Computer systems and architecture

Science, Computer Science

“ Describe how concepts such as RISC, pipelining, cache memory, and virtual memory have evolved over the past 25 years to improve system performance?”

CPU design, and its integration into computers of varying sizes and purposes is understandably a major focus in the electronics industry. More computations computed means larger infrastructure systems that can be connected together, be it in an online marketing/sales network or a series of sensors and gauges reporting to a core management-processor. The Concepts such as RISC, pipelining, cache and virtual memories all add together to make ever faster, more complex, and denser computer control systems.

The RISC architecture (Reduced Instruction Set Computing) initially evolved due to a growing change in the very manner of computer construction: namely the growing ease of access to cheaper and higher quality memory. RISC operates via the use of a much smaller, but more focused set of instructions than its CISC predecessor, relying on more command executions that execute individually faster than the more verbose, but longer-executing CISC equitable. The fewer hardwired execution sets allowed for a higher number of general purpose registers and cache that allowed for a larger flow through of accessed memory. Demonstrating that markets maintain momentum, the RISC architecture did not gain much momentum outside the university mainstream until federal university research grants funded projects requiring their use. The only mainstream PC to sell on the PC market was the PowerPC developed in conjunction by IBM, Apple, and Motorola in 1991. Outside the PC market however, RISC usage accelerated in microprocessor use in embedded systems and other non-pc related hardware requiring simple executions, but near real-time results. Berkeley developed it’s first version of RISC in 1981 with RISC-II coming a few years later. Sun Systems would eventually create it’s SPARC (Scalable Processor Architecture) architecture in 1987 that continues to sell through this day. SPARC took the idea of minimalism to the extreme, to the point of even removing multiply and divide from the execution list, in order to maximize register access. The Scalable element of SPARC derives from SPARCs ability to be used in anything from a single microprocessor to a multi-thousand processor server hub, all using the same register command set. The key determining element of processor design until recently was the continually increasing speed of processors. This has slowed down due to the concepts of Dennard Scaling, or the fact that increased processors speeds combined with increased transistors mean more heat/power. As a result, parallel processing has become a focus, and architecture designs such as EPIC (Explicitly Parallel Instruction Computing) designed from VLIW by Intel, where processing focuses as much on the speed of execution completion as it does on the ability to complete multiple sets of executions simultaneously. As it stands, EPIC might well render RISC obsolete if not CISC (which is heavily used in PC focused systems) as well.

Just as processors as a whole have begun to focus on parallelism in their usage, so have they also in their design. Pipelining is the term used for the process of taking executions calls and breaking them into simultaneously running “ pipes” that process elements of a call in sequence, allowing as one “ pipe” finishes its task, such as fetching, it can hand off to another pipe to start another fetch. An example of a traditional RISC pipeline consists of a pipe for Instruction Fetch, Decode and register fetch, Execute, memory access, and Register writeback. A pipeline is considered “ super” when its depth is increased to such an extent that step circuitry is simplified and clock rates able to be significantly increased. When a pipeline is capable of executing an instruction fetch every single instruction cycle, then the pipeline is considered fully pipelined. Due to the considerable differences in timing that can exist between one required execution cycle and another, fully pipelined processors are hard to produce. In order to bridge this, dynamic pipelining exists where a processor recognizes stalls, due to either failed branches, memory call hangs, or excessively deep recursion. This allows the processor to re-assign a pipe to another process or thread to keep processor productivity up. The IBM 360/91 was one the first major processing system to heavily utilize pipelining. Despite this added feature, that particular model only sold 20 models, mostly to NASA. Ultimately, pipelining leads into the realm of hyper/multithreading in the growing realm of ILP (Instruction Level Parallelism), or Parallel programing as mentioned above regarding EPIC.

In 1965, Maurice Wilkes created the idea of Slave Memory, or what would eventually be known as Cache Memory. This was memory separate from the main memory storage that held often called instructions and data for quicker processing. As with all developments in the computer world, things have changed greatly since it was first proposed. Now, cache exists most often in three different forms or levels. Level 1 (also called L1) cache is a form of memory located directly on processor chip to very quickly access instructions, holding typically between 8 to 128 KB of memory. L2 Cache, which can exist both on or off the immediate chip typically holds between 512KB and 16MB of memory. It has greater volume but is somewhat slower than L1. Finally, residing entirely off the chip is L3 cache exists entirely separate from the processor on the motherboard, and holds typically up to 8Mb of memory and is slower still than L2 cache, but is used to enhance L1 and L2. Cache is important because it capable of being accessed far more quickly than either RAM or ROM and minimizes pipeline stalls. Cache memory is typically organized to the processor registers with either direct mapping, fully associative mapping, and set associative mapping. Modern processors such as Intel’s recent various lines include all 3 levels of cache directly on the chip. Cache is expensive to produce however, and the primary mover in its advancement is the technology to produce it, namely how small and tightly packed can the memory be produced.

Virtual Memory is an idea that was first brought forth by the Atlas team at the University of Manchester in 1959, which was the idea of taking portions of the main memory (or RAM as it’s now called) and copying it to a physical drive for later access. It was first used on the IBM 360/67 system but would require over the years numerous improvements. Virtual memory works much like RAM, but when a call to a certain memory address previously copied to a physical drive, the memory must reach out to pass through this data from the drive to the processor. As a result this takes, in processing terms, a considerable amount of time and specific hardware and software techniques have been required to improve the process over the years; in truth It remains an imperfect science.

In summary, the path of the computer as we know it has been a hectic, yet short one. It has been only three generations since the very idea of a “ computer” became known. Processors have gone from consisting of transistors and vacuum tubes the size of a fist to holding literally millions in the palm of your hand. It hasn’t been a steady progress of development however, as everything from the development of how these systems think (CISC vs RISC vs EPIC vs ???) to how they remember (RAM vs Cache vs Virtual Memory) has had to grow in their own fashions at their own times. Processing speed has for decades progressed at such an incredible, exponential rate that there is an entire formula to its speed, as well as one regarding it’s decline. Human creativity has been at the heart of discovering each step of advancement, and it remains the same as we go from faster machines, to more machines working in tandem, to simply more clever machines with concepts such as fuzzy logic. Computer technology in all fields moves forwards in leaps and bounds and it remains interesting to see what the next great breakthrough will be in computer development.