**Performance of Modern Computer Systems**

Lecture Notes – Week 5

*G. MacKenzie****,*** *I. Bojanova, J. Avery, Dave Madison*

In this session, we will look at the range of enhancements to the simple computer model presented last week that are making computers more powerful. This week your will learn to apply quantitative principles of computer design and performance measurement:

* Compare performance, cost, and power of computer system configurations.
* Analyze pipeline hazards and explain how pipelining is implemented.
* Analyze performance implications of architectural choices.
* Use memory hierarchy to analyze cache performance and optimizations, and to reduce the effective memory latency.
* Describe the role of cache and virtual memory, and the principles of memory management.
* As well as be able to distinguish different-level parallelisms used to improve performance:
  + - Analyze instruction-level parallelism (ILP): superscalar, VLIW, and EPIC processors.
    - Analyze data-level parallelism (DLP): vector, SIMD, and GPU processors.
    - Analyze thread-level parallelism (TLP): small multiprocessors, multiprocessor interconnect large multiprocessors.
    - Analyze how warehouse-scale computers exploit request-level parallelism (RLP) and DLP.”

**Introduction**

Many of these individual topics are typically the focus of semester long graduate courses, some of them at the Ph.D. level. So it is a bit unrealistic to expect complete mastery of the issues. What we will try to do here is give you pointers to the main issues, and a path forward to go deeper if you wish.

Comparing the cost and performance of computer systems can be a very tricky task. The manufactures all seem to have the fastest, cheapest, and most reliable systems, and, surprisingly, benchmarks to back up their claims. I have seen situations in which, during the time it takes to get a purchase order out of the bureaucracy, a manufacture has gone through two upgrades of their systems. This constant progress makes comparisons difficult.

One way to view this is to look at a web site like Dell’s or HP’s corporate computing pages and try to configure a pair of comparable systems, and then determine if you actually feel confident that you have made the correct choice.

The most difficult problem is deciding exactly what the computer system for which you are responsible is going to do. If you haven’t got that down pat, you can’t begin to configure it.

Take as an example the configuration of a blade multiple server system. A reasonable, provider based tutorial is <http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA1-4286ENW.pdf> which, predictably, HP-centric. Another typical article on choosing the form factor of the system is <http://searchdatacenter.techtarget.com/guides/Understanding-server-form-factors-A-guide-to-rackmount-and-blade-servers>. There are many others.

**Pipelining**

The next topic is pipelining is one of the major advances in computer architecture. The ability of software and hardware to utilize the various scalar and superscalar architectures is a significant problem. The remedies are complex.

The logical evolution of pipeline designs has resulted in two high-performance execution techniques: superpipeline designs and superscalar designs.

         Superpipeline designs*:*

o   Observation: a large number of operations do not require the full clock cycle to complete

o   High performance can be obtained by subdividing the clock  cycle into a number of sub-intervals

  This is equivalent to a higher clock frequency!

o   Subdivide the “macro” pipeline H/W stages into small (and thus faster) sub-stages and clock data through at the  higher clock rate

o   Time to complete individual instructions does not change

  Degree of parallelism goes up

  Perceived speed goes up

         Superscalar designs*:*

o   Implement the CPU such that more than one instruction can be performed (completed) at a time

o   Involves replication of some or all parts of the CPU/ALU

o   Examples:

  Fetch multiple instructions at the same time

  Decode multiple instructions at the same time

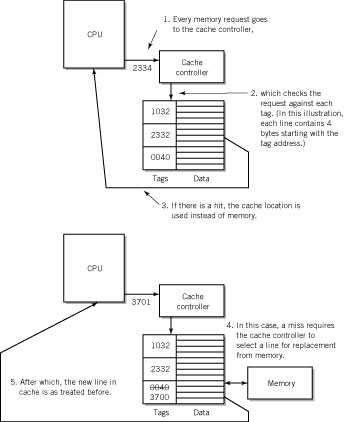
  Perform add and multiple at the same time

  Perform load/stores while performing ALU operation

o   Degree of parallelism and hence the speedup of the machine goes up as more instructions are executed in parallel

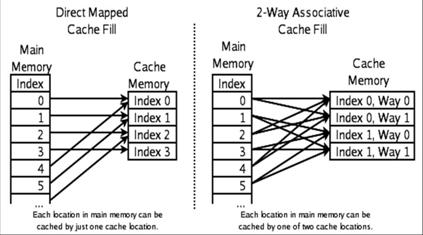
**Memory Enhancement**

Englander (2003) gives a very good overview of major methods employed to enhance memory access rates as a way of improving a computer's speed. You should pay particular attention to the concept of cache memory as illustrated in Figure 1.



***Figure 5.1 - Use of Cache Memory (Englander, 2003)***

Another factor is the concept of **associativity** - the way main memory is mapped to cache memory. If every location in main memory is mapped to a specific cache location, we have direct mapping. If a main memory location can be mapped to two or more cache locations, we have associative mapping. In the latter case the computer requires less cache at a slight decrease in performance. This is illustrated in Figure 2.

 ***Figure 5.2 - Associativity***

**Parallelism Level in Computer Systems**

Let’s focus on the various levels of parallelisms (data, instruction, thread, and request). One interesting place to start is a set of slides from the nvidia.com site: <http://www.nvidia.com/content/cudazone/cudau/courses/ucdavis/lectures/dlp1.pdf>

For request level parallelism (and the difficult calculations that go into analyzing it), see: <http://www.cs.gmu.edu/~menasce/cs465/slides/CAQA5e_ch6.pdf>

This field is quite dynamic. If you google for any of the above topics you will find many auxiliary pages from computer engineering classes from all over the world. The textbooks cannot keep up. As the saying goes, “If you recognize it, it is obsolescent; if you understand it, it is obsolete.”

**Acknowledgement**

The authors are indebted to Dr. David Madison, whose notes formed a basis for much of the material in this lecture.

 References

Englander, Irv (2003). *The Architecture of Computer Hardware, Systems Software & Networking: An Information Technology Approach.* John Wiley and Sons.