**The Evolution of Improving System Performance**

Antonio Scalfaro

Department of Computer Science and Technology, University of Maryland

CMSC 310: Computer Systems and Architecture

Professor William Dumage

December 11th, 2023

From the onset of the computerized age, there has been a quintessential question for researchers and computer scientists: what improvements can be made to the current machine so that the next iteration of it will be able to process more data at a faster rate? In the dawn of computing, the machines were big, the memory sizes small, and the processing speed painfully slow and costly. As time wore on, advancements began to emerge, easing those bottlenecks. First was virtual memory, allowing for increased multitasking abilities, more efficient memory usage, and extended addressable memory space. Pipelining increased instruction throughput, reduced cycle time, and improved instruction-level parallelism. The 1980’s proved pivotal in the history of computing, giving rise to the RISC architecture, which was well suited for pipelining by simplifying instruction decoding and execution, allowing for faster clock speeds and better utilization of system resources. Finally, and most importantly, cache memory reduced memory access latency, increased memory hit rates, and bridged the gap between processing speeds and main memory speeds.

The purpose of virtual memory was to use the hard drive as an extension of the main memory, allowing for programs to have access to more addressable memory spaces. This was imperative in the early days of computing when main memory sizes were in the MB’s, however, its importance still remains even as main memory has grown to the GB range because the programs being utilized on computers are growing in size as well. In the early days of virtual memory, the page file (the area in the hard drive that the excess main memory is stored in chunks awaiting use) was a smaller size but as memory sizes and program sizes grew, so did the size of the page file and the pages themselves (Null & Lobur, 2019). In time, the process of segmentation (allowing logical portions of a program to be divided into variable-sized partitions) helped to advance virtual memory by decreasing internal fragmentation and supporting sharing and protection, which was difficult using paging (Null & Lobur, 2019). A system does not need to choose between paging and segmentation, rather it can utilize both by dividing the virtual memory into segments of variable length and then dividing the segments into fixed-size pages (Null & Lobur, 2019). Using this technique allows newer systems to get the best of both paging and segmentation.

Pipelining was to computational throughput as the assembly line was to the manufacturing of motor vehicles. Like the assembly line, the pipeline is broken into pipeline stages that allow for different parts of different programs to execute in parallel, increasing overall throughput (Null & Lobur, 2019). Each pipeline stage being responsible for a portion of the fetch-decode-execute cycle. The advent of the superscalar architecture created growth in pipelining by performing multiple operations using parallel pipelines, which helped to identify and exploit instruction-level parallelism (ILP) dynamically (Null & Lobur, 2019). The next advancement came in the form of super-pipelining, which combined the superscalar concepts with the pipelining concepts by breaking the pipeline stages into even smaller pieces. This lessened the load for each stage and thus increased throughput even more (Null & Lobur, 2019). Today, as graphical processing units (GPU) have become more common and more sophisticated, pipelining plays a pivotal role in their use. These GPUs often involve highly parallel architectures that allow for execution units to be optimized for specific tasks.

In the early 1980’s, the RISC architecture was created (though its roots stem back to the 1970s), which represented a huge improvement to the current CISC architecture. With the high cost of memory, early in computing, the idea was to shrink the size of programs by creating complex instruction designs. As memory became more plentiful and cheaper to obtain, the RISC architecture leveraged this additional memory and advanced pipelining techniques (superscalar and super-pipelining) (Null & Lobur, 2019). Programs were being created to be larger in size and consist of simpler more predictable instruction, allowing for shorter clock cycles (Null & Lobur, 2019). Isen et al. note that the early days of RISC architecture provided 2.7 times boost to performance compared to that of the CISC architecture (2009). Since its inception, the RISC architecture has remained the preferred architecture, dominating the market. However, as advancements have been made across the board in hardware and software, the gap between RISC architecture and CISC architecture has narrowed, both having use cases that would benefit system performance over the other (Isen et al., 2009).

Last but not least, cache memory has a case for being the most important component of growth in computing. Cache memory is the glue that has helped to make all the previous advancements work in concert with each other. Cache memory comes in much smaller sizes than main memory, yet much faster, and is temporary. This small but fast memory helps to bridge the speed gap between fast processors and slow main memory to reduce latency in memory, by storing frequently used data instead of making the processor fetch it from main memory every time (Null & Lobur, 2019). Though the first idea may have been to make cache memory larger, this has the inverse effect and was slower (Null & Lobur, 2019). Instead, a multilevel cache hierarchy was implemented, and many of today’s computers feature L1, L2, and even L3 caching(Null & Lobur, 2019). As larger core processors emerge (soon 16 core processors will be commonplace), multi-core processing has become more common which spurred experimentation with multilevel caching (Sibai, 2008). Specifically, whether privatized or shared caching systems work more effectively. Sibai asserts that privatized L2 and L3 cache memory may prove more beneficial in lowering latency in multi-core processors than shared L2 and L3 memory (2008). The trend of cache memory being crucial in systems to reduce fetch latency and allow higher throughput will continue as advancements in processor speeds continue.

The first computers, though complex at the time, proved to be rather simple machines in the annals of history. Today’s machines dwarf their predecessor’s complexity many times over. Virtual memory has expanded from small pages to complex fusions of segmentation and pagination. Pipelining has evolved from the simple pipelining of the 70’s and 80’s to the complex superscalar and super-pipelining necessary for embedded systems like GPUs to utilize resources efficiently. RISC architecture, originally providing a large production increase by utilizing advanced pipelining and larger stores of memory, has since seen its dominance relinquished as CISC architectures are catching up. Finally, the most important component of system architecture has been cache memory, which is the gap-fill that allows processors to continue to get faster and faster while main memory does its best to keep up. Cache memory’s advancement to a multi-level hierarchy has provided ample headroom for the next generation of technological advancements to be made.

**References**

Isen, C., John, L. K., & John, E. (2009). A tale of two processors: Revisiting the RISC-CISC debate. In Computer Performance Evaluation and Benchmarking: SPEC Benchmark Workshop 2009, Austin, TX, USA, January 25, 2009. Proceedings (pp. 57-76). Springer Berlin Heidelberg

Null, L., Lobur, J. (2019). The Essentials of Computer Organization and Architecture (4th ed.). Jones & Bartlett Learning.

Sibai, F., On the performance benefits of sharing and privatizing second and third-level cache memories in homogeneous multi-core architectures, Microprocessors and Microsystems, Volume 32, Issue 7, 2008, Pages 405-412, ISSN 0141-9331, <https://doi.org/10.1016/j.micpro.2008.06.002>. (https://www.sciencedirect.com/science/article/pii/S0141933108000707)