CS2410: Computer Architecture (graduate)
University of Pittsburgh
Fall Semester, 2011

Programming Assignment #1 (due October 24)

NOTE: (1) This is an individual assignment. (2) No late submission will be accepted.

Goal

The goal of this programming assignment is to write simple programs to measure the last-level cache and the main memory latencies on a commodity platform.

Background

In order to achieve fast average memory access latency, modern processors incorporate multiple levels of cache memories. Level 1 (L1) caches are typically within the processor core boundary and have a latency of 1 to 4 cycles. Last-level cache (LLC) refers to L2 or L3 cache, depending on the processor architecture. When a memory access does not find the target data in any of the on-chip cache memories, it goes all the way to the (off-chip) main memory. The latency to the memory can be anywhere between 100 to 300 cycles. How can we robustly measure the latency to the LLC and the main memory in software?

Approach

Assume that you have the basic information about the cache parameters. For example, you know the size, block size, and associativity of all caches. The main idea is to write a program having *many memory accesses* in such a way that the memory accesses miss in all caches on the way to the memory hierarchy level that you are measuring the latency of. For example, if you are measuring the latency of the L2 cache, make all your memory accesses miss at L1 and hit at L2. Make sure your memory accesses are serialized so that the latency seen by the memory accesses are all manifested in the execution time; measure the execution time at the cycle level (e.g., using the `rdtsc` instruction on x86) and extract the time spent in memory accesses; and divide the time spent in accessing the memory by so many accesses that contributed.

Note that there are hardware optimizations such as out-of-order multiple-issue processing (i.e., memory accesses may be overlapped) and data prefetching (i.e., determining if a memory access will hit or miss in the cache is harder). You need to write your program to thwart those hardware features.

Programming in Java won't probably help you in this assignment. Programming in C is strongly recommended.

TASK I (50 points). Write a program to measure the latency to the main memory.

• What is the result (average cycles to main memory)? Explain how you calculated the final result. Based on the specification of the memory you have on the platform (e.g., DDR2? DDR3? clock frequency?), relate your result to the hardware specification.

• Explain in detail how you ensured that a memory access is guaranteed to miss in on-chip caches. Explain also how you suppress the impact of superscalar processing and hardware prefetching to robustly measure the memory access latency.

• Does your result (average cycles to main memory) include things other than the pure memory access? What are they? How can you exclude their effects?

• If needed, include plots or block diagrams.

TASK II (50 points). Write a program to measure the latency to the LLC (L2 or L3 depending on the architecture of choice). You may adapt the program you wrote for Task I (or just write a single program to take care of both tasks).

• What is the result (average cycles to the LLC)? Explain how you calculated the final result. Relate your result to the hardware specification.

• Are there other major considerations as you target the LLC instead of the main memory? What are they?

• If needed, include plots or block diagrams.

BONUS (up to 20 points). Choose two other machines that have different cache organizations (total of three). Repeat the main memory latency measurement experiment for the two machines so that you can compare the results for three machines.

• Collect as much information as you can about the specification of the machines. Relate your results to the hardware specification.

• How accurate is your method across different machines? Is your method capable of correctly predicting the memory access latency?

• If needed, include plots or block diagrams.

Submit your report at the class or directly to the mailbox of the instructor (box #276), located in the mail room on 5th floor, Sennott Square. Submit your source codes (readily compilable; include a README for doing that) via e-mail to the instructor.