Comparison between cisc and risc technologies essay sample

Technology

An understanding of the differences between CISC and RISC technologies would not be complete without a historical viewpoint on the development of computer architectures. While computers today perform all sorts of tasks from sending emails, to playing videos to analyzing weather patterns all computers have at their core a microprocessor which forms its central processing unit or CPU. This CPU is responsible for performing all of the number crunching that computers need to do. The CPU can be thought of as the brain of the computer.

While a computer can be made to do many different things, its CPU can only do a limited set of instructions which is hardwired into its design. The limited and finite set of commands a CPU can do is called itsinstruction set. The instruction set are composed of very basic commands which when combined with one another can perform the complex tasks we need from a computer. As an example, the CPU of a calculator may only have basic arithmetic instructions in its instruction set. For it to be able to compute a sine, the calculator may choose to estimate sine using a Taylor series expansion which is a long computation composed of repeated multiplications and additions. Thus we see that by stringing together the basic instructions, the computer can do very complex tasks. This is the essence of a computer program – a combination of computer instructions which achieve a purpose (Patterson & Hennessy, 2007).

In the early days of computing, writing programs was done by literally stringing together the instructions of the CPU. In assembly language programming, each line of code represents one CPU instruction. This process was long and tedious as programmers strive to simplify the task needed to a

series of basic computer commands. To rectify this, CPU designs tended to move towards having more and more instructions to simplify programming tasks. These new instructions simplified the task of the programmer as the new complex instructions could replace many lines of more basic code.

Apart from ease in programming, these complex instructions were also a boon during the time when memory was significantly expensive. By replacing multiple lines of instructions with a single instruction, programs would be more compact and occupy less precious memory space. Additionally, complex instructions were developed as CPU speed was much faster than memory speed. Because memory access was the bottleneck, computer designers had no issue creating complex instructions which ran for a significant amount of time as the instruction will still finish faster than the memory access. Lastly, these complex instructions were seen and sold as additional features. For marketers of CPUs, more instructions meant better a better processor (Patterson & Ditzel, 1980).

One drawback to complex instructions was complexity. To support these complex instructions, CPUs had to evolve more and more complicated control systems. Also as mentioned, complex instructions tended to finish slower than simpler instructions. This was not unique to complex instructions as the presence of these complex instructions also slowed down the execution of the simpler instructions. CPUs execute instructions every fixed period of their clock cycles. CPUs being measured by their “ speed” in megahertz or gigahertz refer to the number of cycles the CPU executes every second. To accommodate all instructions, the CPU cycle time must be

greater than or equal to the most time consuming task. If the CPU cycle time is faster than some instructions, then these instructions would not finish. Thus, having complex instructions also jeopardized the execution time of the simpler instructions. To make a workaround, some of the more complex instructions were broken down to execute in multiple clock cycles (Patterson & Ditzel, 1980).

Starting in the late 70s and early 80s, a new design paradigm was starting to take shape. In contrast with the previous trend of CPU design, these new CPU designs proposed by researchers in industry and academe alike focused on creating CPUs with reduced instruction sets. These new CPU designs were referred to as Reduced Instruction Set Computers or RISC and the previous generation was henceforth referred to as Complex Instruction Set Computers or CISC.

The main motivation for RISC designs was speed. The concept was that running multiple fast and simple instructions would execute faster than a single complex instruction which does the same task. Another motivation was research into the usage of CPU instructions. It was discovered that on a particular computer, only 10 instructions accounted for more than 80% of all instructions executed by the computer. For the IBM 370 CPU with 183 instructions, 99. 08% of all instructions were performed by only 48 instructions. (Shustek, 1978).

Apart from the small instruction set philosophy, RISC designs have some common traits. Unlike CISC designs which put emphasis on the hardware to perform complex tasks, RISC shifted the focus to software to perform the

needed operation. This task was made easier by the development of high level programming languages which eliminated the need for the programmer to write assembly code. Programmers can write code at a high level and a compiler program will automatically generate a very efficient translation of that program into CPU instructions. Another area is in memory access. CISC CPUs had instructions which allowed them to perform operations on data stored directly in main memory.

RISC CPUs cut down on their instruction counts by removing all of these direct memory instructions. RISC CPUs employed the use of registers – small amounts of memory located directly inside the CPU which allowed for far faster memory access times than main memory. Instead of accessing main memory, RISC CPUs would load the data onto their registers, perform all needed operations then store the register contents back to main memory when the operation was finished. This memory architecture is the reason RISC CPUs are also referred to as “ load-store” CPUs. Additionally, RISC instructions are very uniform in structure which simplifies the decoding. The reduction in simplicity in decoding due to the reduced instruction set and the simplified instructions can be reallocated to putting more memory into the CPU chip (Chen, 2000).

RISC also allowed for faster operations by allowing for pipelining. In pipelining, the CPU executes multiple instructions at the same time. The simplified instructions of RISC allow pipelining because they allow all the instructions to be broken down to distinct stages. All of the functional units within the CPU are used by only one stage at any one time. In pipelining,

even before the first instruction is finished executing, the second and succeeding instructions are already run because they will not have any resource conflict with each other. Pipelining allowed RISC based CPU designs to achieve a throughput approaching one instruction per cycle (Patterson & Hennessy, 2007).

Ultimately, this is what RISC's benefit is – the ability to run more instructions per unit time. Comparison of a CISC design VAX CPU with a RISC design MIPS architecture yielded this picture: the VAX took an average of 5. 4 cycles to execute one instruction while the MIPS took only 1. 1 cycles when the same benchmark program was run on both machines. The VAX can take up to 17 cycles to execute a single instruction while the MIPS only took 3. 1 cycles at the longest (Bhandarkar & Clark, 1991). The success of RISC is also evident in the market. RISC designs form the heart of most CPUs from IBM servers running Power® processors, to Sun workstations running SPARC® CPUs, to Intel Servers carrying the company's Itanium® chips. The only remaining market where CISC designs dominate are the desktop consumer market. RISC designs found it hard to make inroads in replacing the CISC x86 architecture processors used in desktops and laptops because these systems were already very entrenched with millions of users depending on them every day. However, for dedicated applications needing superior performance such as servers and workstations, RISC processor families dominate (" Top500 List – June 2008").

Bibliography

Bhandarkar, D., & Clark, D. “ Performance from architecture: comparing a RISC and a CISC with similar hardware organization.” *ACM SIGARCH Computer Architecture News* , no. 19 (1991) 310-319

Chen, C.. “ RISC vs. CISC.” Stanford University . Available from http://cse. stanford. edu/class/sophomore-college/projects-00/risc/risccisc/ (2000). Internet; accessed 28 November 2008.

Patterson, D. & Ditzel, D. “ The case for the reduced instruction set computer.” *ACM SIGARCH Computer Architecture News* , no. 8 (1980): 25-33

Patterson, David, and Hennesy, J.. *Computer Architecture* . San Francisco: Morgan Kaufman, 2007.

Shustek, L. “ Analysis and Performance of Computer Instruction Sets.” *Stanford Linear Accelerator Center Report* , no. 205 (1978): 56.

Top500 Supercomputer Sites. “ TOP500 List – June 2008 (1-100)”. Available from http://www. top500. org/list/2008/06/100 Internet; accessed 28 November 2008.