Appendix C

Supplementary Benchmarks

This is the complete collection of benchmarks run in conjunction with this thesis. See Appendix A for how to obtain the numerical values and the source code for the programs that generated them.

C.1 Layout and Navigation

The results of the benchmarks described in Sections 5.3.2 and 5.3.3.

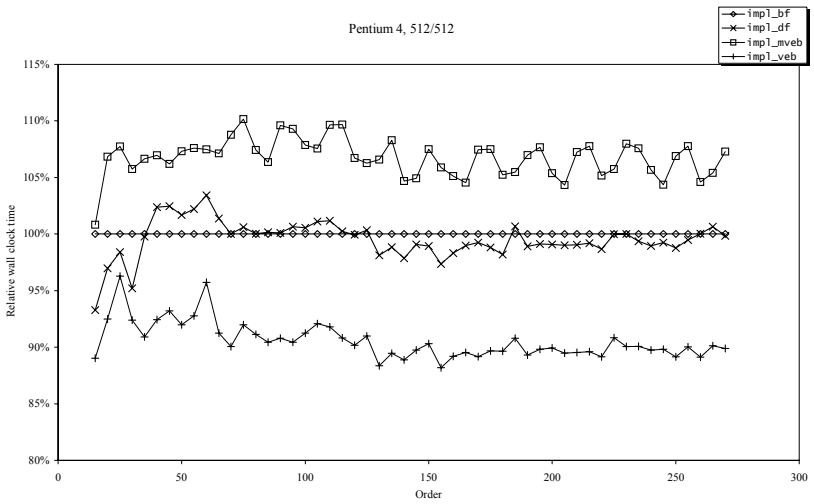

Chart C-1. Implicit layout on Pentium 4.

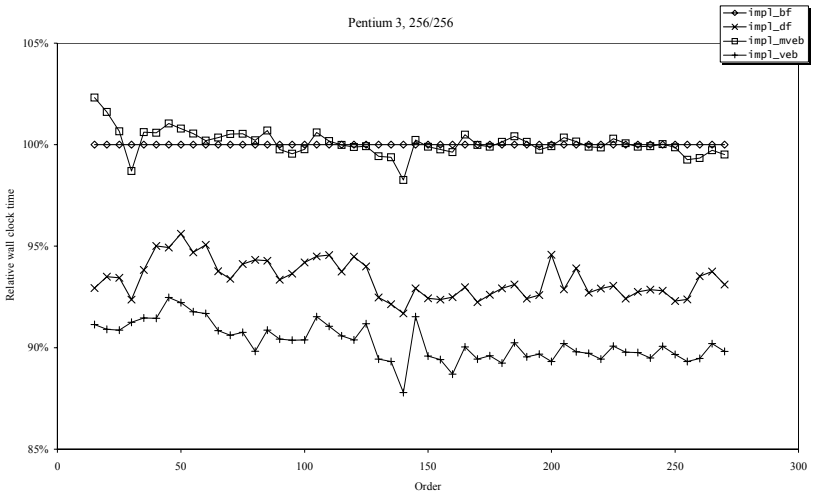

Chart C-2. Implicit layout on Pentium 3.

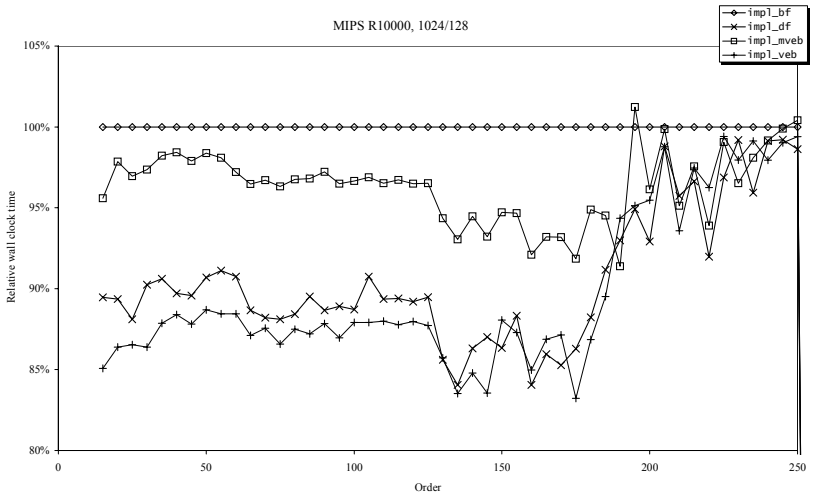

Chart C-3. Implicit layout on MIPS R10000.

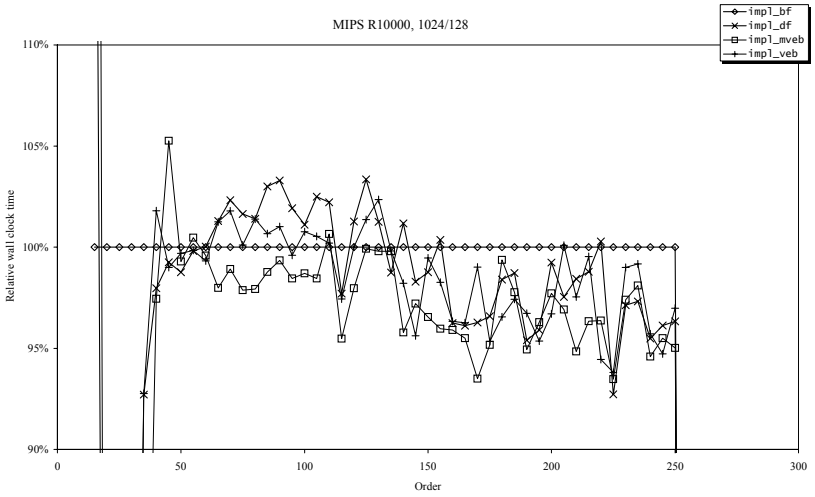

Chart C-4. Implicit layout on MIPS R10000, relative L2 cache misses.

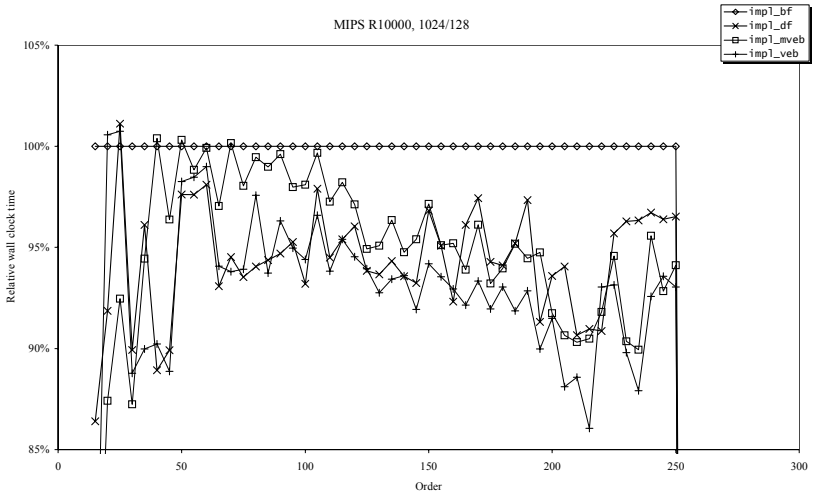

Chart C-5. Implicit layout on MIPS R10000, relative TLB misses.

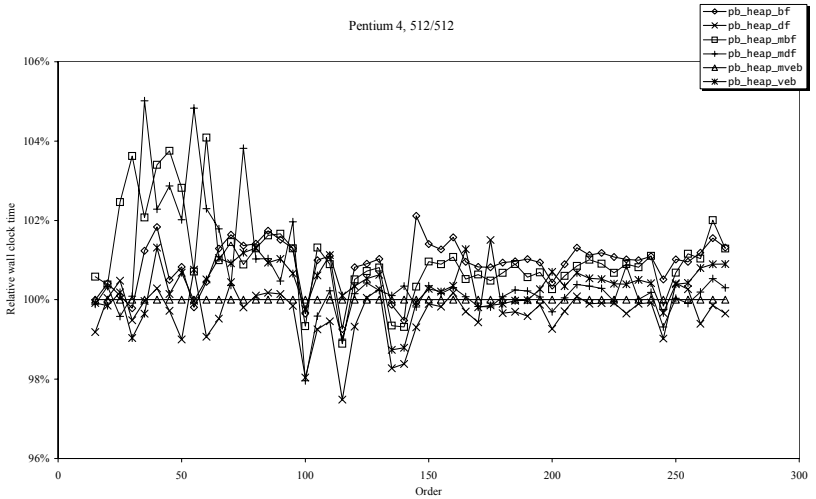

Chart C-6. Layout using std::allocator on Pentium 4.

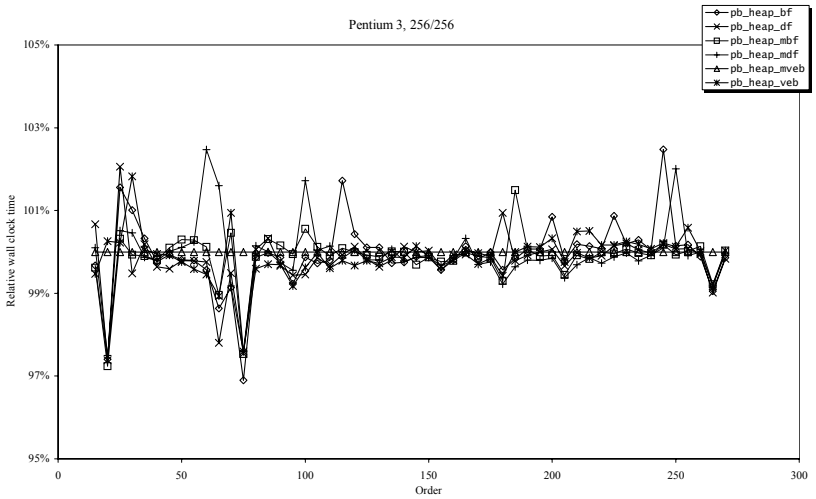

Chart C-7. Layout using std::allocator on Pentium 3.

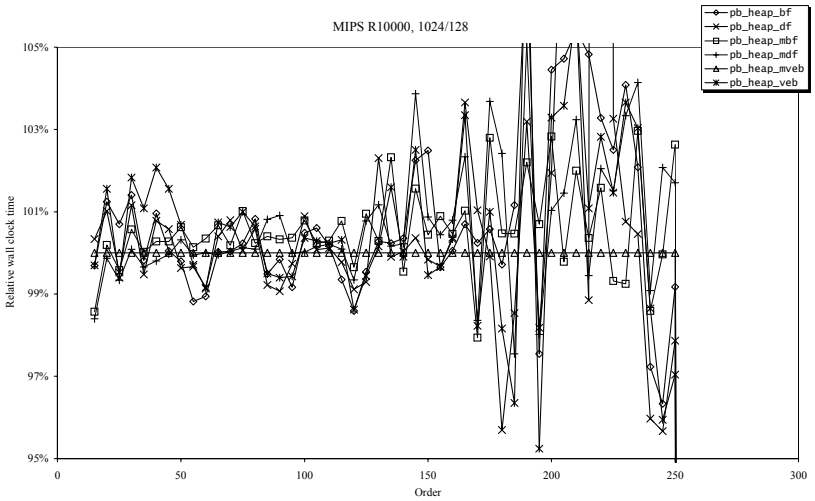

Chart C-8. Layout using std::allocator on MIPS 10000.

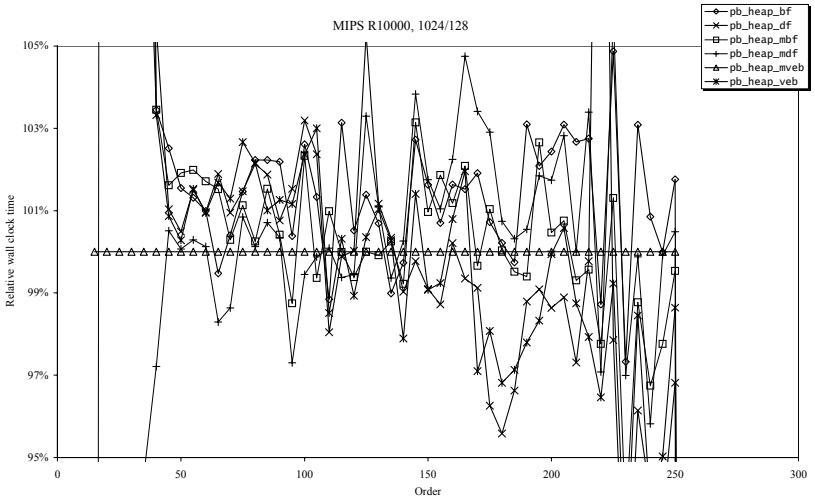

Chart C-9. Layout using std::allocator on MIPS 10000, L2 cache misses.

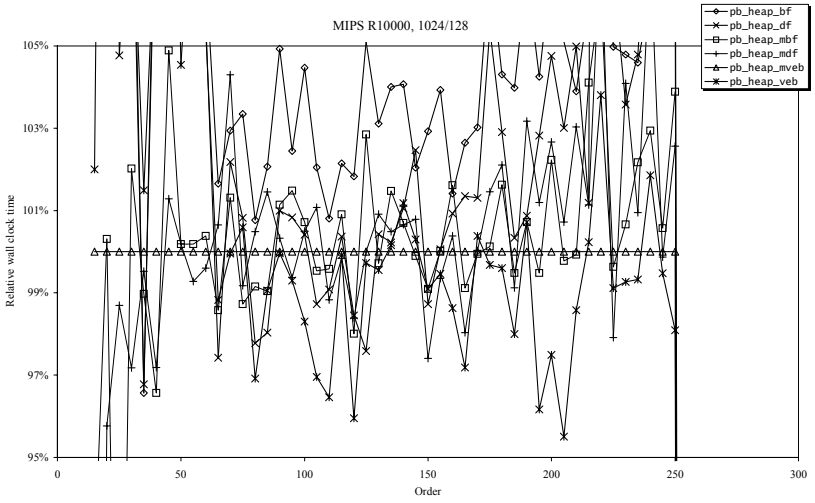

Chart C-10. Layout using std::allocator on MIPS 10000, TLB misses.

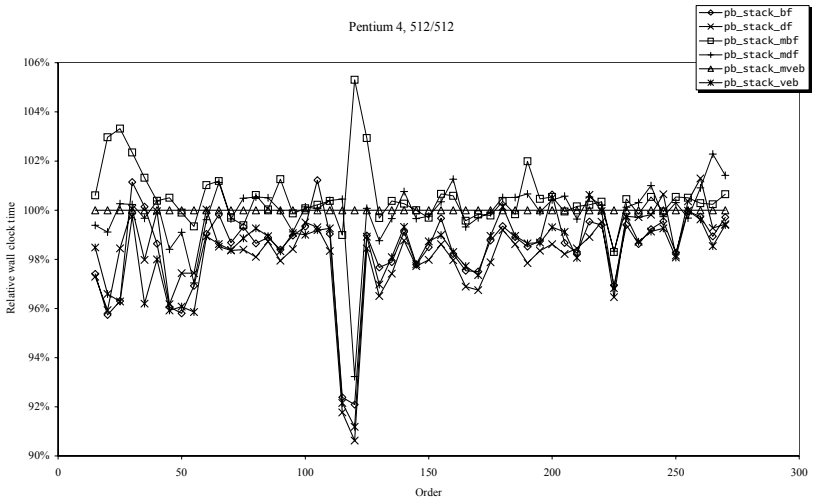

Chart C-11. Layout using stack_allocator on Pentium 4.

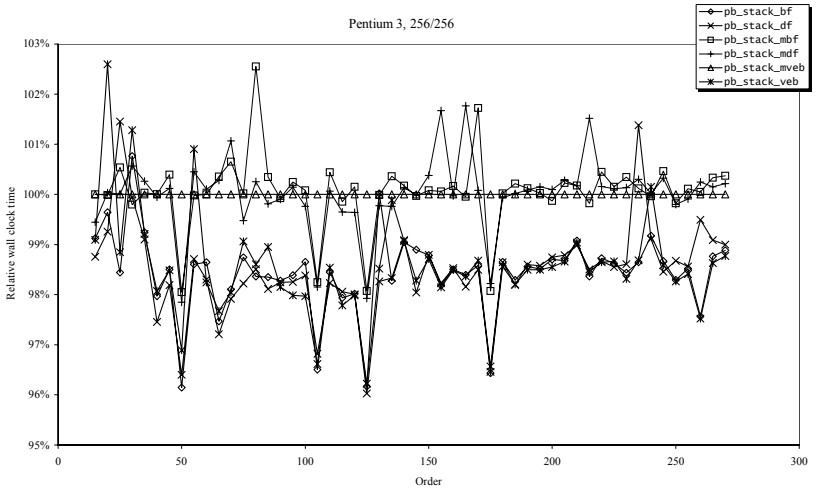

Chart C-12. Layout using stack_allocator on Pentium 3.

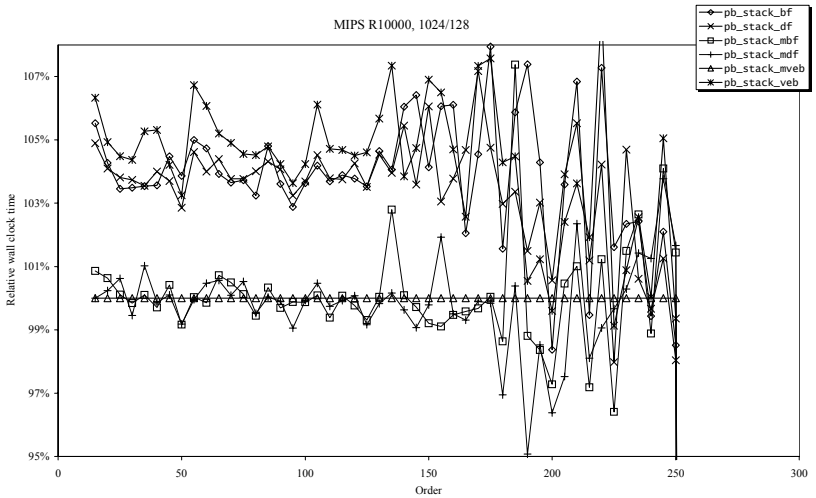

Chart C-13. Layout using stack_allocator on MIPS 10000.

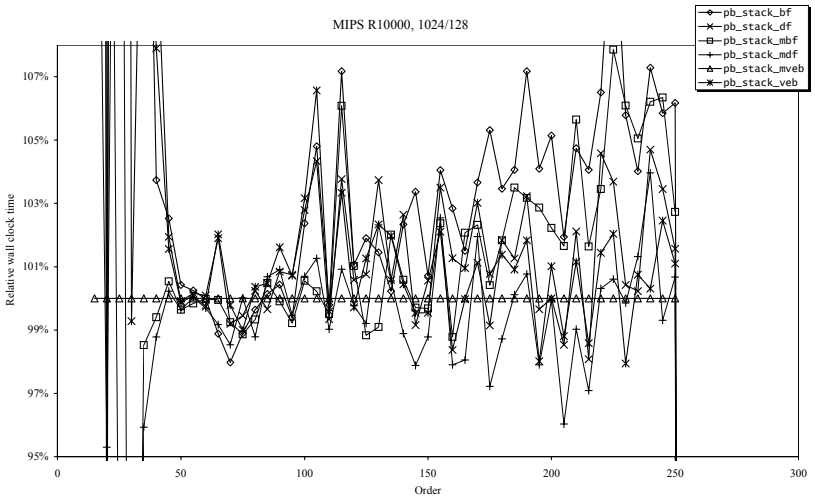

Chart C-14. Layout using stack_allocator on MIPS 10000, L2 cache miss.

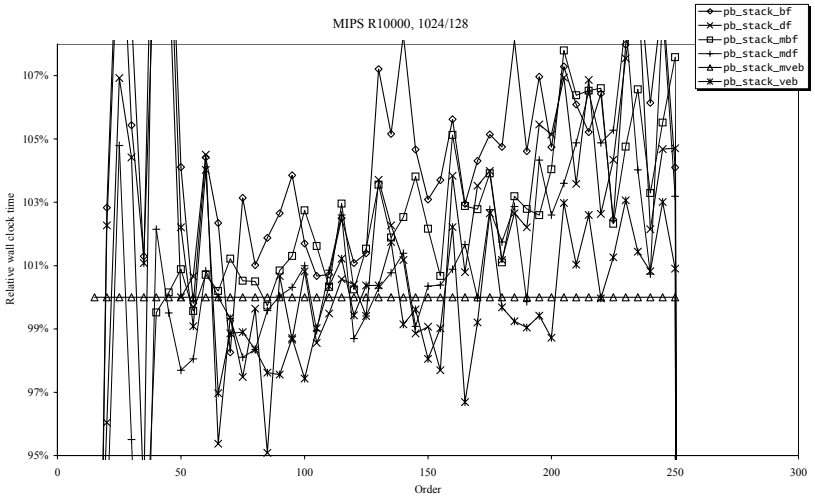

Chart C-15. Layout using stack_allocator on MIPS 10000, TLB miss.

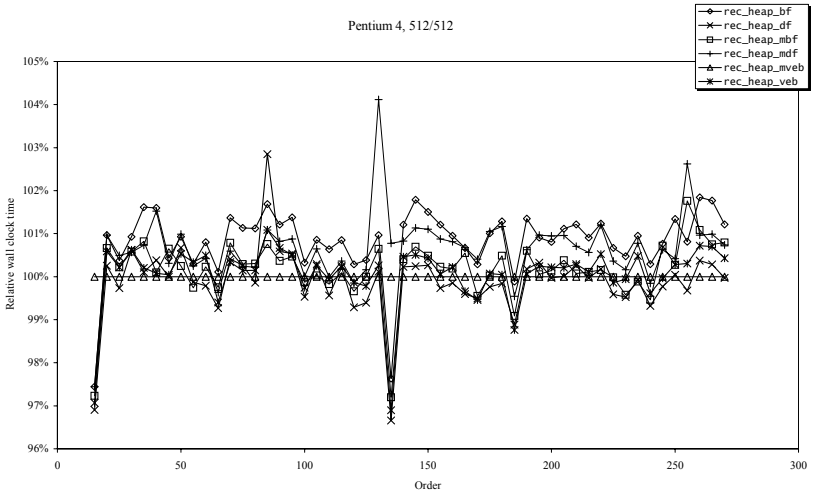

Chart C-16. Recursive fill, heap, Pentium 4.

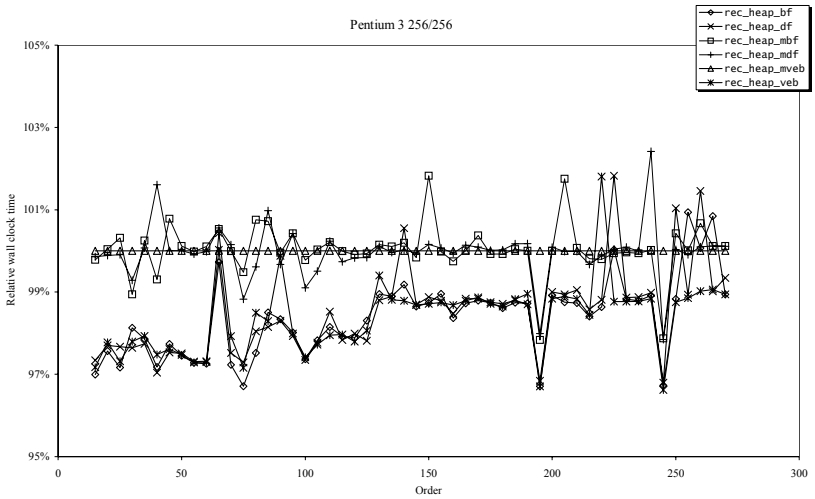

Chart C-17. Recursive fill, heap, Pentium 3.

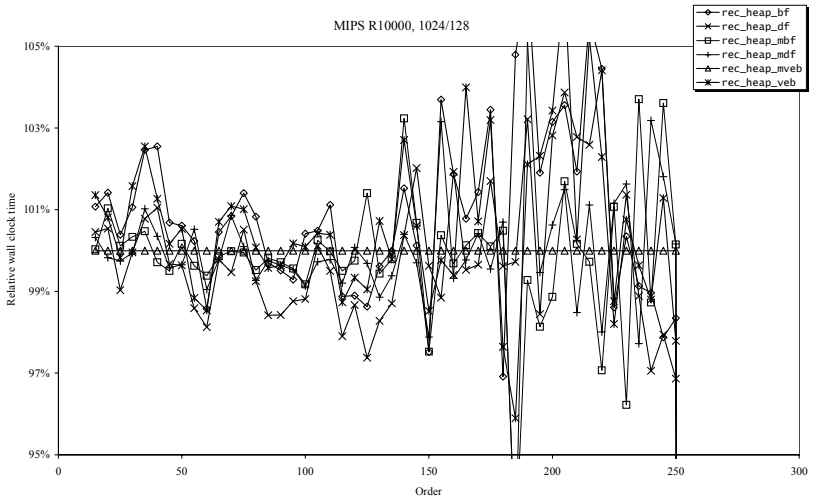

Chart C-18. Recursive fill, heap, MIPS 10000.

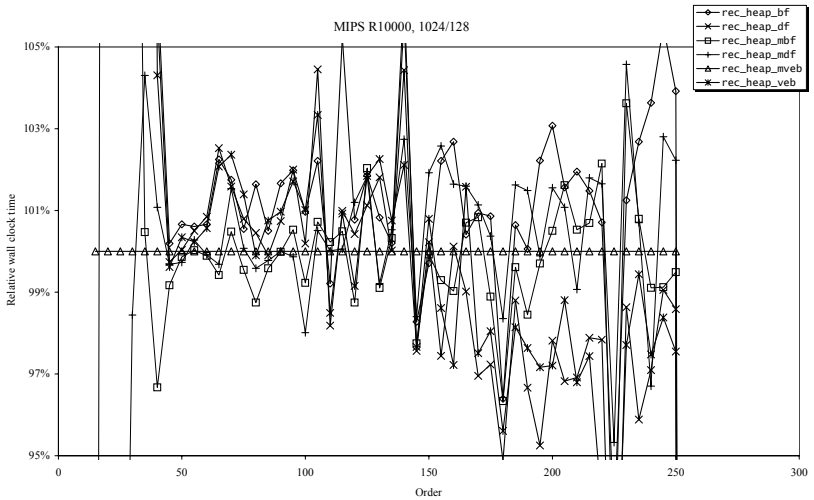

Chart C-19. Recursive fill, heap, MIPS 10000, L2 cache miss.

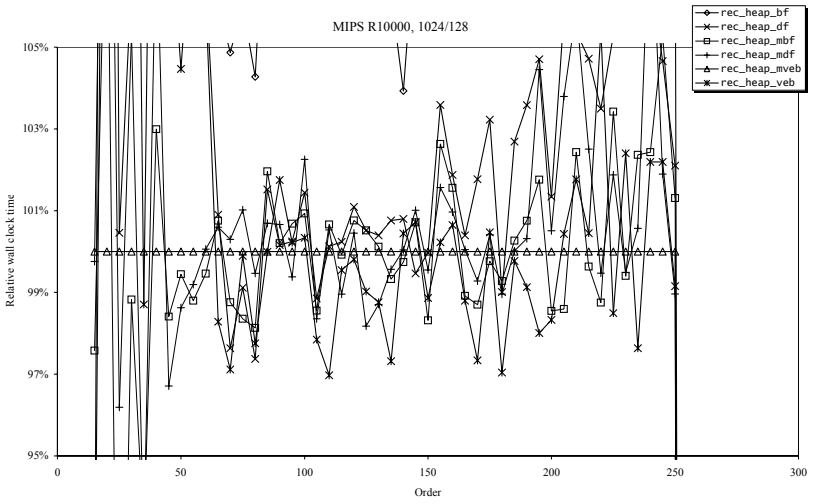

Chart C-20. Recursive fill, heap, MIPS 10000, TLB miss.

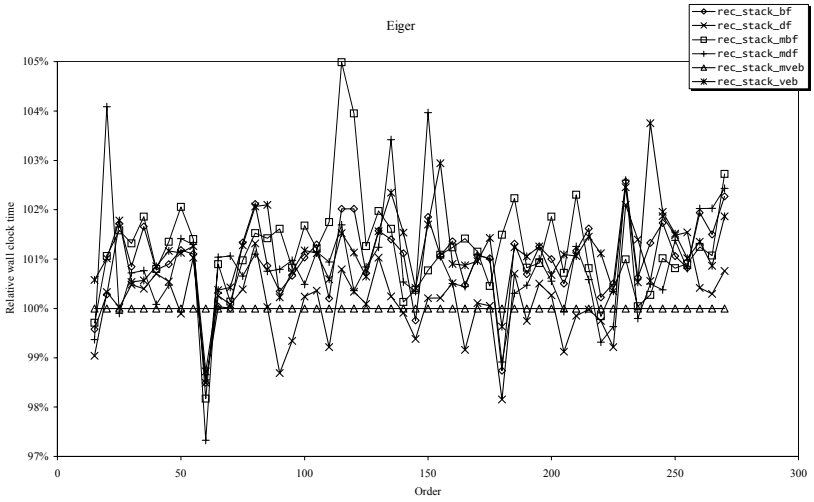

Chart C-21. Recursive fill, stack, Pentium 4.

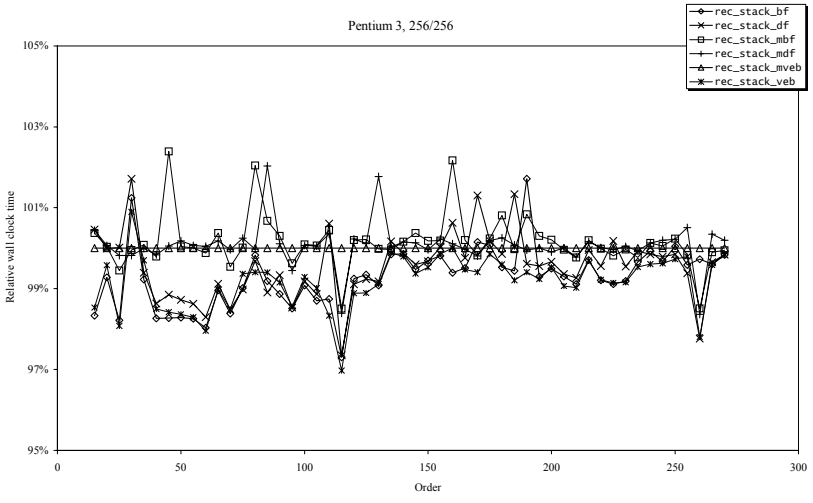

Chart C-22. Recursive fill, stack, Pentium 3.

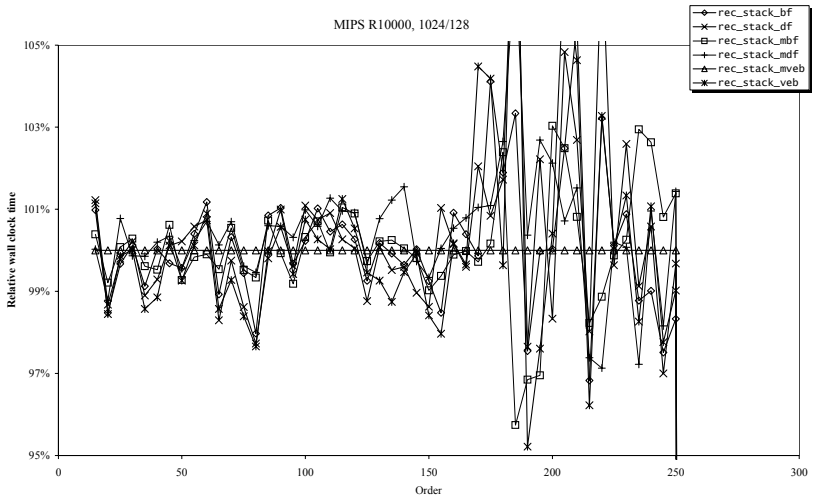

Chart C-23. Recursive fill, stack, MIPS 10000.

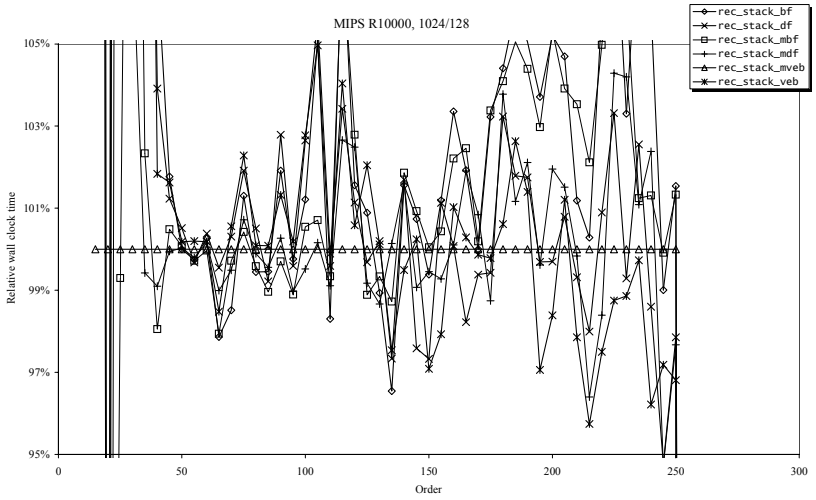

Chart C-24. Recursive fill, stack, MIPS 10000, L2 cache miss.

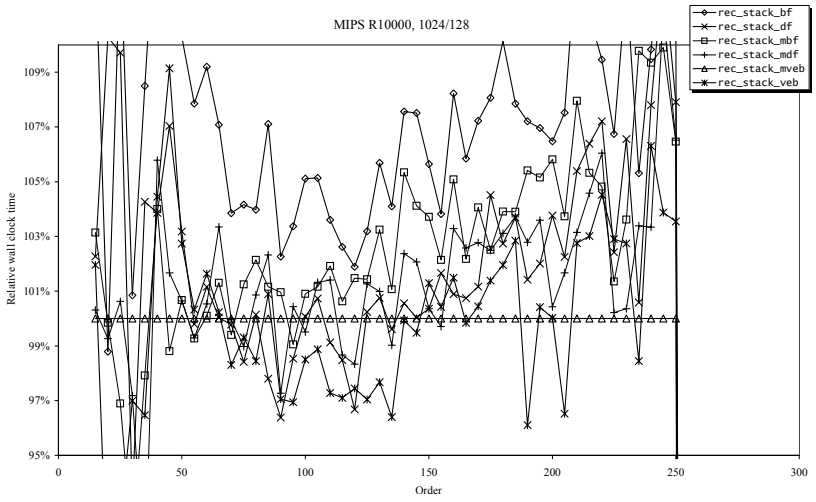

Chart C-25. Recursive fill, stack, MIPS 10000, TLB miss.

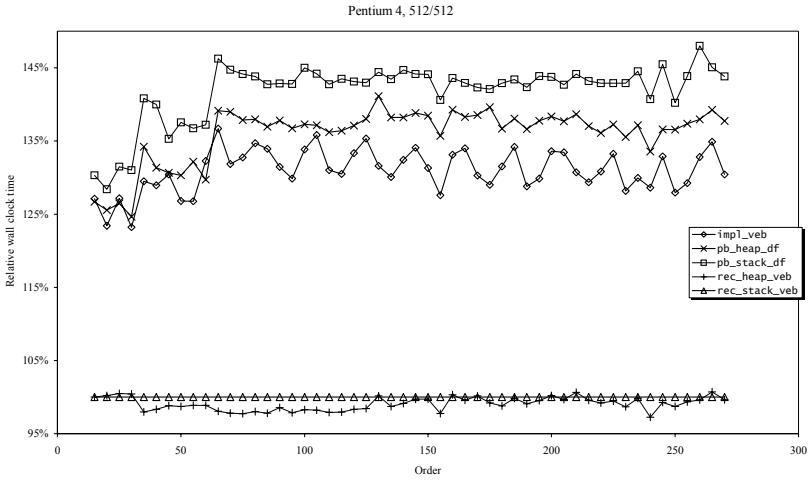

Chart C-26. Final, Pentium 4.

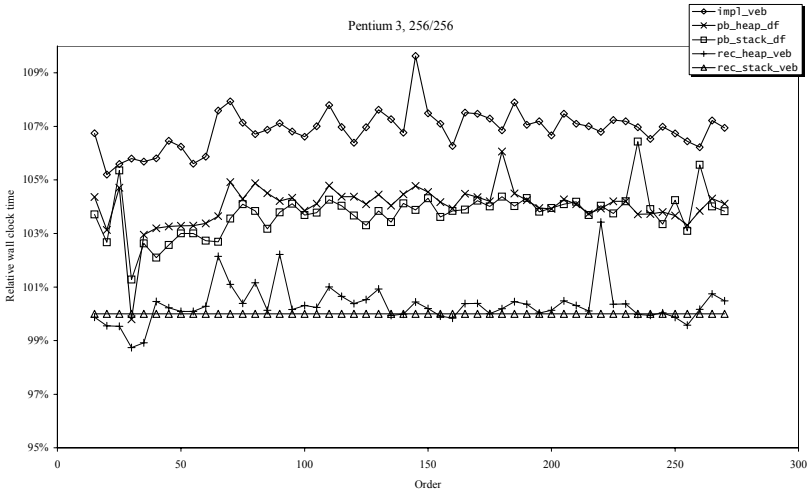

Chart C-27. Final, Pentium 3.

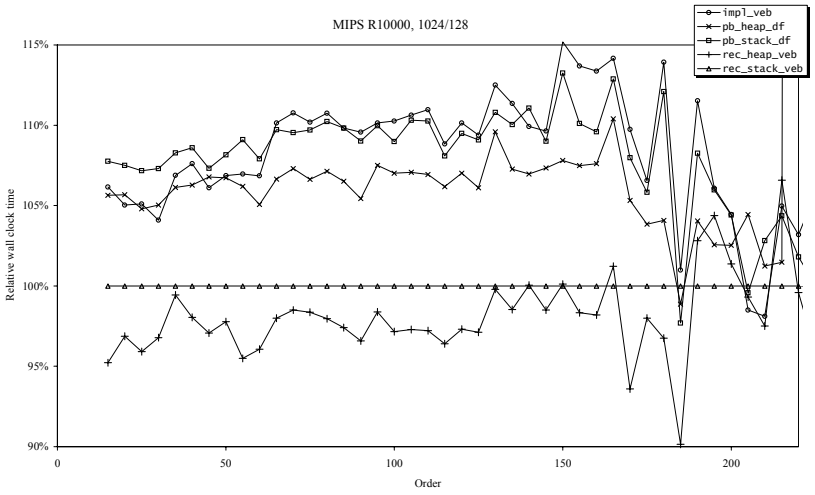

Chart C-28. Final, MIPS 10000.

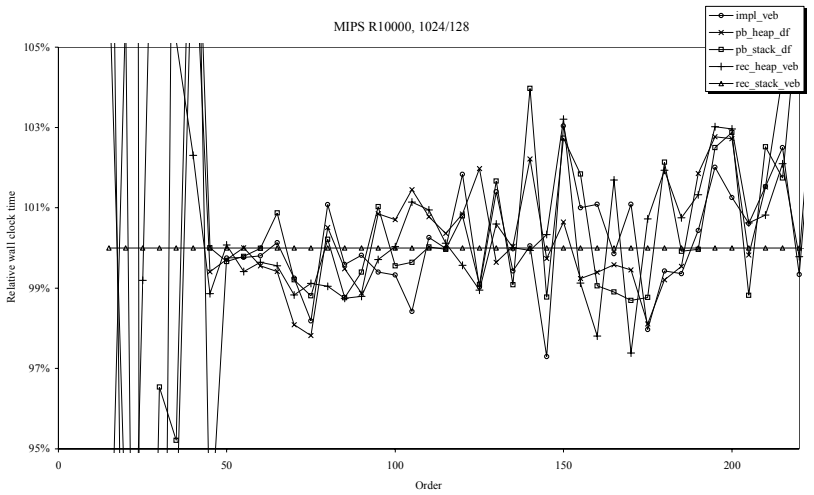

Chart C-29. Final, MIPS 10000 L2 cache misses.

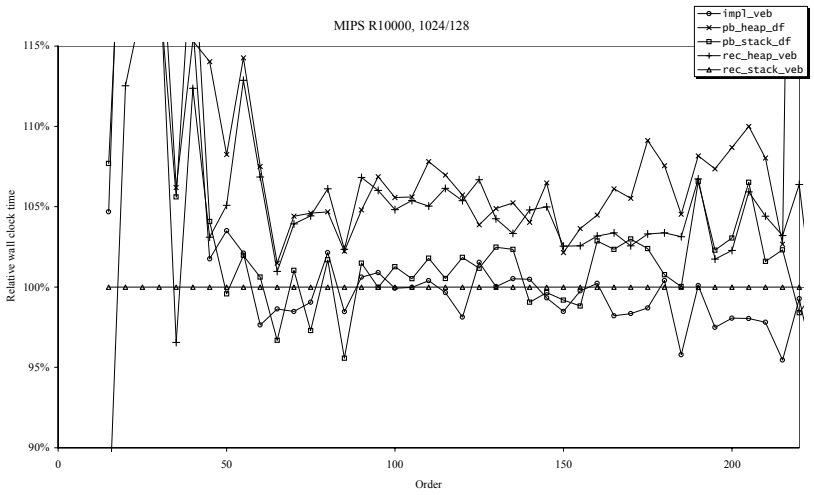

Chart C-30. Final, MIPS 10000 TLB miss.

C.2 Basic Merger

The results of the benchmarks described in Section 5.3.4.

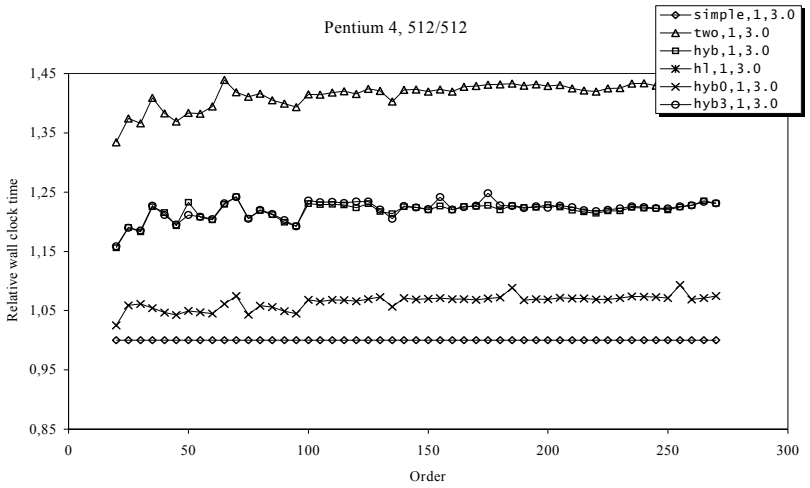

Chart C-31. Basic mergers, $(\alpha,d) = (1,3)$, Pentium 4.

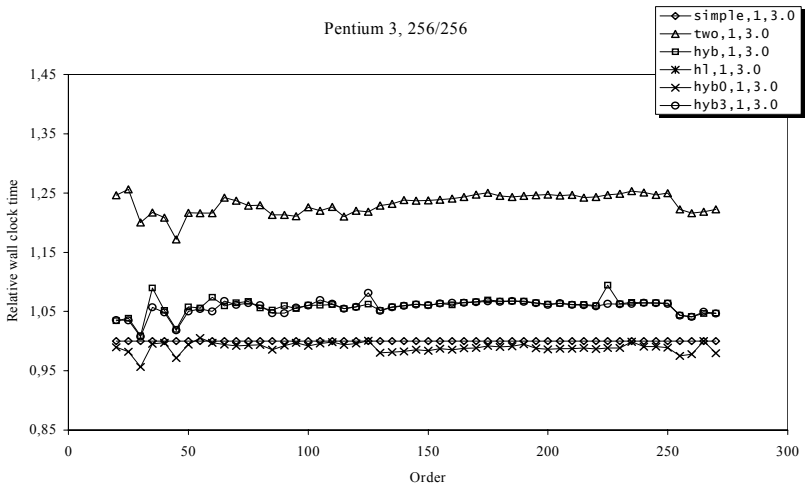

Chart C-32. Basic mergers, $(\alpha,d) = (1,3)$, Pentium 3.

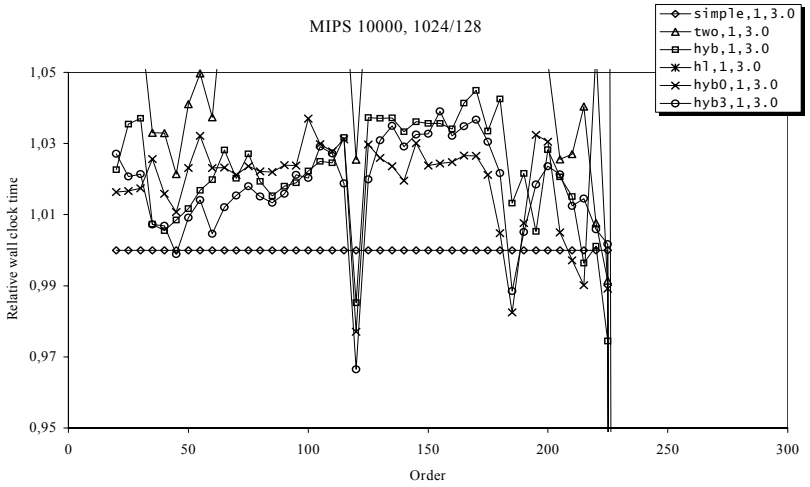

Chart C-33. Basic mergers, $(\alpha,d) = (1,3)$, MIPS 10000.

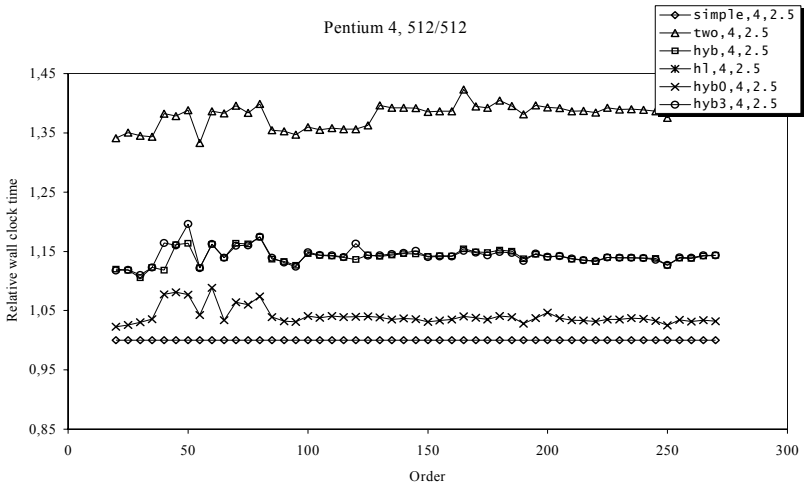

Chart C-34. Basic mergers, $(\alpha,d) = (4,2.5)$, Pentium 4.

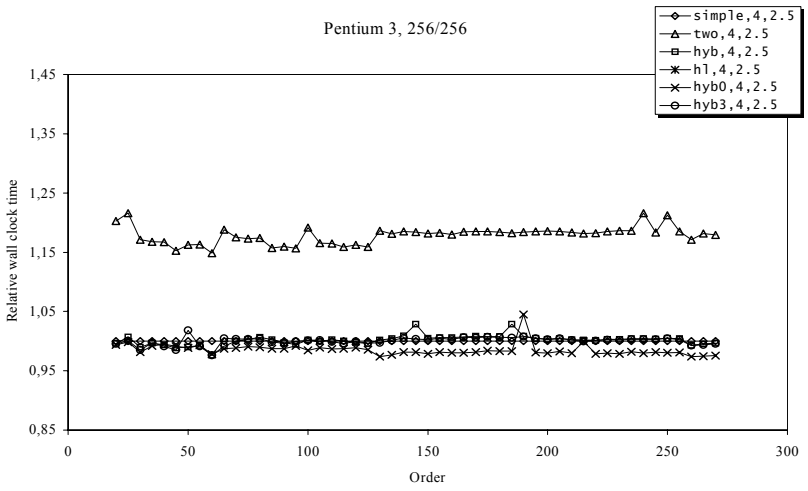

Chart C-35. Basic mergers, $(\alpha,d) = (4,2.5)$, Pentium 3.

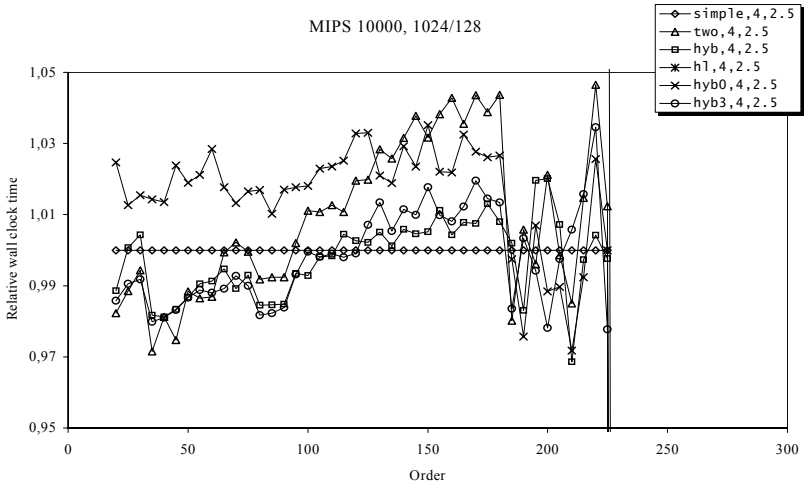

Chart C-36. Basic mergers, $(\alpha,d) = (4,2.5)$, MIPS 10000.

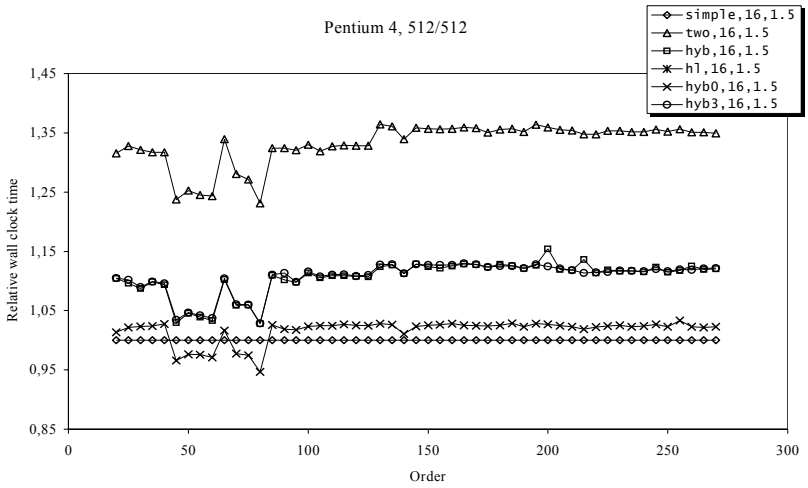

Chart C-37. Basic mergers, $(\alpha,d) = (16,1.5)$, Pentium 4.

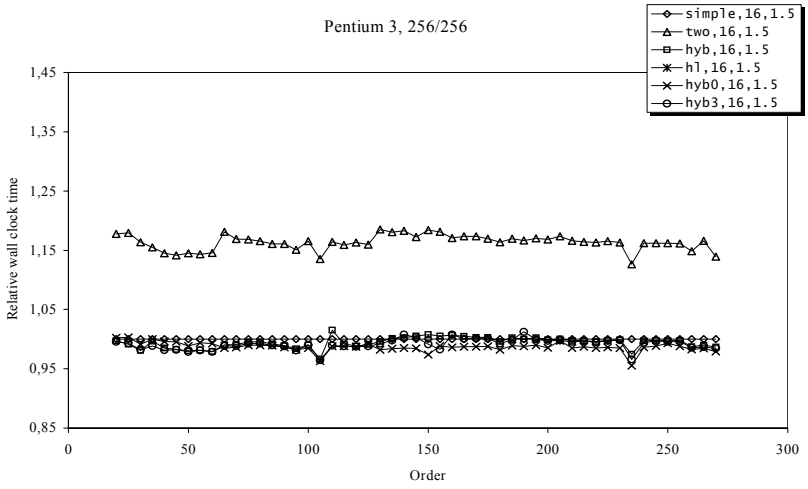

Chart C-38. Basic mergers, $(\alpha,d) = (16,1.5)$, Pentium 3.

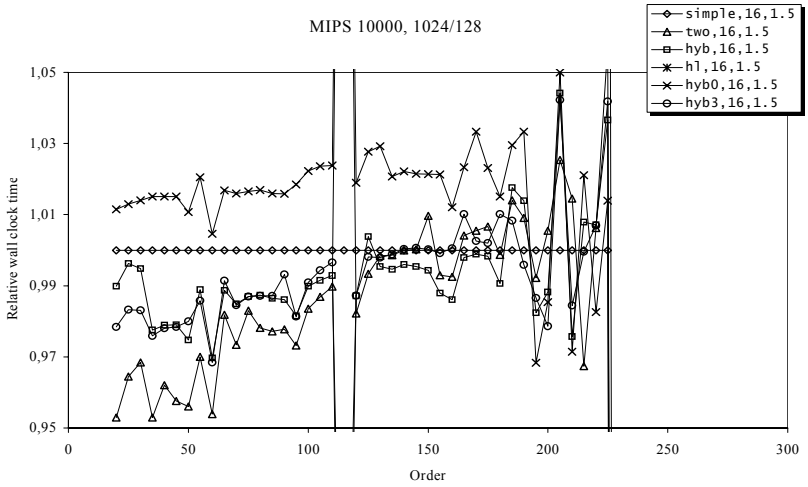

Chart C-39. Basic mergers, $(\alpha,d) = (16,1.5)$, MIPS 10000.

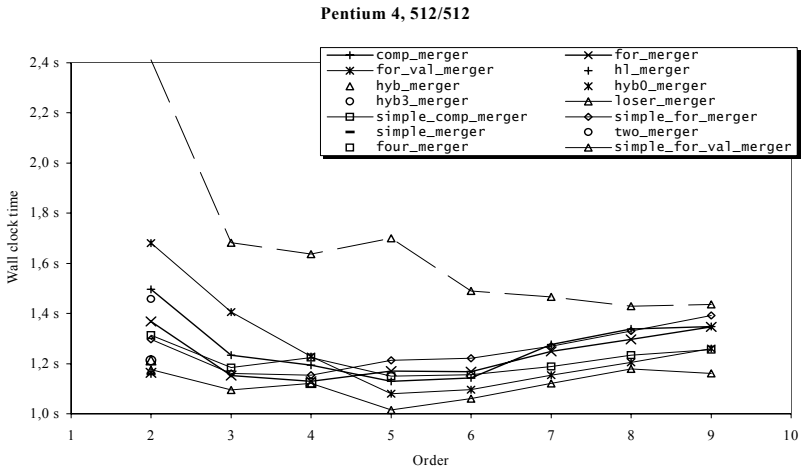

Chart C-40. Basic mergers, Pentium 4.

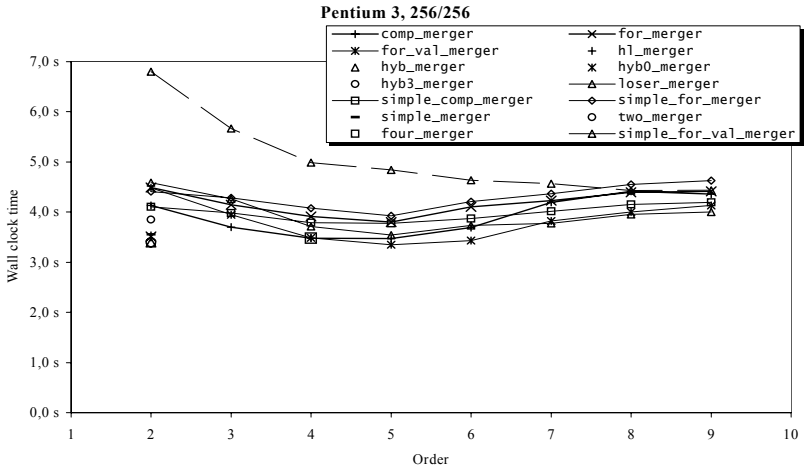

Chart C-41. Basic mergers, Pentium 3.

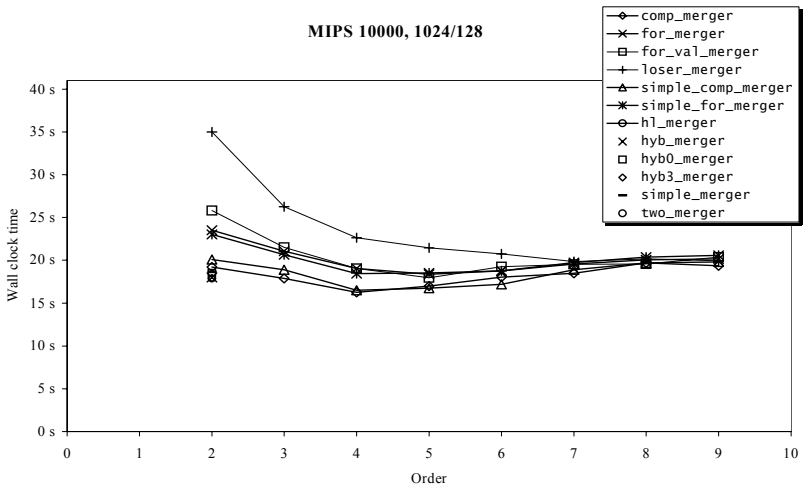

Chart C-42. Basic mergers, MIPS 10000.

C.3 Merger Caching

The results of the benchmarks described in Sections 5.4.1 and 5.4.2.

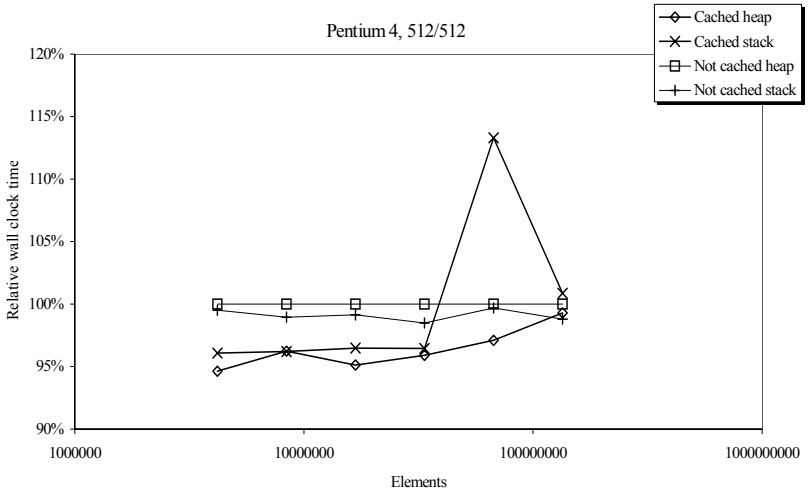

Chart C-43. Effects of merger caching, Pentium 4.

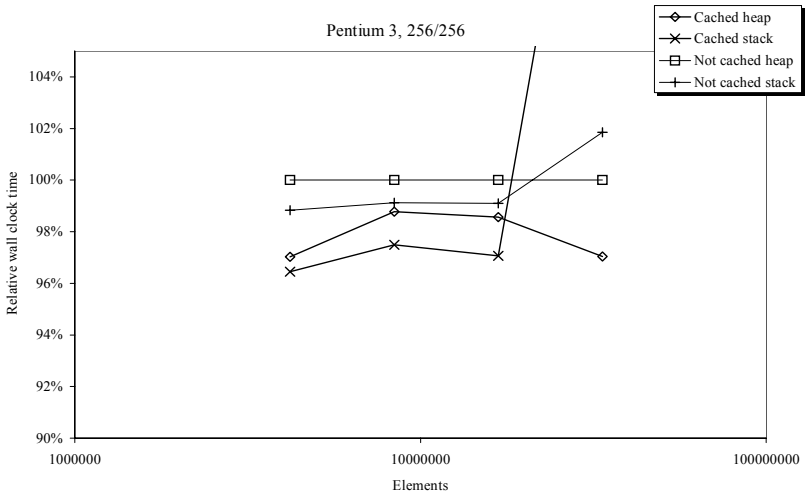

Chart C-44. Effects of merger caching, Pentium 3.

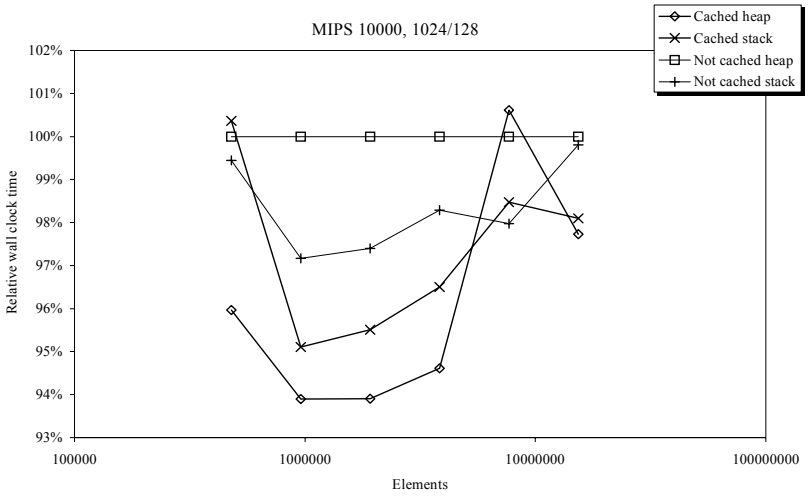

Chart C-45. Effects of merger caching, MIPS 10000.

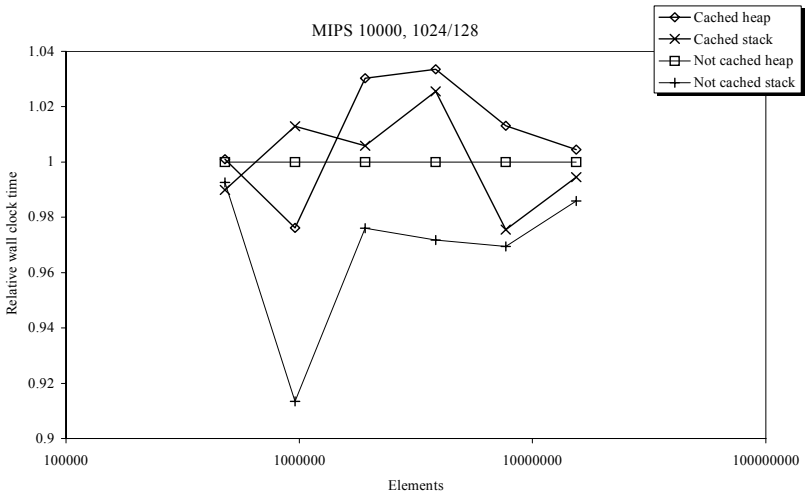

Chart C-46. Effects of merger caching, MIPS 10000, L2 cache misses.

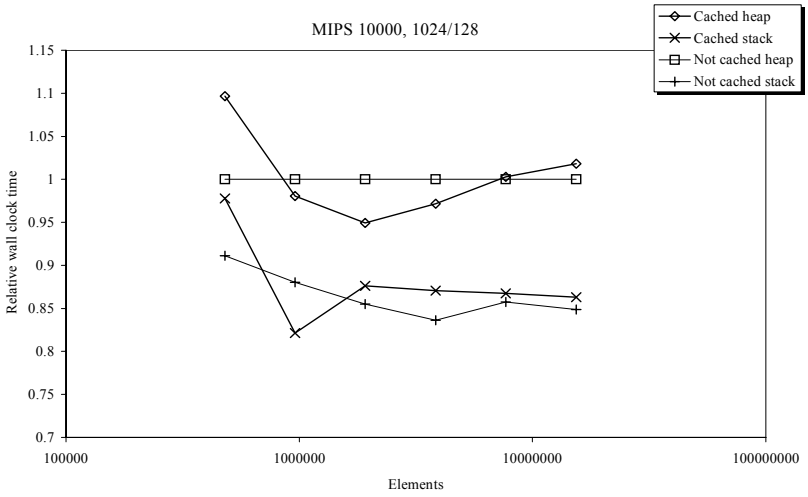

Chart C-47. Effects of merger caching, MIPS 10000, TLB misses.

C.4 Base Sorting Algorithms

The results of the benchmarks described in Section 5.4.3.

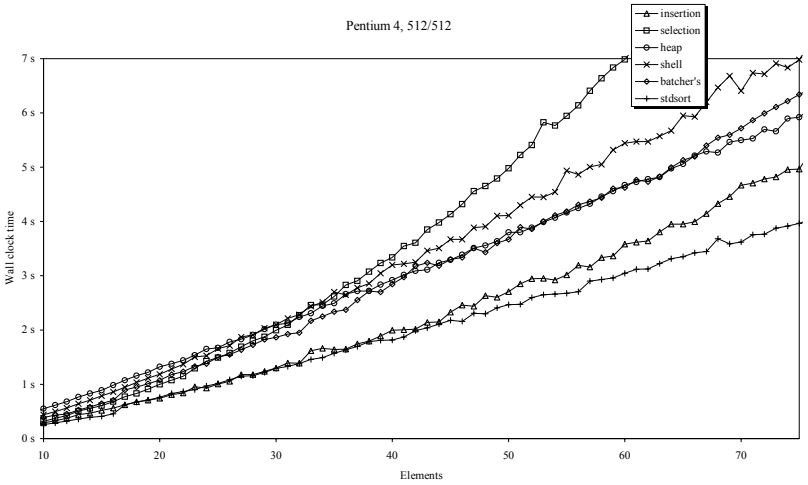

Chart C-48. Base sorting algorithms, Pentium 4.

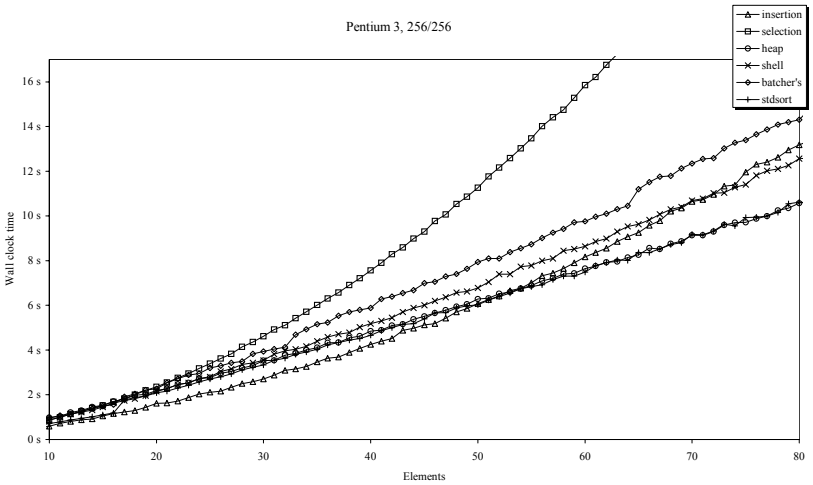

Chart C-49. Base sorting algorithms, Pentium 3.

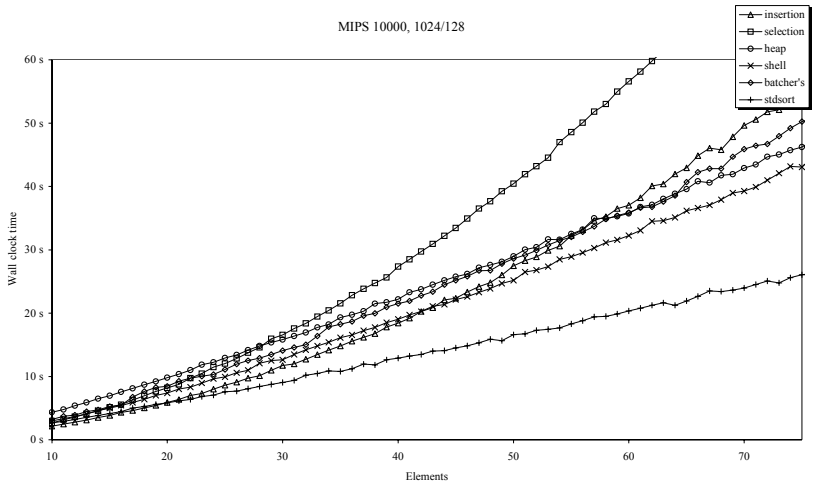

Chart C-50. Base sorting algorithms, MIPS 10000.

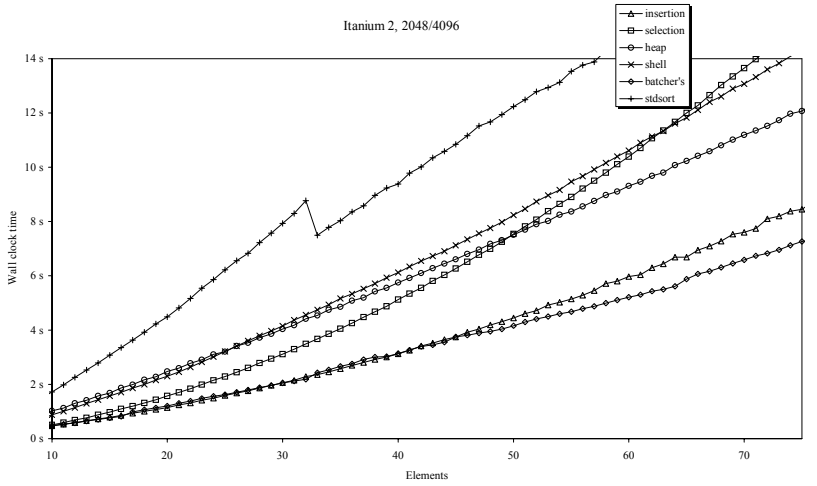

Chart C-51. Base sorting algorithms, Itanium 2.

C.5 Buffer Sizes

The results of the benchmarks described in Section 5.4.4.

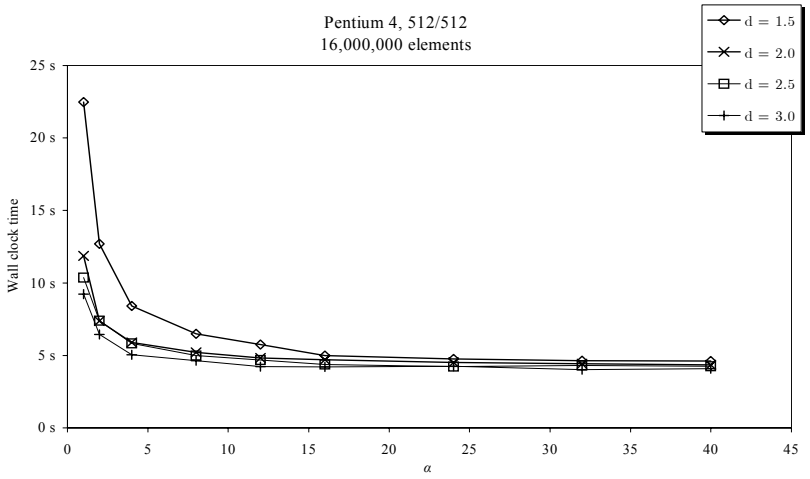

Chart C-52. Buffer parameters, sorting 16,000,000 elements on Pentium 4.

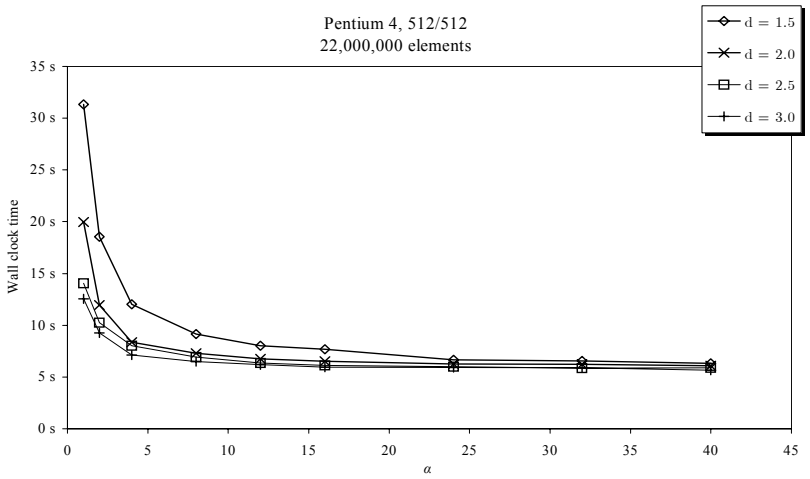

Chart C-53. Buffer parameters, sorting 22,000,000 elements on Pentium 4.

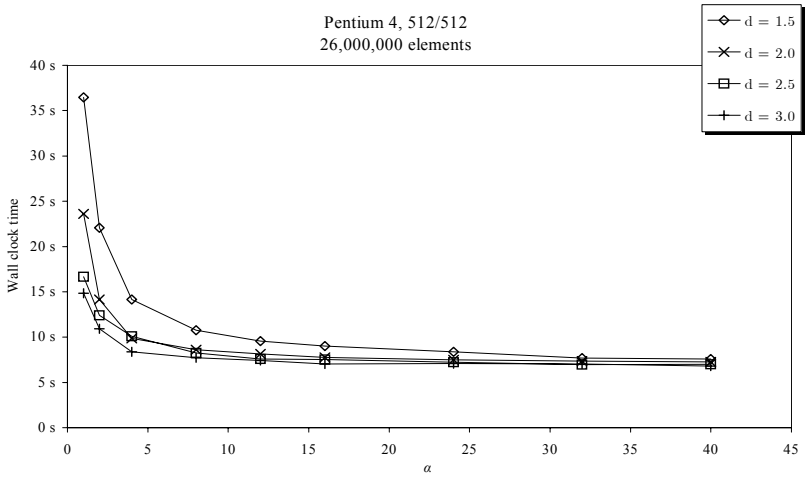

Chart C-54. Buffer parameters, sorting 26,000,000 elements on Pentium 4.

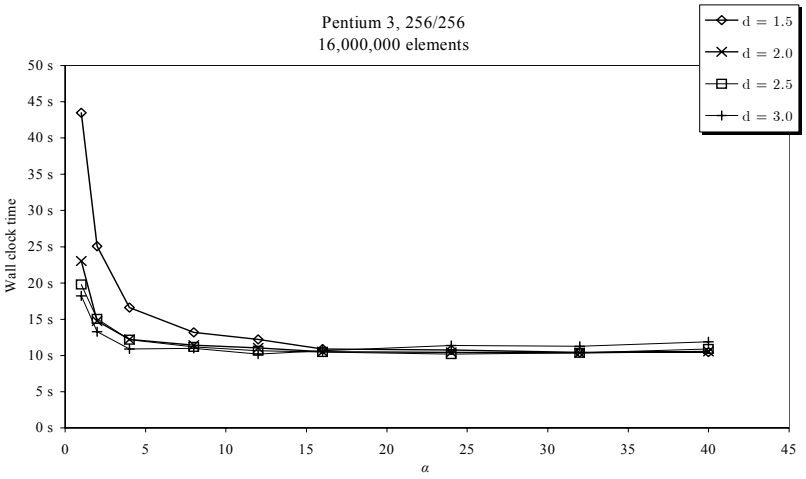

Chart C-55. Buffer parameters, sorting 16,000,000 elements on Pentium 3.

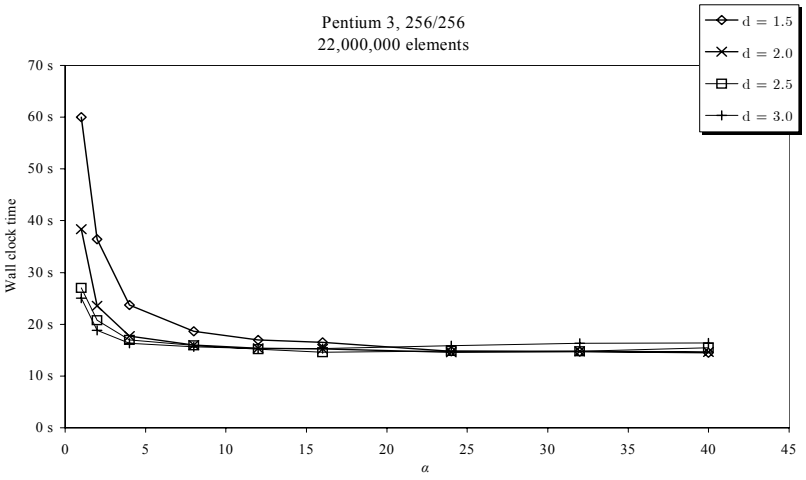

Chart C-56. Buffer parameters, sorting 22,000,000 elements on Pentium 3.

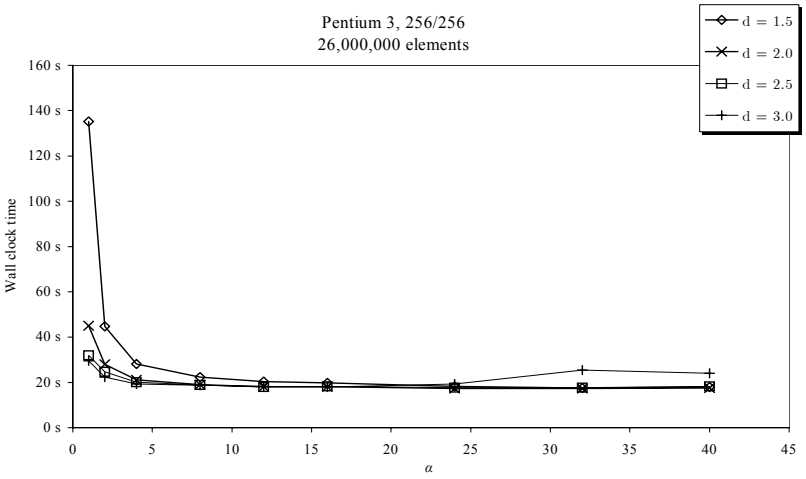

Chart C-57. Buffer parameters, sorting 26,000,000 elements on Pentium 3.

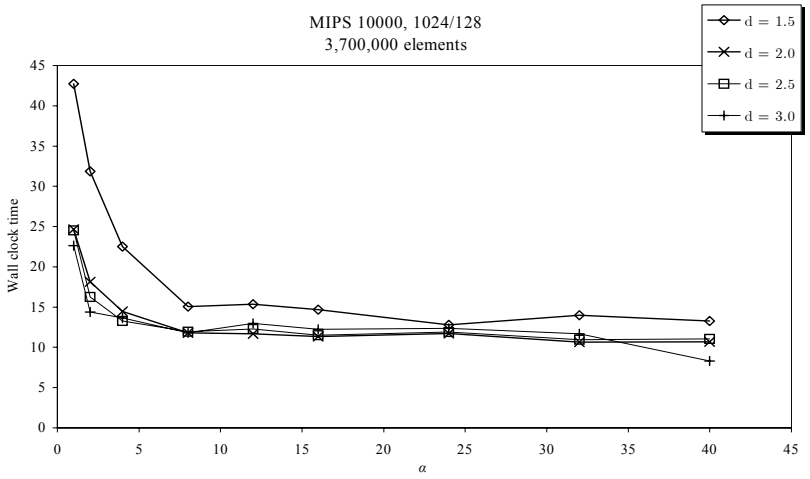

Chart C-58. Buffer parameters, sorting 3,700,000 elements on MIPS.

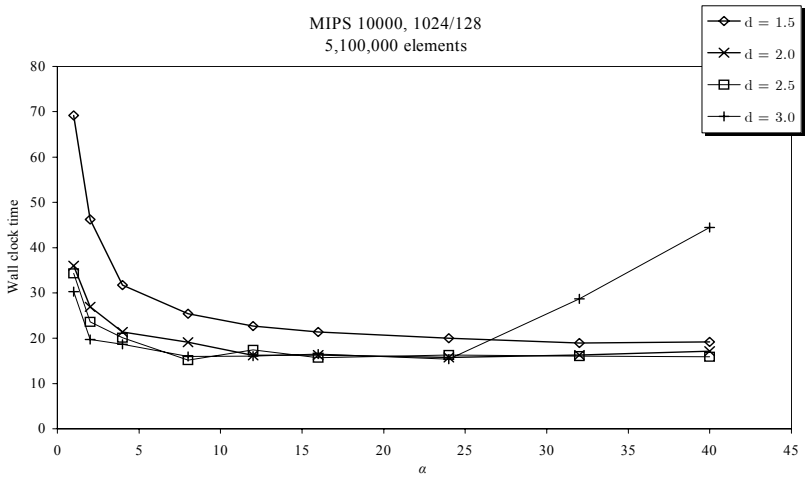

Chart C-59. Buffer parameters, sorting 5,100,000 elements on MIPS.

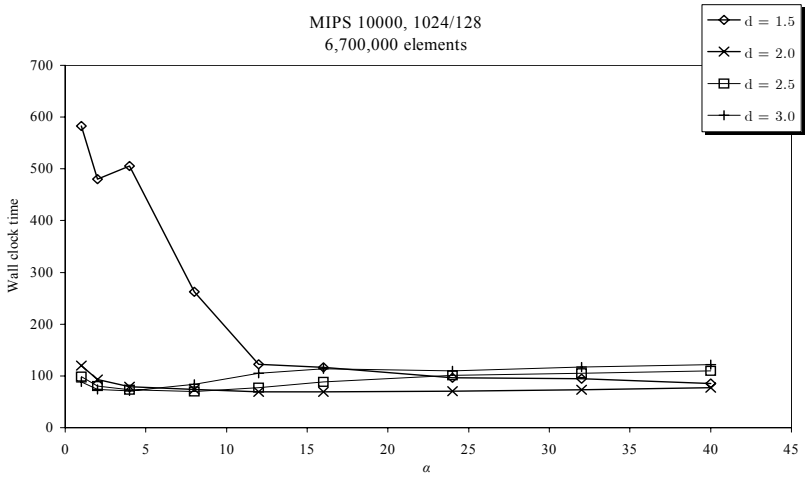

Chart C-60. Buffer parameters, sorting 6,700,000 elements on MIPS.

C.6 Straight Sorting

The results of the benchmarks described in Section 6.1 and 6.2.

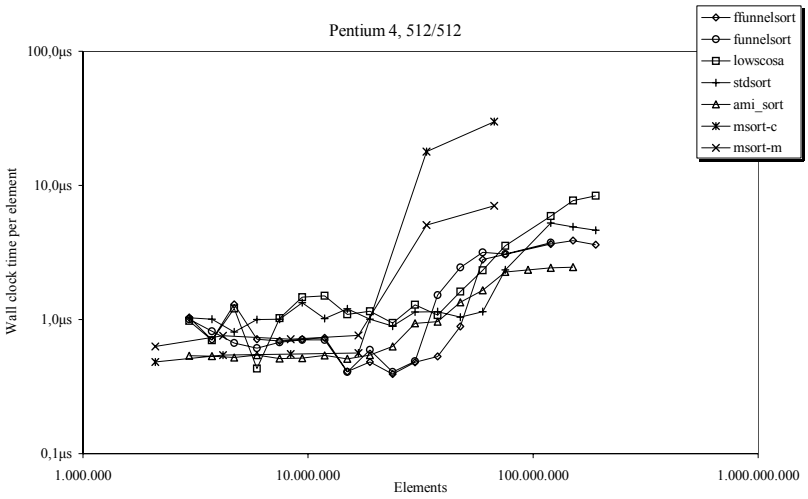

Chart C-61. Wall clock time sorting uniformly distributed pairs on Pentium 4.

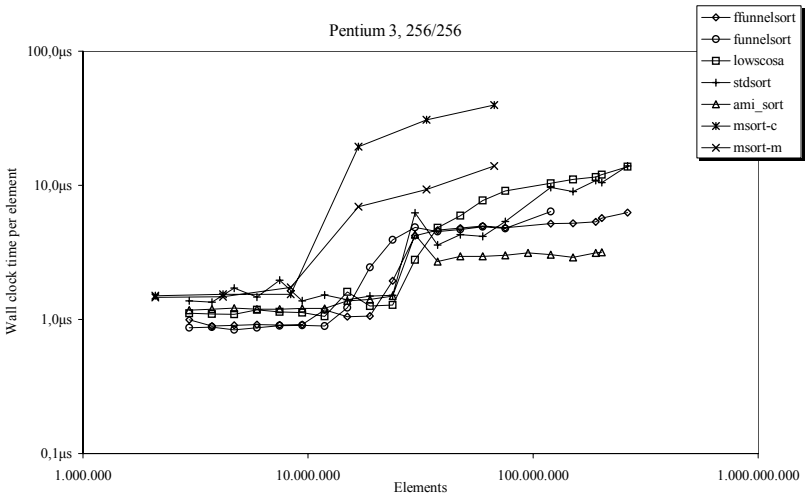

Chart C-62. Wall clock time sorting uniformly distributed pairs on Pentium 3.

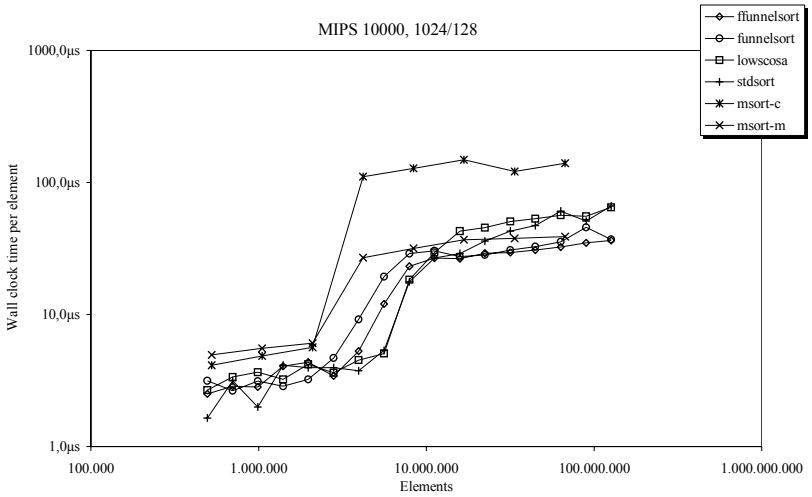

Chart C-63. Wall clock time sorting uniformly distributed pairs on MIPS 10000.

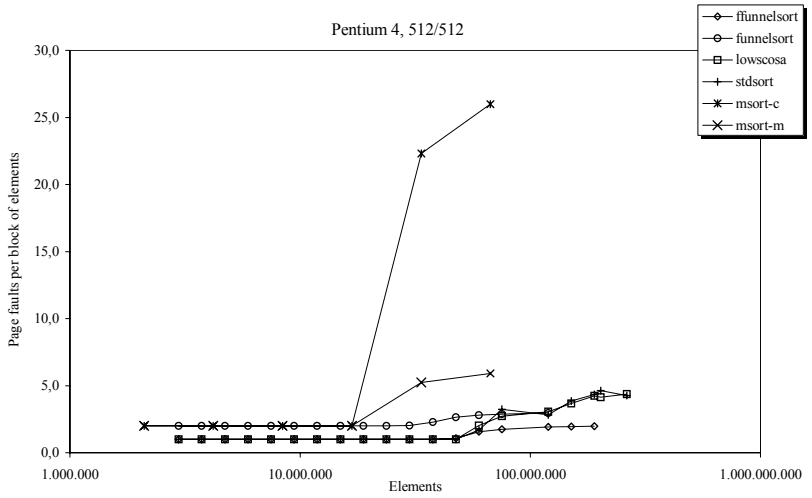

Chart C-64. Page faults sorting uniformly distributed pairs on Pentium 4.

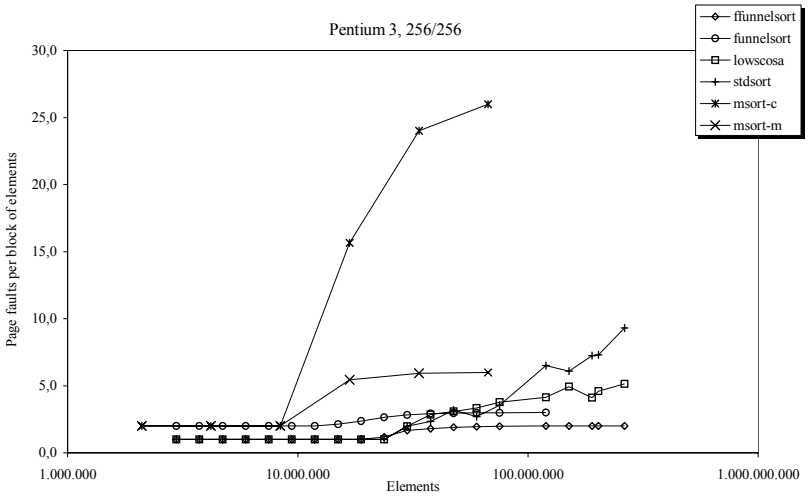

Chart C-65. Page faults sorting uniformly distributed pairs on Pentium 3.

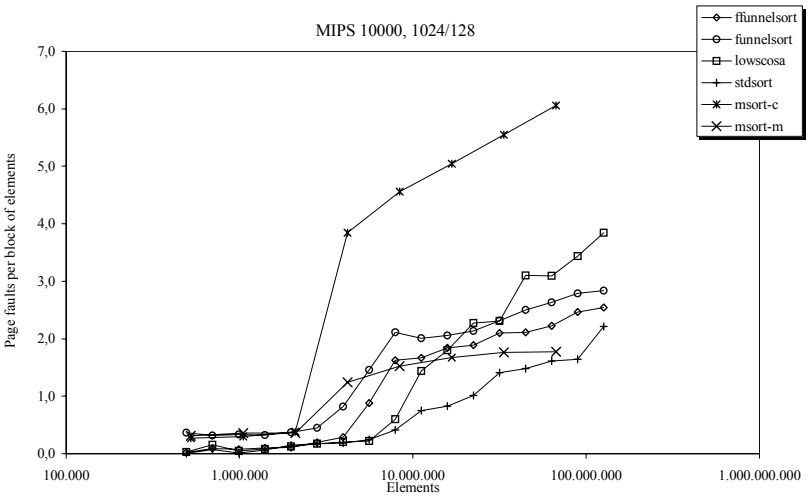

Chart C-66. Page faults sorting uniformly distributed pairs on MIPS 10000.

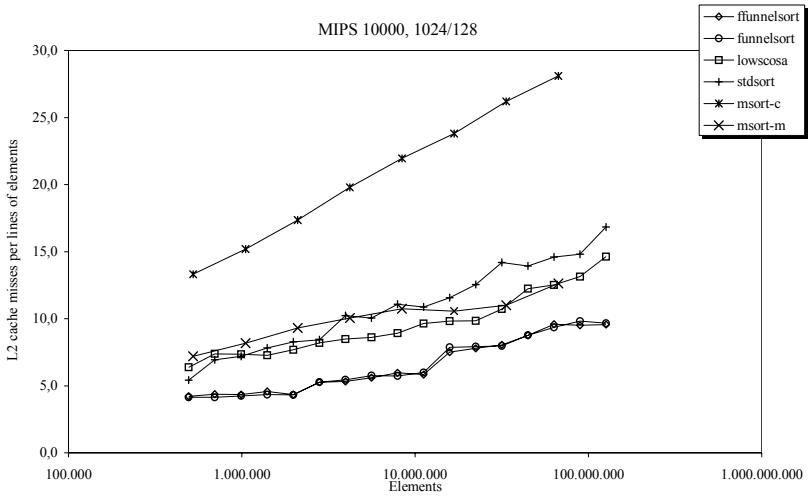

Chart C-67. Cache misses sorting uniformly distributed pairs on MIPS 10000.

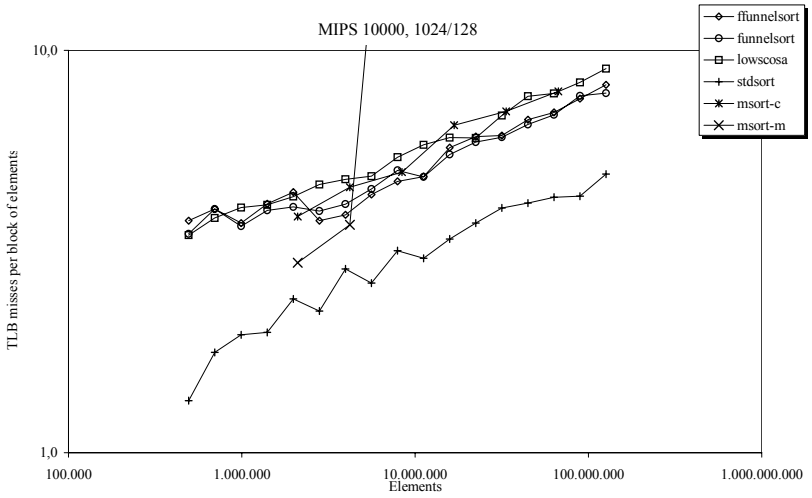

Chart C-68. TLB misses sorting uniformly distributed pairs on MIPS 10000.

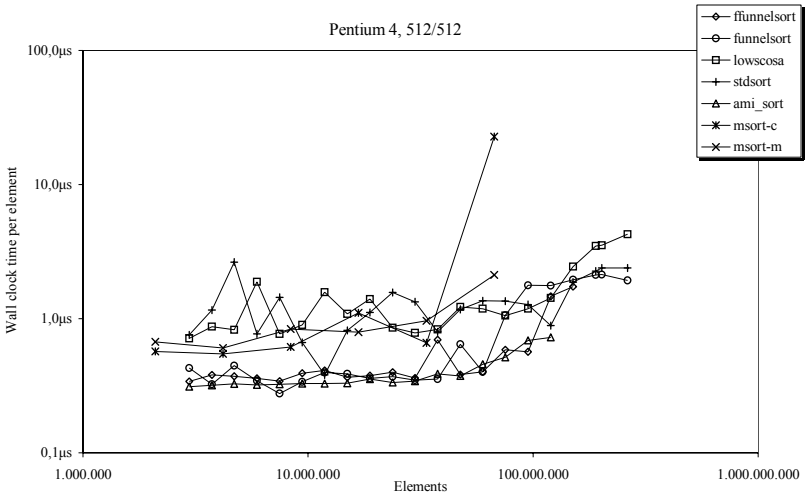

Chart C-69. Wall clock time sorting uniformly distributed integers on Pentium 4.

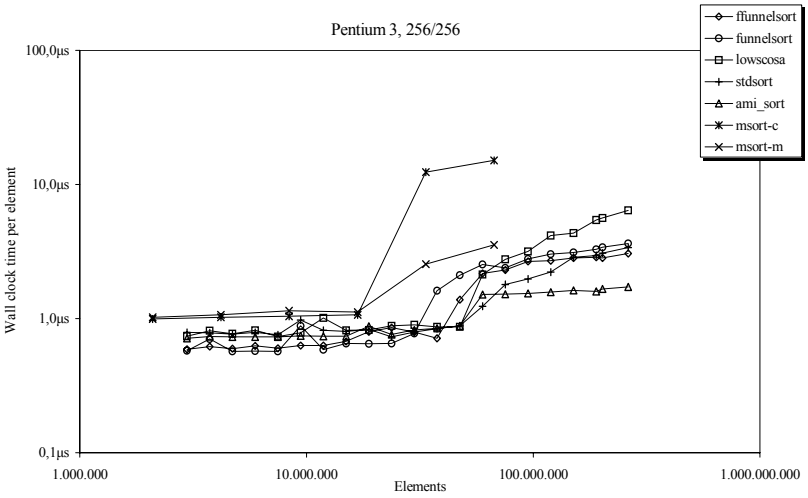

Chart C-70. Wall clock time sorting uniformly distributed integers on Pentium 3.

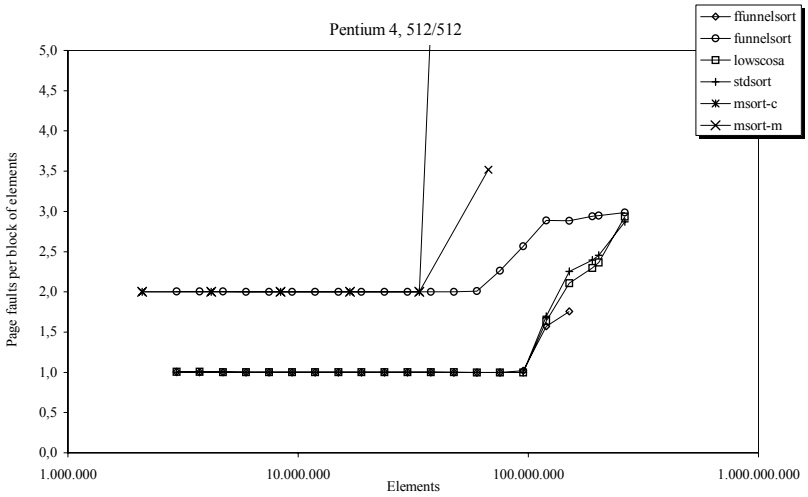

Chart C-71. Page faults sorting uniformly distributed integers on Pentium 4.

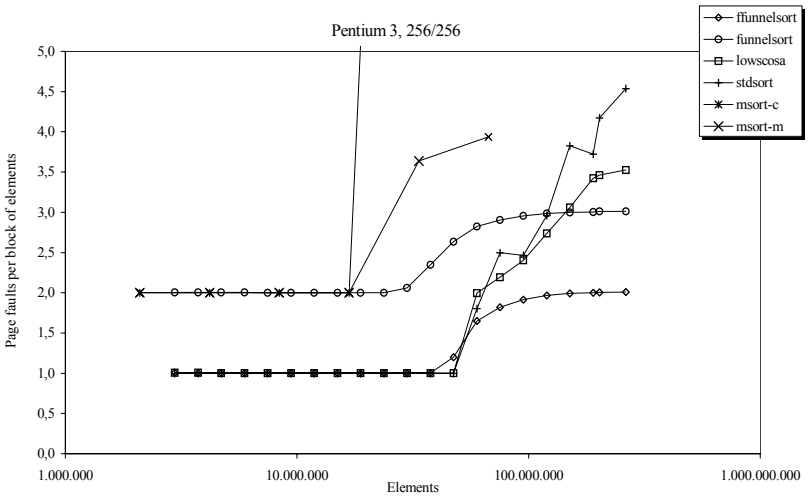

Chart C-72. Page faults sorting uniformly distributed integers on Pentium 3.

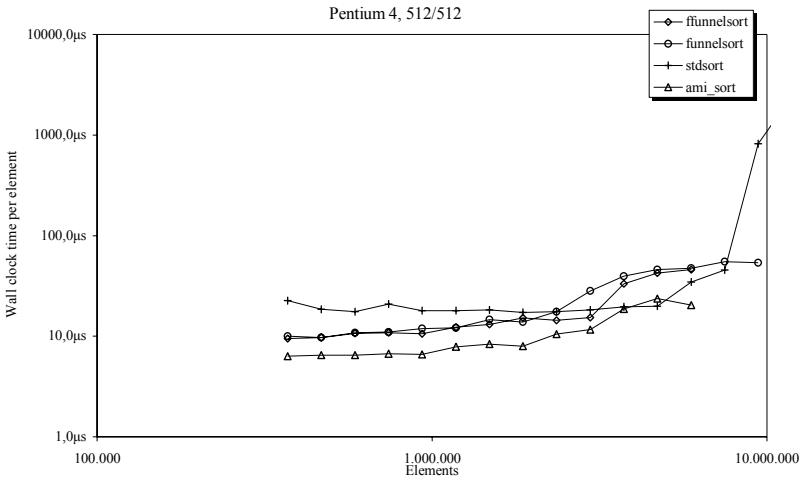

Chart C-73. Wall clock time sorting uniformly distributed records on Pentium 4.

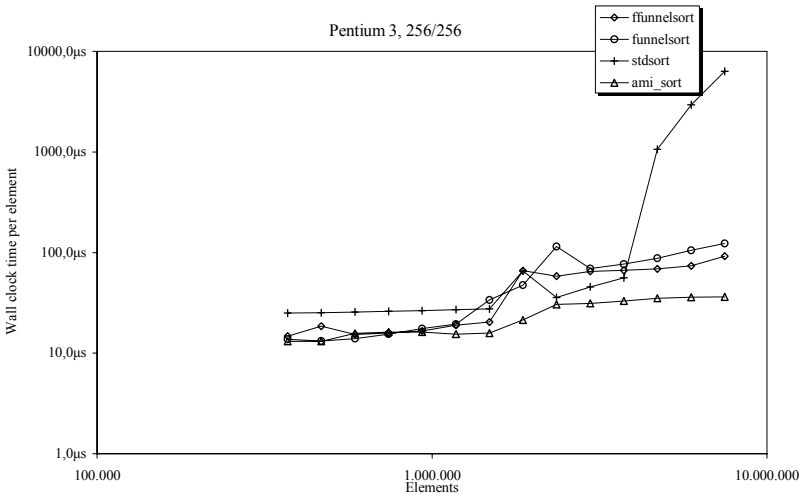

Chart C-74. Wall clock time sorting uniformly distributed records on Pentium 3.

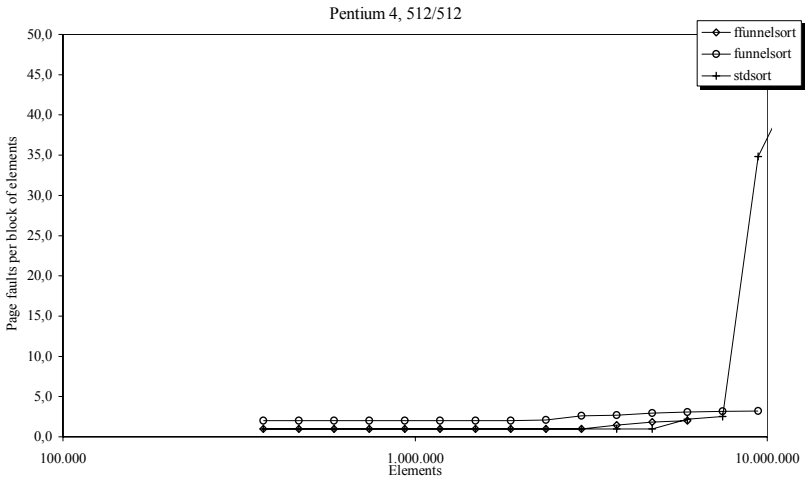

Chart C-75. Page faults sorting uniformly distributed records on Pentium 4.

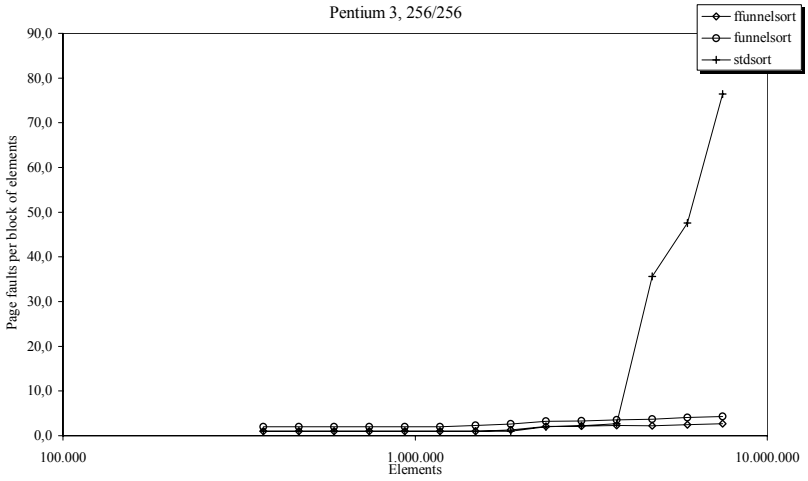

Chart C-76. Page faults sorting uniformly distributed records on Pentium 3.

C.7 Special Cases

The results of the benchmarks described in Sections 6.2 and 6.3.

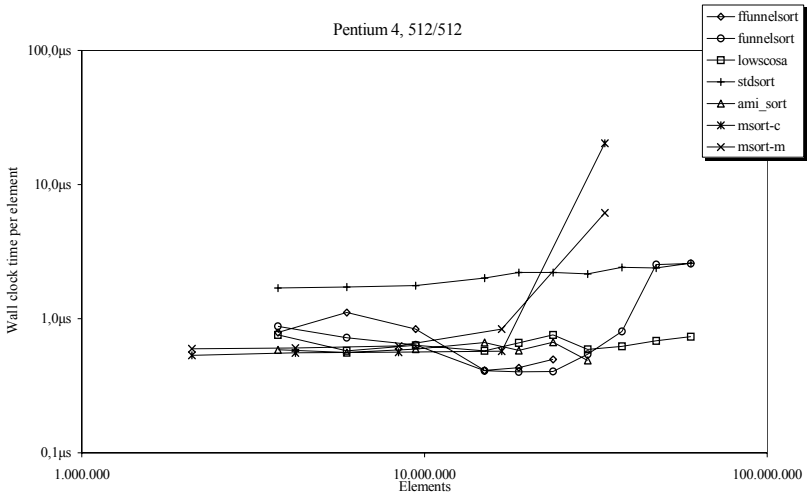

Chart C-77. Wall clock time sorting almost sorted pairs on Pentium 4.

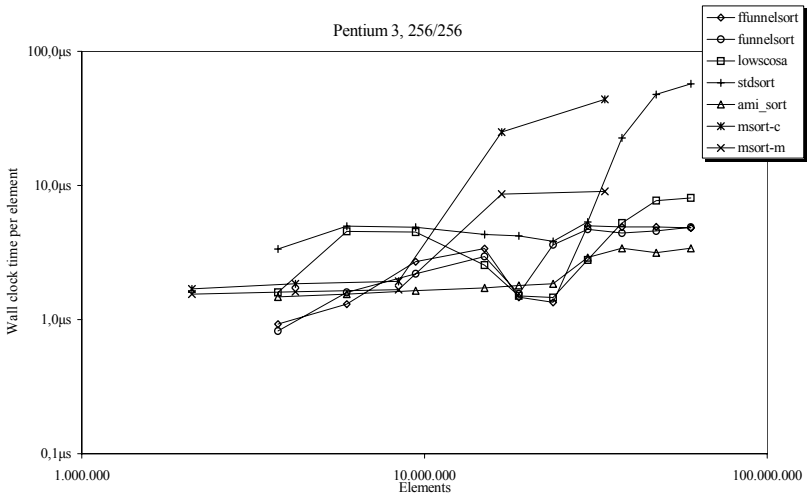

Chart C-78. Wall clock time sorting almost sorted pairs on Pentium 3.

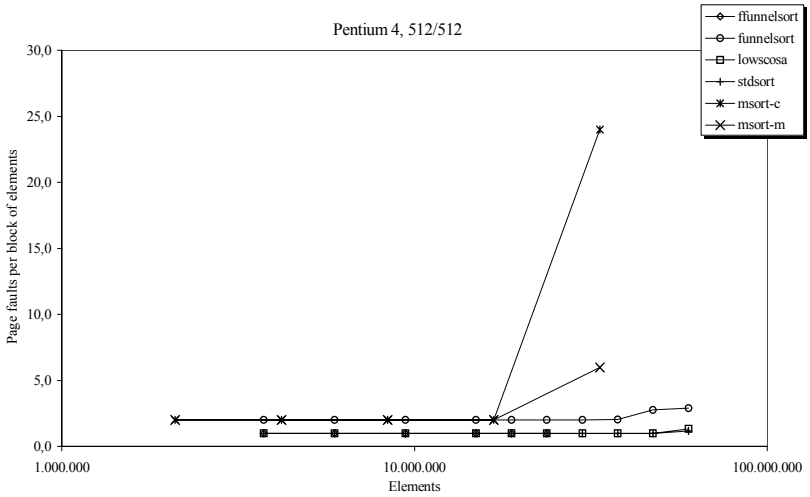

Chart C-79. Page faults sorting almost sorted pairs on Pentium 4.

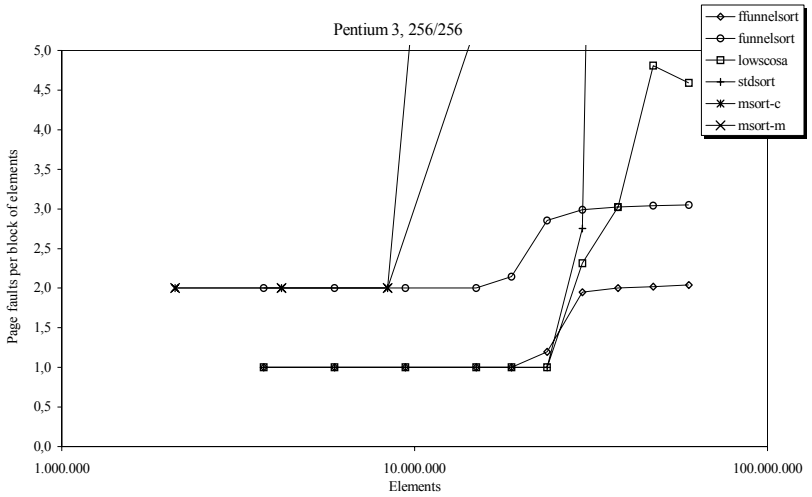

Chart C-80. Page faults sorting almost sorted pairs on Pentium 3.

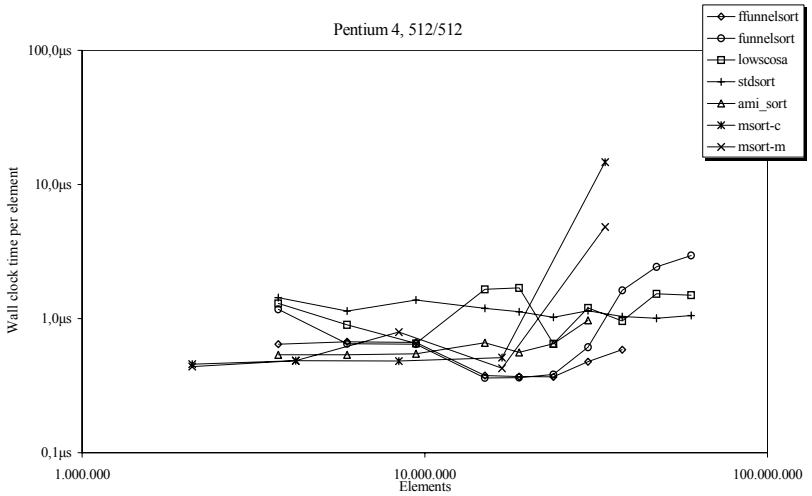

Chart C-81. Wall clock time sorting few distinct pairs on Pentium 4.

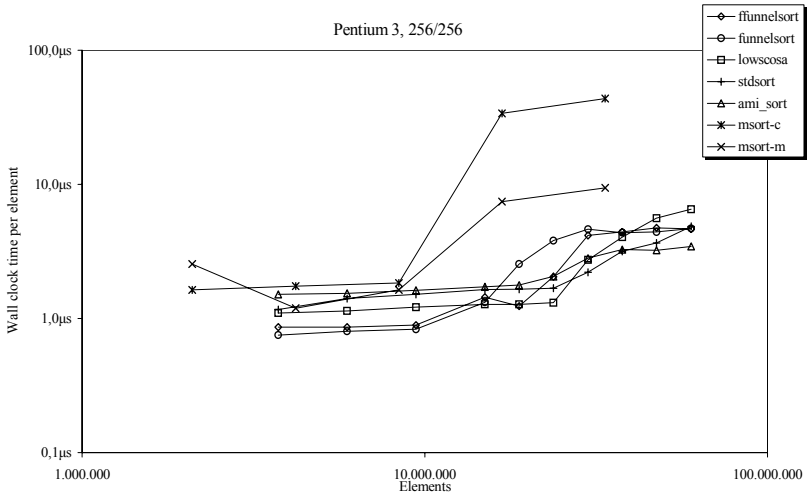

Chart C-82. Wall clock time sorting few distinct pairs on Pentium 3.

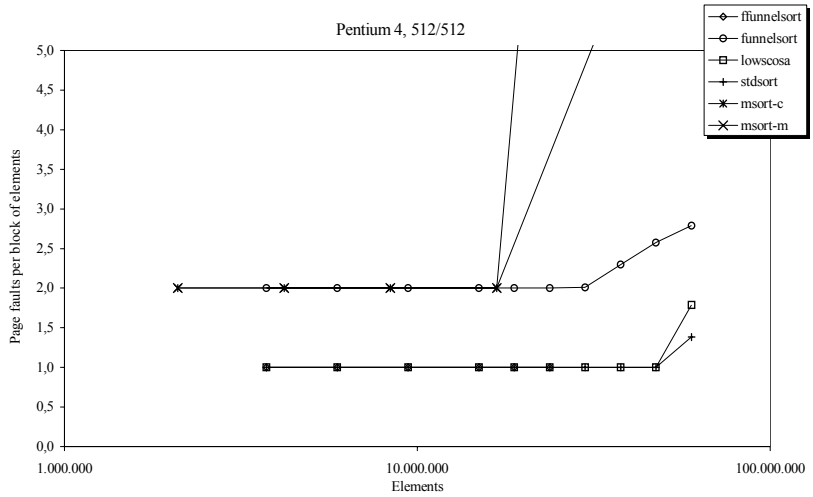

Chart C-83. Page faults sorting few distinct pairs on Pentium 4.

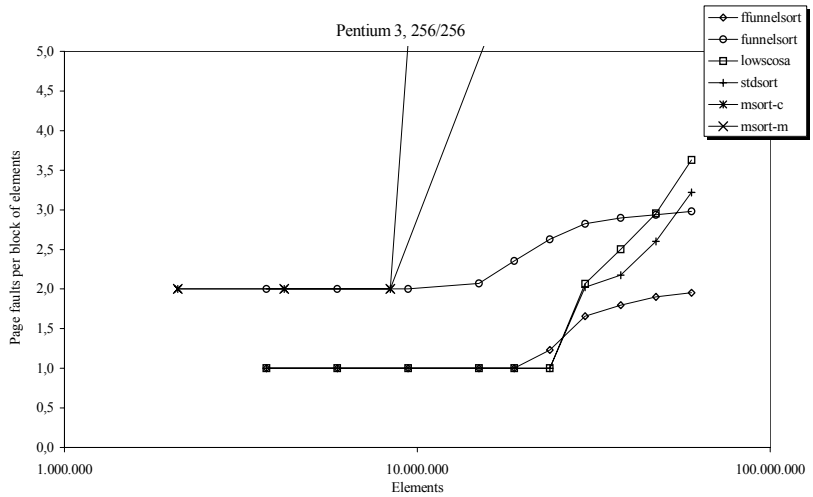

Chart C-84. Page faults sorting few distinct pairs on Pentium 3.