ECE4074 Assignment 3

Due: Friday, Week 12.

Aim:

Adapt your cache simulator to calculate the expected processing time for matrix multiplication on a shared memory multi-processor system.

Method:

This is an individual assignment. Using your 4 way set associative, 4 words per block, 32kB and 8kB versions of your cache simulator; adapt your 50x50 matrix multiplication algorithm to run on an N processor shared memory system. Step N from 1 to 50. Show and explain the speed increases gained by using increasing numbers of processors in your multiprocessor system.

Use the same DRAM parameters and assumptions as assignment 2.

You may assume that memory arbitration takes 0 clock cycles.

As this is generally a highly parallel algorithm, you may ignore the cache coherency problem.

There is no L2 cache.

Hints:

While, ultimately how you achieve this is up to you, there is a template solution available on moodle.

The template requires you first to split your program up among your processors such that each processor independently calculates elements of the final matrix. You should provide an assembler program that can achieve this splitting. You may assume at the start of the program that the processor number is available to you within a register.

Second, the template intends to run your normal cache simulator for each processor and record the times **between** each access to DRAM (everything but the RAS/CAS latencies) for each processor and the addresses required (and also record the time between your last DRAM access and the termination of your program). Using these recorded times and addresses as input, the template provide a way to run your multiprocessor system. The solution can calculate when each processor needs access to the shared DRAM from these results. In the event that multiple processors need access to the DRAM simultaneously, an arbitration method is need. The template solution is designed to work on a first come, first served basis with lower numbered processors gaining priority. When access is granted, the processor given access is delayed by the amount of DRAM delay calculated (remember to keep track of DRAM rows) and prevent access to the DRAM until at least this access is complete. A simple worked example of how access times are calculated is shown over the page:

Input Data:

P1 time	P1 address	P2 time	P2 address	P3 time	P3 address
20	0	20	50	20	100
25	1	25	51	25	101
5	NA	5	NA	5	NA

At time 0ns:

All 3 processors start their program.

At time 20ns:

All 3 processors require access to the memory, arbitrator grants access to processor 1. DRAM access time is calculated as 32.3ns. Processor 1 may resume at time 52.3ns (20ns+32.3ns).

At time 52.3ns:

Processor 1 resumes. Processor 2 is now given access to memory. DRAM access time is calculated as 32.3ns. Processor 2 may resume at 84.6ns (52.3ns+32.3ns).

At time 77.3ns (52.3ns+25ns):

Processor 1 needs access to the DRAM. It must wait until at least 84.6ns (The next time the DRAM is possibly available).

At time 84.6ns:

Processor 2 resumes processing. Processor 1 and 3 are waiting to access DRAM. Processor 3 is granted access by the arbitrator. DRAM access time is calculated as 32.3ns. Processor 3 may resume at 116.9ns (84.6ns+32.3ns).

At time 109.6ns:

Processor 2 needs access to the DRAM. It must wait until at least 116.9ns (The next time the DRAM is possibly available).

At time 116.9ns:

Processor 3 resumes processing. Processor 1 and 2 are waiting to access DRAM. Processor 1 is granted access by the arbitrator. DRAM access time is calculated as 32.3ns. Processor 1 may resume at 149.2ns (116.9ns+32.3ns).

At time 141.9ns:

Processor 3 needs access to the DRAM. It must wait until at least 149.2ns 9ns (The next time the DRAM is possibly available).

At time 149.2ns:

Processor 1 resumes processing. Processor 2 and 3 are waiting to access DRAM. Processor 2 is granted access by the arbitrator. DRAM access time is calculated as 32.3ns. Processor 2 may resume at 181.5ns (149.2ns+32.3ns).

At time 154.2ns (149.2ns+5ns):

Processor 1 finishes its program.

At time 181.5ns:

Processor 2 resumes processing. Processor 3 is waiting to access DRAM. Processor 3 is granted access by the arbitrator. DRAM access time is calculated as 32.3ns. Processor 3 may resume at 213.8ns (181.5ns+32.3ns).

At time 186.5 (181.5ns+5ns):

Processor 2 finished its program.

At time 213.8ns:

Processor 3 resumes processing.

At time 218.8ns (213.8ns+5ns):

Processor 3 finishes its program. All processors have finished, program is complete.

Note:

These the times given as input data are the complete time taken by the processor in between DRAM accesses. The example times are just examples, they are not real results.

When accessing a line words for your cache, you may assume that your processor gets exclusive access to the DRAM while the entire line is fetched. (Your cache results file would indicate 0 ns between successive addresses on the same processor, but the arbitrator would not give access to the other processors)

There are only 2 cache configurations that need to be tested.

Your report should explain the features of your results. Start by plotting relative speed increases (this is expected to be non-linear) and go from there. Keep a close eye out for features that don't make immediate intuitive sense. Hit and miss data from your caches may also be useful for explaining your results as well as the total number of times all processors needed to access DRAM. Research into other potential limits of multiprocessing system (other than Amdahl's Law) may be useful. You should also suggest what architectural changes would need to be made to increase the multiprocessor gains.