HY425

Machine Assignment 3

Assignment: 10/5/2008

Due Date: 31/5/2008

Instructions: Note that this assignment is substantially harder than the previous ones and it requires you to invest a significant amount of time and start early! For this assignment you will need to submit a well-written report (in .pdf) with experimental findings from the SimpleScalar simulator. Send your tarball to Stamatis Kavadias (kavadias@ics.forth.gr), with a copy to the instructor (dsn@ics.forth.gr). Use the following subject in your e-mail: **HY425: Machine Assignment 2 Submission.** Please the aforementioned subject only, so that your homework is read and graded. Last, but not least, **start early!**

Assignment

The purpose of this assignment is to experiment with a non-trivial extension to an out-of-order microprocessor, using the SimpleScalar toolset. You are asked to implement a data prefetching mechanism in the out-of-order processor provided by SimpleScalar. The prefetching mechanism should prefetch data from the L2 cache, if a prefetch request finds the data in the L2 cache, or from main memory otherwise. The prefetched data should be stored in a stream buffer. The stream buffer serves processor requests (loads and stores), upon an L1 cache miss, if the data is already in the stream buffer. The data gets copied to the L1 cache from the stream buffer -instead of copying the data from the L2 cache or main memory-, therefore the memory hierarchy answers faster to the processor request. Otherwise, the stream buffer initiates prefetching of cache lines using a fixed stride between cache lines. These lines are prefetched from the L2 cache, if they are present there, otherwise from main memory. The prefetch buffer itself is a FIFO queue, typically with a small number of entries. Note that while the stream buffer is triggering data prefetching requests, the processor may also trigger load/store requests that proceed to be serviced from the L2 cache, or main memory. In other words, the prefetching stream buffer competes with the regular requests for data issued by the processor.

The design and implementation parameters, such as the number of different stream buffers and the number of entries per stream buffer, the mechanism used to specify the stride (or next line to prefetch in general) for the stream buffers etc., will be left to your choosing. This is the part of the assignment where there is no prescribed solution and you are tested on your intuition and creativity! You will need to think as architects and make judicious choices considering the trade-off between performance and complexity.

Preparation

For this exercise you are asked to work and think as researchers. Your first step is to read two seminal papers from the Annual International Symposium on Computer Architecture. The first:

- N. Jouppi. Improving Direct Mapped Cache Performance by the Addition of a Small Fully Associative Cache and Prefetch Buffers. *Proc. of the 17th Annual International Symposium on Computer Architecture.*

introduces stream buffers. The second:

- S. Palacharla and R. E. Kessler. Evaluating Stream Buffers as a Secondary Cache Replacement. *Proc. of the 21st Annual International Symposium on Computer Architecture*.

introduces several ideas for optimizing the design of stream buffers. These two papers will provide you guidelines for your implementation of stream buffers. Note however, that the papers are "dated" and concern older generations of microprocessors. You will instead simulate a more modern superscalar microprocessor with two levels of cache, out-of-order execution, non-blocking caches, etc., as provided by the SimpleScalar Toolkit. The following third paper

- K. Farkas, P. Chow, N. Jouppi and Z. Vranesic. Memory-System Design Considerations for Dynamically-Scheduled Processors. *Proc. of the 24th Annual International Symposium on Computer Architecture.*

provides a deeper investigation of prefetching with stream buffers for superscalar processors and you should consider it as a more accurate guidelines for implementation.

Assignment

Select the best L1-L2-TLB system configuration using the results from Machine Assignment 2. Note that you need to use the latencies and cache sizes from this exercise. Introduce a streaming buffer extension to the out-of-order processor of SimpleScalar, which shall implement data prefetching. The stream buffers should be looked up in parallel with the L1 cache by the microprocessor in one cycle. If the processor misses in the L1 cache but hits in the stream buffer, then the data is transferred from the stream buffer to the L1 cache in one cycle (second cycle of the memory access in total), and in parallel the data gets written to the registers for which there is an outstanding request to the block fetched from the stream buffer. In other words, an L1 cache hit takes 1 cycle to service, and a stream buffer hit takes 2 cycles to service. If both the L1 cache and the stream buffer miss, then both issue requests to the L2 cache. Obviously, processor requests should have some kind of priority over prefetching requests from the stream buffers. Design prefetching policies and mechanisms, including selecting a stride for prefetching the next cache line in each stream buffer, size and number of stream buffers, triggering mechanism for prefetching data into stream buffers, and techniques to prevent stream buffers from using up too much L2 cache bandwidth and too much memory bandwidth, are up to you to design and implement. The aforementioned papers provide valuable guidelines for this exercise. Use the benchmarks from the course web page (http://www.csd.uoc.gr/~hy425/homework/benchmarks.tar.gz) to evaluate your prefetching mechanisms and policies.

In your evaluation you should answer the following questions:

i) How many times does a processor request that misses in the L1 cache is a hit on the stream buffer and what is the hit rate of the stream buffer)?

ii) How many times does a processor request that misses in the L1 cache is also a miss on the stream buffer and what is the miss rate of the stream buffer?

iii) What are the reasons for the misses in the stream buffer? More specifically, possible reasons for miss buffer misses can be the inaccuracy of the prefetching policy (e.g. the buffer uses a stride which prefetches data not requested by the processor), or, the timeliness of prefetching (e.g. the buffer fetches data too late to be used by the program). How many misses are attributed to each reason?

iv) What is the additional bandwidth (expressed in bytes requested by the L2 cache per cycle and bytes requested by main memory per cycle), that the prefetching mechanism requires?

v) What is the performance improvement (or non-improvement) from prefetching in IPC? What applications benefit the most from prefetching and what applications do not benefit, or lose performance due to prefetching?

Your report should discuss details of your implementation of stream buffers in SimpleScalar in a clear and comprehensive manner. Furthermore, you should discuss results using tables or charts. A short interview with the instructor will be scheduled to discuss your implementation and results for this assignment. Note that for this assignment, we are more interested for a correct implementation of stream buffers, rather than for impressive results. Negative results are acceptable, as soon as they are well explained and justified by the design that you have implemented.