 385
in garbage collection, 273
of spill effect, 236
of strictness, 345
argument, see parameter
array, 160, 162, 167
bounds check, 164, 425–429
Assemble module, 206
associativity, see right-associative, left-
associative, nonassociative
attribute grammar, 13
available expressions, 389
Baker's algorithm, 290
basic block, 185, 187, 394, 398, 416
beta reduction, see inline expansion
binding, 103–111, see also precedence
in FindEscape module, 140
in type environment, 112
in value environment, 115
Bison, 5, 69, 96
blacklist, 297
block structure, see function, nested
blocking, 511–514, 516
boxing, 372–377
branch prediction, 490–493
buffered input, 35
bypass datapaths, 476, 478
C programming language, 95
linking to, 168
writing compiler for, 18, 94, 125, 126,
131, 139, 153, 160–162, 166, 167,
213, 265, 338, 402, 404, 407, 411
writing compiler in, 5, 9–11, 145
C++, 265, 308, 350, 351, 371, 402
cache, 498–501
alignment, 502–504
and garbage collection, 283, 514–515
cache alignment, 515
CALL, 176, 177, 183
call
by name, 338
by need, 339
INDEX

by reference, 132, 133
callee-save, see register, callee-save
caller-save, see register, caller-save
Canon module, 177
canonical tree, see intermediate representation, canonical
card marking, 286
CISC, 195, 203–205, 493
class
descriptor, 303–306, 309–314
classless language, 310
cloning, 310
closure
conversion, 332–334, 336
e., 27, 28, 34
function, 318, 320, 347
Kleene, 19, 40
of LR state, 60, 64
coaesing, 239–256, 261, 336, 393
conservative, 239
of SSA variables, 463
code generation, see instruction selection
code-generator generator, 201
Codegen module, 212
coloring, see graph coloring
comma operator, see expression sequence
common-subexpression elimination, 389, 392
commute, 179–189
complex instruction set, see CISC
computer, see CISC and RISC
conditional jump, 154, 165, 176, 185
conditional move, 488
conflict
in predictive parser, 47
reduce-reduce, 69, 70, 72, 75, 97
resolution of, 72–75
shift-reduce, 62, 68–70, 72, 74, 97
conservative approximation, see approximation
constant folding, 453
constant propagation, 389, 452–453
conditional, 453–456
constraint
functional-unit, 475, 477
constructor, 9
continuation, 323, 348
continuation-passing style, 469
deep-first search
for dataflow analysis, 222, 224, 395, 396
garbage collection, 273, 283, 295
spanning tree, 444–445
derivation, 42
descriptor
class, 292, 303–306, 309–314
level, 170
record, 292, 294
DFA, see finite automaton
display, 148
class hierarchy, 307, 312, 313
dominance frontier, 438, 470
dependence
control, see control dependence
data, 457, 476, 510
loop-carried, 479
memory and array, 457–459, 479
read-after-write, see dependence, data
write-after-read, 457, 475, 510
write-after-write, 457, 475, 510
dangling else, 68
dangling reference, 131
data type, abstract, see abstract data type
dataflow, see also liveness, reaching definitions, available expressions, etc.
analysis, 6
bit vector, 394
equations, 220–225, 385, 387, 389, 401, 406, 413
iteration, see iteration algorithms
work-list algorithms, 396
dead
state, 23
dead code, 329, 342, 393, 397, 398, 401, 402, 423, 428, 451, 461
def (of variable), 220
def-use chain, 433, 472
deforestation, 342–344
dependence
control, see control dependence
data, 457, 476, 510
loop-carried, 479
memory and array, 457–459, 479
read-after-write, see dependence, data
write-after-read, 457, 475, 510
write-after-write, 457, 475, 510
dependence
control, see control dependence
data, 457, 476, 510
loop-carried, 479
memory and array, 457–459, 479
read-after-write, see dependence, data
write-after-read, 457, 475, 510
write-after-write, 457, 475, 510
dependence
control, see control dependence
data, 457, 476, 510
loop-carried, 479
memory and array, 457–459, 479
read-after-write, see dependence, data
write-after-read, 457, 475, 510
write-after-write, 457, 475, 510
derivation, 42
descriptor
class, 292, 303–306, 309–314
level, 170
record, 292, 294
DFA, see finite automaton
display, 148
class hierarchy, 307, 312, 313
dominance frontier, 438, 470
<table>
<thead>
<tr>
<th>INDEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>dominance property, see static single-assignment form</td>
</tr>
<tr>
<td>dominator, 413–416, 418, 426–428, 430, 470</td>
</tr>
<tr>
<td>efficient calculation of, 444–450, 468</td>
</tr>
<tr>
<td>dynamic programming for instruction selection, 197–201</td>
</tr>
<tr>
<td>for register allocation, 257–260</td>
</tr>
<tr>
<td>dynamic scheduling, see out-of-order execution</td>
</tr>
<tr>
<td>edge splitting, 442</td>
</tr>
<tr>
<td>edge-split SSA, see static single-assignment form</td>
</tr>
<tr>
<td>else, dangling, 68</td>
</tr>
<tr>
<td>emission in instruction selection phase, 198, 201</td>
</tr>
<tr>
<td>of assembly code, 5, 6, 214, 260</td>
</tr>
<tr>
<td>encapsulation, 299</td>
</tr>
<tr>
<td>end-of-file marker, 46</td>
</tr>
<tr>
<td>Env module, 115</td>
</tr>
<tr>
<td>environment, 12, 103–112, 114, 115, 123, 140, 302, 318, 333</td>
</tr>
<tr>
<td>functional, 107</td>
</tr>
<tr>
<td>imperative, 105</td>
</tr>
<tr>
<td>multiple, 105</td>
</tr>
<tr>
<td>equational reasoning, 315–319, 324, 337, 347</td>
</tr>
<tr>
<td>error message, 95, 99</td>
</tr>
<tr>
<td>error recovery, 54</td>
</tr>
<tr>
<td>escape, 101, 133, 140–141, 265, 319, 337, 347, see also FindEscape</td>
</tr>
<tr>
<td>Escape module, 140</td>
</tr>
<tr>
<td>ESEQ, 176–184</td>
</tr>
<tr>
<td>expression sequence, 99, see also ESEQ</td>
</tr>
<tr>
<td>FindEscape module, 140</td>
</tr>
<tr>
<td>finite automaton, 18, 21–30</td>
</tr>
<tr>
<td>deterministic, 22</td>
</tr>
<tr>
<td>minimization, 37</td>
</tr>
<tr>
<td>nondeterministic, 24</td>
</tr>
<tr>
<td>converting to DFA, 27–32</td>
</tr>
<tr>
<td>FIRST set, 48–53, 64</td>
</tr>
<tr>
<td>fixed point, 49, 221, 390, 407</td>
</tr>
<tr>
<td>least, 224, 233, 401, 453</td>
</tr>
<tr>
<td>Flex, 5, 35</td>
</tr>
<tr>
<td>flow graph, 218</td>
</tr>
<tr>
<td>reducible, 411</td>
</tr>
<tr>
<td>flow, data, see dataflow</td>
</tr>
<tr>
<td>FlowGraph module, 231</td>
</tr>
<tr>
<td>FOLLOW set, 49–51, 53, 55, 63</td>
</tr>
<tr>
<td>forward reference, see recursion, mutual</td>
</tr>
<tr>
<td>forwarding, 281–284</td>
</tr>
<tr>
<td>fragmentation, 277</td>
</tr>
<tr>
<td>frame, see activation record</td>
</tr>
<tr>
<td>Frame module, 136, 268</td>
</tr>
<tr>
<td>frame pointer, 127–129, 148, 158, 159, 170, 213, 215</td>
</tr>
<tr>
<td>on Pentium, 203</td>
</tr>
<tr>
<td>freeze, 240, 249, 255</td>
</tr>
<tr>
<td>function</td>
</tr>
<tr>
<td>dead, 329</td>
</tr>
<tr>
<td>declaration, 122</td>
</tr>
<tr>
<td>header, 122</td>
</tr>
<tr>
<td>higher-order, 126, 315</td>
</tr>
<tr>
<td>integration, see inline expansion</td>
</tr>
<tr>
<td>leaf, 131</td>
</tr>
<tr>
<td>nested, 101, 126–127, 133–135, 143, 144, 148, 159, 265, 315, 318–319,</td>
</tr>
<tr>
<td>402, 521</td>
</tr>
<tr>
<td>functional intermediate form, 369, 464–469</td>
</tr>
<tr>
<td>functional programming, 13, 104, 315–349, see also side effect</td>
</tr>
<tr>
<td>impure, 316–317</td>
</tr>
<tr>
<td>pure, 319–325</td>
</tr>
<tr>
<td>symbol tables, 107–108</td>
</tr>
<tr>
<td>functional unit, 475, 476</td>
</tr>
<tr>
<td>multiple, 476</td>
</tr>
<tr>
<td>garbage collection, 12, 167, 273–298, 318, 337, 349</td>
</tr>
<tr>
<td>and cache, 283, 514–515</td>
</tr>
<tr>
<td>Baker’s algorithm, 290</td>
</tr>
<tr>
<td>compiler interface, 291–294, 370, 372, 378</td>
</tr>
<tr>
<td>concurrent, 288</td>
</tr>
<tr>
<td>conservative, 296</td>
</tr>
<tr>
<td>copying, 280–285</td>
</tr>
<tr>
<td>cost, 275, 279, 284, 287, 297, 290</td>
</tr>
<tr>
<td>flip, 290</td>
</tr>
<tr>
<td>generational, 285–287, 514</td>
</tr>
<tr>
<td>incremental, 287–291</td>
</tr>
<tr>
<td>mark-sweep, 273–277</td>
</tr>
<tr>
<td>reference counts, 278–280</td>
</tr>
<tr>
<td>generic, 350, 380</td>
</tr>
<tr>
<td>grammar, 5, 41–46, see also syntax</td>
</tr>
<tr>
<td>ambiguous, 43, 51, 52, 68–69, 94, 199</td>
</tr>
<tr>
<td>attribute, 13</td>
</tr>
</tbody>
</table>

539
INDEX

- factoring, 54
- for intermediate representation, 7, 9
- hierarchy of classes, 67
- LALR, 65, 67, 69
- LL(1), 52
- of straight-line programs, 7
- to specify instruction set, 199–201
- transformations, 52, 90, 94
- unambiguous, 51
- interference graph, 227–248, 260
- construction of, 228, 231–233, 252
- for SSA form, 463
- from SSA form, 463, 472
- intermediate representation, 6, 151–154,
  see also Tree
- canonical, 176–184
- functional, 369, 464–469
- interpreter, 92, 93
- invariant, see loop invariant
- IR, see intermediate representation
- item
  - LR(0), 59
  - LR(1), 64
- iteration algorithms
  - alias analysis, 406
  - dominators, 413
  - ε-closure, 28
  - efficient, 393–397
  - first and follow sets, 49
  - invention of, 407
  - liveness analysis, 221–222
  - LR parser construction, 61
  - minimization of finite automata, 37
  - reaching definitions, 388
- iterative modulo scheduling, see modulo scheduling
- Java, 351
  - writing compiler for, 18, 94, 105, 161,
    265, 292, 298, 307–309, 314, 338,
    403, 404, 426, 427
  - Jouette, 191–195, 208–210
  - Schizo, 199
- Kleene closure, 19, 40
- label, 141
- lambda calculus, 347, 464
  - second-order, 368, 369, 377, 379
- landing pad, 469
- lattice, 454
- lazy evaluation, 337–342, 469
- leaf function, 131
- left factoring, 53
- left recursion, 52
- left-associative operator, 73, 74, 524
- Lengauer-Tarjan algorithm, 444–450, 468,
  see also dominator
- level, static nesting, 143, 144

IBM 360/91, 489
- ImplicitPoly-Tiger, 362
- induction variable, 419–425
  - coordinated, 423, 424, 426
  - linear, 421
- inheritance, 299, 302
  - multiple, 304
  - single, 302, 311, 312
- inline expansion, 292, 326–332, 348, 466
  - of polymorphic functions, 371
- instantiation of variable, 125
- instruction
  - fetch, 490, 504
  - representation of, 206
  - pipeline, see pipeline
  - resource usage of, 476
  - selection of, 6, 191–217
  - side effect of, 203, 213
  - three-address, 203
  - two-address, 203, 209
  - variable-length, 203
- instruction set, see CISC and RISC
- instruction-level parallelism, 474
- Intel, see Pentium
- interfaces, 5
- interference, 54
- for intermediate representation, 7, 9
- hierarchy of classes, 67
- LALR, 65, 67, 69
- LL(1), 52
- of straight-line programs, 7
- to specify instruction set, 199–201
- transformations, 52, 90, 94
- unambiguous, 51
- interference graph, 227–248, 260
- construction of, 228, 231–233, 252
- for SSA form, 463
- from SSA form, 463, 472
- intermediate representation, 6, 151–154,
  see also Tree
- canonical, 176–184
- functional, 369, 464–469
- interpreter, 92, 93
- invariant, see loop invariant
- IR, see intermediate representation
- item
  - LR(0), 59
  - LR(1), 64
- iteration algorithms
  - alias analysis, 406
  - dominators, 413
  - ε-closure, 28
  - efficient, 393–397
  - first and follow sets, 49
  - invention of, 407
  - liveness analysis, 221–222
  - LR parser construction, 61
  - minimization of finite automata, 37
  - reaching definitions, 388
- iterative modulo scheduling, see modulo scheduling
- Java, 351
  - writing compiler for, 18, 94, 105, 161,
    265, 292, 298, 307–309, 314, 338,
    403, 404, 426, 427
  - Jouette, 191–195, 208–210
  - Schizo, 199
- Kleene closure, 19, 40
- label, 141
- lambda calculus, 347, 464
  - second-order, 368, 369, 377, 379
- landing pad, 469
- lattice, 454
- lazy evaluation, 337–342, 469
- leaf function, 131
- left factoring, 53
- left recursion, 52
- left-associative operator, 73, 74, 524
- Lengauer-Tarjan algorithm, 444–450, 468,
  see also dominator
- level, static nesting, 143, 144
INDEX

Lex, 5, 30, 34
lexical analyzer, 6, 16–38, 95
generator, 30
lexical scope, see function, nested
Lisp, 379
live range, 218, 228
live-in, 220
live-out, 220
liveness, 6, 218–234, 252, 391, 393, 396, 398, 400, 401
in SSA form, 463
of heap data, 273
Liveness module, 231
LL(1), see parser, LL(1)
local variable, 125
locality of reference, see cache
lookahead, 38
loop, 410
header, 410, 415–416
inner, 415
interchange, 510–511
invariant, 330, 341, 416, 418–424, 432
natural, 415–416
nested, 416
postbody, see postbody node
scheduling, 478–490
unrolling, 429–430, 478, 508
LR(1), see parser, LR(1)
l-value, 161–163

macro preprocessor, 17
Maximal Munch, 195
memory
allocation, see allocation and garbage
collection
method, 299
instance, 302
lookup, 303, 311–313
multi-, 313
private, 310
replication, 314
static, 303
MIPS, 476, 493
MIPS computer, 138, 145
ML, 94, 350, 379
writing compiler for, 105, 126, 133, 149, 161, 316, 338, 348, 402, 403, 407, 426, 427
Modula-3, 94, 292, 350

modularity, 11, 299
modulo scheduling, 482–490
Motorola 68000, 199
MOVE, 182
multimethod, 313
negation, 100, 164
NFA, see finite automaton
nonassociative operator, 73, 524
nonterminal symbol, 41
nullable symbol, 48–51, 53
object-oriented
classless language, 310
language, 161, 292, 299–314, 380
occs parser generator, 69
occurs check, 364
out-of-order execution, 489, 505
output dependence, see dependence, write-after-write
overloaded operator, 116
overloading, 350, 378–380
parallel processing
instruction-level, 474
parameter, see also view shift
actual, 212, 328, 335
address of, 131
allocating location of, 133
by-reference, 132, 141, 402, 404
declaration, 120, 122
formal, 115, 138, 176
in frame, 131, 136, 137
in Tiger, 520, 521, 523
lazy, 338
nesting level of, 140
of method, 300
outgoing, 129, 212, 269
register, 130, 138, 145
self, 300, 301
static link, 140, 319, 333
substitution, 328
type-checking, 118
variable number of, 131
parse tree, 43, 92
abstract, 96
parser, 6, 95
dynamic programming, 199
error recovery, 54
generator, 69

© Cambridge University Press  www.cambridge.org
INDEX

LL(1), 47–55
LL(k), 52, 56
LR, 92
LR(0), 59–63
LR(1), 56–59
LR(k), 56–59
predictive, 47–55
construction, 51
recursive-descent, 46–48, 51–55, 88
SLR, 62–64
Pascal, 94, 125, 126, 133, 149, 160–162, 167, 292, 338, 402, 403
pattern, see tile
Pentium, 138, 203, 204, 213
persistent data structure, 15, 108
phases
of a compiler, 4, 351
order of, 5
φ-function, 434
pipeline, 476
software, see scheduling
pointer
derived, 293
reversal, 276
Poly-Tiger, 352
polymorphism, 350
поз, 95
position
in source code, 95
postbody node, 431, 432, 437, 469
postdominance, 460
precedence, 45
precedence directive, 72–75
predicated execution, 488
prediction, see branch prediction
predictive parser, see parser
prefetching, 504–509, 514, 515
preheader, 329–330, 416, 422, 426, 431, 432, 469, 492
preprocessor, 17
procedure, see function
profile-based optimization, 492, 494
pure functional language, see functional programming, pure
quadruple, 386, 464
RAW, see dependence, data
reachable data, 273
reaching definitions, 311, 387, 403
reaching expressions, 391
recursion
mutual, 94, 99
tail, see tail recursion
recursive descent, see parser
red-black tree, 108
reduce-reduce conflict, see conflict
reduced instruction set, see RISC
reducible flow graph, 411
reference counts, 278–280
reference parameter, see call by reference
register
allocation, 6, 204, 393
for trees, 257–260
Sethi-Ullman algorithm, 258
callee-save, 129–130, 171, 172, 215, 243–244, 267, 293, 335, 347
caller-save, 129–130, 147, 212, 244, 336
classes of, 199, 203, 204
windows, 131, 137, 138
zero, 215
register allocation
for SSA form, 462
regular expression, 18–21, 34
converting to NFA, 25–26
remembered set, 286, 295
representation analysis, 373–377, 379
reservation table, 488, 506
reserved word, 17
return
address, 215
right-associative operator, 73, 74, 316
RISC, 145, 195
roots of garbage-collected heap, 273, 280, 282, 286, 297
rule priority, 20, 30
runtime system, 167
scalar replacement, 512, 516
scanner, see lexical analyzer
scheduling, 478–490, 505
modulo, see modulo scheduling
Scheme, 126, 149, 316, 348
Schizo-Jouette, 199
scope, 103–108, 132, 142, 464, 520–525
lexical, see function, nested
nested, 160
search tree, 15

542
INDEX

balanced, 15
semantic action, 31, 88–97, 101–102
analysis, 6, 103
stack, 91, 92, 95
value, 18, 42, 90, 91
semantics, 13, 92
semidominator, 446
sentinel, 35
Sethi-Ullman algorithm, 258
shift of view, see view shift
shift-reduce conflict, see conflict
side effect, 13, 92, 315
in semantic action, 92
of instruction, 203, 213
significant degree, 236, 239–249, 264
Simula, 312
Sparc, 137–139, 145
sparse conditional constant, see constant propagation
spill, 204, 235–237, 239, 242, 260, 267
cost, 261
potential, 236, 240, 242, 263, 264
work-list, 249–251
splitting, see edge splitting
SSA, see static single-assignment form
stack frame, see activation record
illustration of, 128
of activation records, 127–129
pointer, 127–129, 171, 213, 215
on Pentium, 203
semantic, 91, 92, 95
start state, 32
state
LR(0), 61, 62
LR(1), 64
static single-assignment form, 433–436
converting from, 462–463
converting to, 436–442
data structures for, 451
dominance property, 438, 456, 463
dominance splitting, 443, 456
optimizations using, 451–457
size of, 471, 472
unique successor or predecessor property, 442, 456
straight-line program, 7–9, 94
interpreter, 12
strength reduction, 419, 422–424
strictness analysis, 344–347
string literal, 166
substitution, 337, see type substitution
superscalar, 475
Symbol, 100
symbol in Tiger compiler, 108
Symbol module, 109, 110
symbol table, see environment
symbol, grammar, 41
syntax, 39, see also grammar
abstract, 5, 6, 92–95
of straight-line programs, 8–9
of Tiger, 97–101
tree, 95
concrete, 94, 95
vs. semantics, 75
table compression, 34
tableau, 480
tagging values, 373
tail recursion, 335–336
Temp module, 141
template, 350
temporary variable, 141
terminal symbol, 41
thunk, 338–341, 344, 348
Tiger
abstract syntax, 97–101
instruction selection, 205–214
language, 518–525
tile, 191–206, 212, 257–259
cost, 197
optimum vs. optimal, 194
tools, compiler generation, 5, 201
trace, 186–188	race scheduling, 489, 493
Translate module, 142, 154
tree
canonical, see intermediate representation, canonical
data structures, 7
intermediate representation, 6
parse, see parse tree
pattern, see tile
pattern matching, 201

543

© Cambridge University Press  www.cambridge.org
## INDEX

<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>red-black</td>
<td>108</td>
</tr>
<tr>
<td>register allocation for</td>
<td>257–260</td>
</tr>
<tr>
<td>search, see search tree</td>
<td></td>
</tr>
<tr>
<td>Tree module, 152</td>
<td></td>
</tr>
<tr>
<td>type checking</td>
<td>115–122, 354–359</td>
</tr>
<tr>
<td>constructor</td>
<td>352</td>
</tr>
<tr>
<td>equivalence</td>
<td>114, 355, 357</td>
</tr>
<tr>
<td>generalization</td>
<td>364</td>
</tr>
<tr>
<td>hierarchy analysis</td>
<td>311</td>
</tr>
<tr>
<td>Hindley-Milner</td>
<td>362, 368, 379</td>
</tr>
<tr>
<td>inference</td>
<td>359–369</td>
</tr>
<tr>
<td>instantiation</td>
<td>365</td>
</tr>
<tr>
<td>passing</td>
<td>377, 379</td>
</tr>
<tr>
<td>polymorphic</td>
<td>352</td>
</tr>
<tr>
<td>propagation</td>
<td>311</td>
</tr>
<tr>
<td>recursive</td>
<td>114, 359</td>
</tr>
<tr>
<td>substitution</td>
<td>355, 373</td>
</tr>
<tr>
<td>unification, see unification</td>
<td>357, 362–364</td>
</tr>
<tr>
<td>union-find</td>
<td>468</td>
</tr>
<tr>
<td>unique successor or predecessor property</td>
<td>462</td>
</tr>
<tr>
<td>unreachable code</td>
<td>428, 453</td>
</tr>
<tr>
<td>unroll and jam</td>
<td>513, 516</td>
</tr>
<tr>
<td>unrolling, see loop unrolling</td>
<td></td>
</tr>
<tr>
<td>use (of variable)</td>
<td>220</td>
</tr>
<tr>
<td>use-def chain</td>
<td>472</td>
</tr>
<tr>
<td>factored</td>
<td>469</td>
</tr>
<tr>
<td>useless variable</td>
<td>424</td>
</tr>
<tr>
<td>value numbering</td>
<td>398, 407, 434, 468</td>
</tr>
<tr>
<td>varargs</td>
<td>131</td>
</tr>
<tr>
<td>variable</td>
<td></td>
</tr>
<tr>
<td>escaping, see escape free</td>
<td>332</td>
</tr>
<tr>
<td>in frame</td>
<td>132</td>
</tr>
<tr>
<td>live, see liveness</td>
<td></td>
</tr>
<tr>
<td>local</td>
<td>125</td>
</tr>
<tr>
<td>useless</td>
<td>424</td>
</tr>
<tr>
<td>view shift</td>
<td>137, 139, 266</td>
</tr>
<tr>
<td>VLIW</td>
<td>493</td>
</tr>
<tr>
<td>WAR, see dependence</td>
<td></td>
</tr>
<tr>
<td>WAW, see dependence</td>
<td></td>
</tr>
<tr>
<td>work-list algorithms</td>
<td>249, 396, 441, 452, 455</td>
</tr>
<tr>
<td>Yacc, 5, 69–90, 92, 96</td>
<td></td>
</tr>
</tbody>
</table>

544