

---

## References

- AA94. American National Standards Institute and Accredited Standards Committee X3 for Information Technology. *ANSI X3.221-1994: AT Attachment Interface for Disk Drives*. American National Standards Institute, 1430 Broadway, New York, NY 10018, USA, 1994.
- AH08. Eyad Alkassar and Mark A. Hillebrand. Formal Functional Verification of Device Drivers. In Natarajan Shankar and Jim Woodcock, editors, *2nd IFIP Working Conference on Verified Software: Theories, Tools, and Experiments (VSTTE'08)*, Toronto, Canada, volume 5295 of *LNCS*, pages 225–239. Springer, 2008.
- AHL<sup>+</sup>09. Eyad Alkassar, Mark A. Hillebrand, Dirk Leinenbach, Norbert Schirmer, Artem Starostin, and Alexandra Tsyban. Balancing the Load. *J. Autom. Reasoning*, 42(2-4):389–454, 2009.
- ASS08. Eyad Alkassar, Norbert Schirmer, and Artem Starostin. Formal Pervasive Verification of a Paging Mechanism. In C. R. Ramakrishnan and Jakob Rehof, editors, *Proc. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08)*, Budapest, Hungary, volume 4963 of *LNCS*, pages 109–123. Springer, 2008.
- Bau14. Christoph Baumann. *Ownership-Based Order Reduction and Simulation in Shared-Memory Concurrent Computer Systems*. PhD thesis, Saarland University, Saarbrücken, 2014. <http://www-wjp.cs.uni-saarland.de/publikationen/Ba14.pdf>.
- Bec87. Bernd Becker. An Easily Testable Optimal-time VLSI-multiplier. *Acta Inf.*, 24(4):363–380, August 1987.
- BJ01. Christoph Berg and Christian Jacobi. Formal Verification of the VAMP Floating Point Unit. In Tiziana Margaria and Thomas F. Melham, editors, *Proc. 11th Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'01)*, Livingston, Scotland, UK, volume 2144 of *LNCS*, pages 325–339. Springer, 2001.
- BJK<sup>+</sup>03. Sven Beyer, Christian Jacobi, Daniel Kroening, Dirk Leinenbach, and Wolfgang J. Paul. Instantiating Uninterpreted Functional Units and Memory System: Functional Verification of the VAMP. In Daniel Geist and Enrico Tronci, editors, *Correct Hardware Design and Verification Methods, Proc. 12th IFIP WG 10.5 Advanced Research Working Conference (CHARME'03)*, L'Aquila, Italy, volume 2860 of *LNCS*, pages 51–65. Springer, 2003.

- BJMY89. William R. Bevier, Warren A. Hunt Jr., J Strother Moore, and William D. Young. An Approach to Systems Verification. *J. Autom. Reasoning*, 5(4):411–428, 1989.
- BP05. Dominique Borriero and Wolfgang J. Paul, editors. *Proc. CHARME'05*, Saarbrücken, Germany, volume 3725 of *LNCS*. Springer, 2005.
- CDH<sup>+</sup>09. Ernie Cohen, Markus Dahlweid, Mark Hillebrand, Dirk Leinenbach, Michal Moskal, Thomas Santen, Wolfram Schulte, and Stephan Tobies. VCC: A Practical System for Verifying Concurrent C. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Markus Wenzel, editors, *Proc. TPHOLs'09*, Munich, Germany, volume 5674 of *LNCS*, pages 23–42. Springer, 2009.
- CPS13. Ernie Cohen, Wolfgang Paul, and Sabine Schmaltz. Theory of Multi Core Hypervisor Verification. In Peter van Emde Boas, Frans C. A. Groen, Giuseppe F. Italiano, Jerzy Nawrocki, and Harald Sack, editors, *Proc. SOFSEM'13: Theory and Practice of Computer Science*, Spindleruv Mlyn, Czech Republic, volume 7741 of *LNCS*, pages 1–27. Springer, 2013.
- CS10. Ernie Cohen and Bert Schirmer. From Total Store Order to Sequential Consistency: A Practical Reduction Theorem. In Matt Kaufmann and Lawrence C. Paulson, editors, *Proc. Interactive Theorem Proving, First International Conference (ITP'10)*, Edinburgh, UK, volume 6172 of *LNCS*, pages 403–418. Springer, 2010.
- DHP05. Iakov Dalinger, Mark A. Hillebrand, and Wolfgang J. Paul. On the Verification of Memory Management Mechanisms. In Borriero and Paul [BP05], pages 301–316.
- Dör10. Jan Dörrenbächer. *Formal Specification and Verification of a Microkernel*. PhD thesis, Saarland University, Saarbrücken, 2010. <http://www-wjp.cs.uni-saarland.de/publikationen/JD10.pdf>.
- DSS09. Matthias Daum, Norbert W. Schirmer, and Mareike Schmidt. Implementation Correctness of a Real-Time Operating System. In Dang Van Hung and Padmanabhan Krishnan, editors, *Proc. SEFM'09*, Hanoi, Vietnam, pages 23–32. IEEE Computer Society, 2009.
- HKS08. Ralf Huuck, Gerwin Klein, and Bastian Schlich, editors. *Proc. 3rd Intl Workshop on System Software Verification (SSV'08)*, Sydney, Australia, volume 217 of *ENTCS*. Elsevier, 2008.
- HT09. Mark A. Hillebrand and Sergey Tverdyshev. Formal Verification of Gate-Level Computer Systems. In Anna E. Frid, Andrey Morozov, Andrey Rybalchenko, and Klaus W. Wagner, editors, *Computer Science - Theory and Applications, Proc. 4th International Computer Science Symposium in Russia (CSR'09)*, Novosibirsk, Russia, volume 5675 of *LNCS*, pages 322–333. Springer, 2009.
- HW73. C. A. R. Hoare and Niklaus Wirth. An Axiomatic Definition of the Programming Language PASCAL. *Acta Inf.*, 2:335–355, 1973.
- IdRP08. T. In der Rieden and W. J. Paul. Beweisen als Ingenieurwissenschaft: Verbundprojekt Verisoft (2003–2007). In B. Reuse and R. Vollmar, editors, *Informatikforschung in Deutschland*, pages 321–326. Springer, 2008.
- IdRT08. T. In der Rieden and A. Tsyban. CVM - A Verified Framework for Microkernel Programmers. In Huuck et al. [HKS08], pages 151–168.
- ISO11. ISO. *ISO/IEC 9899:2011 Information technology — Programming languages — C*. International Organization for Standardization, Geneva, Switzerland, December 2011.
- JR05. Warren A. Hunt Jr. and Erik Reeber. Formalization of the DE2 Language. In Borriero and Paul [BP05], pages 20–34.

- KAE<sup>+</sup>10. Gerwin Klein, June Andronick, Kevin Elphinstone, Gernot Heiser, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, et al. seL4: formal verification of an operating-system kernel. *Communications of the ACM*, 53(6):107–115, 2010.
- KMP14. M. Kovalev, S.M. Müller, and W.J. Paul. *A Pipelined Multi-core MIPS Machine: Hardware Implementation and Correctness Proof*, volume 9000 of LNCS. Springer, 2014.
- KO63. Anatoly A. Karatsuba and Y. Ofman. Multiplication of Multidigit Numbers on Automata. *Soviet Physics Doklady*, 7:595–596, 1963.
- KP95. Jörg Keller and Wolfgang J. Paul. *Hardware Design*. Teubner-Texte zur Informatik. Teubner, 1995.
- Kr01. Daniel Kröning. *Formal Verification of Pipelined Microprocessors*. PhD thesis, Saarland University, 2001. <http://www-wjp.cs.uni-saarland.de/publikationen/Kr01.pdf>.
- Lam79. Leslie Lamport. How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs. *IEEE Trans. Computers*, 28(9):690–691, 1979.
- Lan30. Edmund Landau. *Grundlagen der Analysis*. Akademische Verlagsgesellschaft, 1930.
- Ler09. Xavier Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009.
- LMW89. Jacques Loeckx, Kurt Mehlhorn, and Reinhard Wilhelm. *Foundations of Programming Languages*. John Wiley, 1989.
- LP08. D. Leinenbach and E. Petrova. Pervasive Compiler Verification – From Verified Programs to Verified Systems. In Huuck et al. [HKS08], pages 23–40.
- LPP05. D. Leinenbach, W. Paul, and E. Petrova. Towards the Formal Verification of a C0 Compiler: Code Generation and Implementation Correctness. In Bernhard K. Aichernig and Bernhard Beckert, editors, *Proc. SEFM'05*, Koblenz, Germany, pages 2–12. IEEE Computer Society, 2005.
- MIP01. MIPS Technologies, Inc. *MIPS32 Architecture For Programmers – Volume 2*, March 2001.
- MP98. Silvia M. Müller and Wolfgang J. Paul. On the correctness of hardware scheduling mechanisms for out-of-order execution. *Journal of Circuits, Systems, and Computers*, 8(02):301–314, 1998.
- MP00. Silvia M. Müller and Wolfgang J. Paul. *Computer Architecture, Complexity and Correctness*. Springer, 2000.
- NK14. Tobias Nipkow and Gerwin Klein. *Concrete Semantics - With Isabelle/HOL*. Springer, 2014.
- Pau78. Wolfgang J. Paul. *Komplexitätstheorie*. Leitfäden der angewandten Mathematik und Mechanik; Vol. 39. Teubner, 1978.
- Pau13. Wolfgang J. Paul. Computer Architecture 2 (unpublished lecture notes). Saarland University, 2013.
- PH90. David A. Patterson and John L. Hennessy. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 1990.
- PSS12. Wolfgang Paul, Sabine Schmaltz, and Andrey Shadrin. Completing the Automated Verification of a Small Hypervisor - Assembler Code Verification. In George Eleftherakis, Mike Hinchey, and Mike Holcombe, editors, *Proc. SEFM'12*, Thessaloniki, Greece, volume 7504 of LNCS, pages 188–202. Springer, 2012.
- Rog67. H. Rogers. *Theory of recursive functions and effective computability*. McGraw-Hill series in higher mathematics. McGraw-Hill, 1967.

- SH02.     Jun Sawada and Warren A Hunt. Verification of FM9801: An out-of-order micro-processor model with speculative execution, exceptions, and program-modifying capability. *Formal Methods in System Design*, 20(2):187–222, 2002.
- SMK13.     Thomas Sewell, Magnus Myreen, and Gerwin Klein. Translation Validation for a Verified OS Kernel. In Hans-Juergen Boehm and Cormac Flanagan, editors, *Proc. ACM-SIGPLAN Conference on Programming Language Design and Implementation (PLDI'13)*, Seattle, Washington, USA, pages 471–481. ACM, 2013.
- SS12.     Sabine Schmaltz and Andrey Shadrin. Integrated Semantics of Intermediate-Language C and Macro-Assembler for Pervasive Formal Verification of Operating Systems and Hypervisors from VerisoftXT. In Rajeev Joshi, Peter Müller, and Andreas Podelski, editors, *4th International Conference on Verified Software: Theories, Tools, and Experiments (VSTTE'12)*, Philadelphia, USA, volume 7152 of *LNCSE*, pages 18–33. Springer, 2012.
- SU70.     Ravi Sethi and J. D. Ullman. The Generation of Optimal Code for Arithmetic Expressions. *J. ACM*, 17(4):715–728, October 1970.
- SVZN<sup>+</sup>13.     Jaroslav Sevcik, Viktor Vafeiadis, Francesco Zappa Nardelli, Suresh Jagannathan, and Peter Sewell. CompCertTSO: A Verified Compiler for Relaxed-Memory Concurrency. *Journal of the ACM*, 60(3):22, 2013.
- VBC<sup>+</sup>15.     Viktor Vafeiadis, Thibaut Balabonski, Soham Chakraborty, Robin Morisset, and Francesco Zappa Nardelli. Common Compiler Optimisations are Invalid in the C11 Memory Model and what we can do about it. In Sriram K. Rajamani and David Walker, editors, *Proc. 42nd Symposium on Principles of Programming Languages (POPL'15)*, Mumbai, India, pages 209–220. ACM, 2015.
- VN13.     Viktor Vafeiadis and Chinmay Narayan. Relaxed separation logic: a program logic for C11 concurrency. In Antony L. Hosking, Patrick Th. Eugster, and Cristina V. Lopes, editors, *Proc. ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications, (OOPSLA'13)*, Indianapolis, IN, USA, pages 867–884. ACM, 2013.
- Wal64.     C.S. Wallace. A suggestion for a fast multiplier. *Electronic Computers, IEEE Transactions on*, EC-13(1):14–17, February 1964.

---

# Index

- A operand, *see* sequential processor
- ABI, *see* application binary interface
- abstract kernel, 434
- adder, 81
  - carry-chain, 82
  - carry-look-ahead, 88
  - CSA, 83
- addition, 9
  - algorithm, *see* school method
- address range
  - of translated code, 255
  - of variables, 259
- address translation, 384
- alignment, 113, 120, 123
- ALU, *see* arithmetic logic unit
- application binary interface, 498
- arithmetic logic unit, 94
- arithmetic sum, 25
- arithmetic unit, 89
  - negative bit, 90
  - negative signal, 92
  - overflow bit, 90
  - overflow signal, 94
- ASCII symbols, 182
- assembler syntax, 145
- asymptotic growth, 17
- AU, *see* arithmetic unit
  
- B operand, *see* sequential processor
- base address, *see* C0 variable
- base pointer, 255
- BCE, *see* branch condition evaluation
- big O notation, *see* asymptotic growth
  
- binary numbers, 33
  - addition, 36
  - subtraction, 39
- binary representation, 35
- bit operations, 20
- bit strings, 20
  - as binary numbers, 33
  - as two's complement numbers, 37
  - decomposition, 35
- bit vectors, *see* bit strings
- Boolean equations, 41
  - identities, 41, 42
  - solving equations, 41, 44
- Boolean expression, 39
  - as a circuit, 54
  - evaluation, 40
  - syntax, 169
- Boolean values, 17
- boot loader, 472
- border word, *see* derivation tree
- branch condition evaluation, 98
  - in sequential processor, 133
- buffer, *see* hard disk ports
- bytes, 20
  - byte-addressable memory, 112
  - consecutive bytes in memory, 113
  
- C abstraction, 333
- C address, *see* sequential processor
- C0
  - configuration, 199
  - initial, 209
  - well-formed, 334

- constant, 180
- expression
  - evaluation, 212
  - in function body, 207
  - node, 204
  - subexpression, 211
- function
  - current function, 200
  - table, 194
- global memory, 201
- heap, 184
- invariants, 204, 208, 209, 212
- memory content, 202
- program, 184
- program rest, 207
- recursion depth, 200
- result destination stack, 209
- stack, 200
- statement, 183
  - execution, 222
  - in function body, 207
  - node, 204
- syntax, 181
- type
  - correctness, 202
  - declaration, 184
  - default value, 199
  - elementary, 184
  - expression, 190
  - height, 203
  - pointer, 185
  - range, 197
  - simple, 191
  - size, 256
  - table, 190
- variable, 196
  - base address, 256
  - binding, 215
  - dynamic allocation, 184
  - global variables, 193
  - identifiers, 182
  - names, 180
  - subvariable, 196, 260
  - type, 201
  - value, 202
  - visibility rule, 215
- caller stack, 269
- circuit, 51
  - as a graph, 53
  - cost, 55
  - depth, 55
  - evaluation, 53
  - for a function, 55
  - signals, 51
- clock enable input, 61
- clocked circuit, *see* digital clocked circuit
- code region, 255
- command and status, *see* hard disk ports
- compiler
  - consistency, 253
    - of data, 261
    - of pointers, 261
    - of program counter, 265
    - of return addresses, 269
    - of return destinations, 262
    - of the code, 263
  - encoding, 260
  - memory map, 254
- computation
  - hardware, 62
  - MIPS, 112
  - normal, 406
  - processor equivalent, 407
- concatenation
  - of sequences, 19
  - of sets, 20
- concrete kernel, 438
- conditional-sum adder, *see* CSA
- configuration
  - C0 equivalent, 360
  - of C0, 199
  - of hardware, 62
  - of MIPS, 112
  - of sequential processor, 123
- congruence mod, *see* equivalence mod
- consistency, *see* compiler
- constant, *see* C0
- context-free grammar, 160
  - ambiguity, 169
  - for arithmetic expressions, 170
  - for Boolean expressions, 169
  - for Boolean variables, 161
  - of C0, 181
- cost, *see* circuit
- counting, 8
- CSA, 83
- current instruction, *see* MIPS ISA
- CVM, 430

- correctness theorem, 450
- primitives, 430, 436, 452
- DAG, *see* directed acyclic graph
- dangling pointer, 203
- decimal numbers, 11
- decoder, 59
- decomposition lemma, 35
- default value, *see* C0 type
- depth, *see* circuit
- derivation, *see* context-free grammar, 166
  - generated language, 168
- derivation tree, 161, 165
  - border word, 161, 166
  - composition, 167
  - labeling, 165
  - representation in C0, 186
  - subtree, 167
- difference equation, 82
  - solution, 84
- digital clocked circuit, 61
- digital model, *see* digital clocked circuit
- direct addressing, 120
- directed acyclic graph, 27
  - depth, 28
- directed graph, 25
  - indegree, 26
  - length, 26
  - outdegree, 26
  - path, 26
  - sink, 27
  - source, 27
- disjunctive normal form, 45
- dispatcher, 498
- displacement, 257
- DNF, *see* disjunctive normal form
- effective address, *see* MIPS ISA
  - generalized, 401
- empty sequence, 19
- encoding of values, *see* compiler
- equality tester, 59
- equivalence mod, 21
  - equivalence relation, 21
  - properties, 22
  - solutions, 24
  - system of representatives, 23
- exception, *see* interrupt
  - registers (esr, eca, epc, edata), 377
- execute cycle, *see* sequential processor
- exponentiation, 12
- expression, *see* C0
- father, *see* rooted tree
- fetch cycle, *see* sequential processor
- finite sequence, 18
  - prefix, 19
  - subsequence, 20
- flattened sequence, 189
- full adder, 56
- function table, *see* C0 function
- garbage collection, 355
  - garbage collected version, 361
- gates, 51
  - as functions, 54
- general-purpose register, 112
  - implementation, 75
  - in sequential processor, 132
  - write signal, 122
- generalized circuit model, 86
- geometric sum, 24
- global memory, *see* C0
  - pointer, *see* base pointer
  - region, 255
- global variable, *see* C0 variable
- GPR, *see* general-purpose register
- half adder, 56
- hard disk, 391
  - driver, 417
  - model, 402
  - ports, 391, 392
- hardware
  - computation, *see* computation
  - configuration, *see* configuration
- head, 18
- heap, *see* C0
  - isomorphisms, 347
  - pointer, 255
  - region, 255
- Hilbert  $\epsilon$ -Operator, 17
- I-type instructions, *see* MIPS ISA
- I/O-ports, 391
- identifier, *see* C0 variable
- illegal instruction interrupt, 381
- immediate constant, *see* MIPS ISA

- implicit quantification, 41, 121
- incrementer, 81
  - carry-chain, 82
- induction proof, 8
- instruction decoder, *see* sequential processor
- instruction set architecture, *see* MIPS ISA
- integer numbers, 17
- intermediate result, *see* sequential processor
- interrupt, 376
  - JISR, 378
  - level, 379
  - masking, 379
  - resume type, 376
  - return from exception, 380
  - semantics, 379
- interval, 17
  - of bit strings, 259
- inverter, 58
  
- J-type instructions, *see* MIPS ISA
- jump and link, *see* MIPS ISA
- jump to interrupt service routine, *see* interrupt JISR
  
- labeling, *see* derivation tree
- leaves, *see* rooted tree
- left value, 212
- link address, 119
- literal, 45
- load, *see* MIPS ISA
- logical right shift, 96
  
- memory
  - components, 66
  - of MIPS, 112
- memory map
  - of C0 compiler, *see* compiler
  - of CVM, 461, 471, 481
- memory mapped I/O, 396
- memory system, 395
- memory word, 123
- MIPS ISA, 110
  - ALU-operations, 115
  - branches and jumps, 118
    - branches, 119
    - J-type jumps, 119
    - jump and link, 119
    - R-type jumps, 119
  - computation, 112
    - configuration, 112
      - next configuration, 112, 122
  - implementation, *see* sequential processor
  - instructions
    - current instruction, 113
    - decoding, 114
    - I-type, 110, 113
    - immediate constant, 114
    - instruction fields, 113
    - J-type, 111, 113
    - R-type, 111, 113
  - loads and stores, 120
    - effective address, 120
    - loads, 122
    - stores, 121
  - op-code, 113
  - shift operation, 118
  - software conditions, 124
  - transition function, 112
- misalignment, 113, 120
- modulo operator, 23
- monomial, 45
- multiplexer, 56
- multiplication, 12
- MUX, *see* multiplexer
  
- natural numbers, 8, 17
- negative signal, *see* arithmetic unit
- nondeterminism, 370, 398
- nonterminal, *see* context-free grammar
- NOT-gate, *see* inverter
- null pointer, 180
- number of elements, 17
  
- op-code, *see* MIPS ISA
- OR-tree, 58
- overflow
  - interrupt, 381
  - signal, *see* arithmetic unit
  
- page fault, 387
- page index, *see* page table
- page table, 385
- parallel prefix, 86
- path, *see* directed graph
- PC, *see* program counter
- Peano's axioms, 8
- pigeonhole argument, 27
- pointer, *see* C0 type



- chasing, 356
- dereferencing, 217, 292
- ports, *see* I/O-ports
- power up, 64
- PP, *see* parallel prefix
- prefix, *see* finite sequence
- production, *see* context-free grammar
- program annotations, 454
- program counter, 112
  - generalized, 400
- program rest, *see* C0
  
- quantifier, 16
  
- R-label, 279
- R-type instructions, *see* MIPS ISA
- RAM, *see* random access memory
- RAM-ROM, *see* random access memory
- random access memory, 71
  - GPR-RAM, 75, 123
  - RAM-ROM, 73, 123
  - ROM, 73
  - semantics, 72
  - SPR-RAM, 77
  - SRAM, 71
- recursive construction
  - carry-chain adder, 82
  - conditional-sum adder, 83
  - decoder, 60
  - incrementer, 82
  - parallel prefix circuit, 86
- registers, 60, 66
  - semantics, 64
- reset
  - in sequential processor, 125
  - interrupt, 378
  - signal, 62
- resume type, *see* interrupt
- return from exception, *see* interrupt
- right value, 212, 280
- ROM, *see* random access memory
- root, *see* rooted tree
- rooted tree, 28
  - father, 29
  - leaves, 29
  - root, 29
  - son, 29
- scheduler, 438
- school method
  - for addition, 36
  - for division, 149
- scratch memory, 461
- sequence, *see* finite sequence
  - elements, 189
- sequential processor, 123
  - A and B operand, 132
  - C address, 130
  - configuration, 123
  - correctness, 124
  - execute cycle, 128
  - instruction decoder, 128
  - instruction fetch, 125
  - intermediate result, 122
  - next PC, 134
  - reset, 125
  - simulation relation, 124
- shift amount, 114
- shifter, *see* logical right shift
- sign bit, 38
- sign extension, 38
  - sign extended immediate constant, 114
- signal, *see* circuit
- simulation relation
  - for CVM, 430, 451
  - for sequential processor, 124
- software conditions, *see* MIPS ISA
- son, *see* rooted tree
- source program, 253
- special function, *see* CVM primitives
- special purpose register
  - implementation, 77
  - in sequential processor, 376
- SPR, *see* special purpose register
- SRAM, *see* random access memory
- stack, *see* C0
  - pointer, 255
  - region, 255
- start symbol, *see* context-free grammar
- statement, *see* C0
- status register, *see* special purpose register
- store, *see* MIPS ISA
- subsequence, *see* finite sequence
- subtraction algorithm, 39
- subtree, *see* derivation tree
- subvariable, *see* C0 variable
  - in implementation, 342
  - reachable, 340

- support, 45
- swap page address, *see* hard disk ports
- syntactic sugar, 417
- system call
  - instruction, 377
  - interrupt, 381
- system mode, 377
  
- tail, 18
- target program, 253
- terminal, *see* context-free grammar
- translation
  - of expressions, 275
  - of programs, 315
  - of statements, 302
- tree, *see* rooted tree
- tree region, 163
  - ancestor, 165
  - father, 165
  
- two's complement numbers, 37
  - modulo operator, 24
  - properties, 38
- two's complement representation, 38
- type, *see* C0
  - table, *see* C0 type
  
- user mode, 377
  
- variable, *see* C0
- virtual address, *see* address translation
- virtual machine, 431
- visibility rule, *see* C0 variable
  
- word, *see* memory word
- word addressable, 123
  
- zero, *see* Peano's axioms
- zero tester, 59