ExamHeaderSEECS7.5in.eps

**CEG 4136 Computer Architecture III: FINAL EXAM**

|  |  |
| --- | --- |
| **Date:** December 15th | **Professor:** Dr. M. Bolic |
| **Duration:** 3 hours | **Session:** Fall 2015-2016 |
| **Total Points =** 100 out of 106 |  |

**Note:** Closed book exam. Calculators are not allowed.

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Student ID:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **Question** | **Maximum** | **Score** |
| **1** | 36 |  |
| **2** | 12 |  |
| **3** | 18 |  |
| **4** | 20 |  |
| **5** | 20 |  |
|  |  |  |
| **Total** | 106 |  |

1. Short questions (4 points each – total 36)
2. Why is it difficult to further increase the number of pipelining stages in the processors?
3. What is the difference between SIMD and SPMD?
4. List several features that good benchmark programs need to have.
5. Where is the mutex stored in general purpose processing systems:

a. In memory

b. In special hardware module next to each processor

1. Simultaneous multithreading (SMT) increases the processor area approximately by
2. 100%
3. 50%
4. 5%
5. Using X-Y routing, show how the message is routed between the following source destination pairs:
6. (1,1) and (3,2)
7. (2,3) and (3,1)



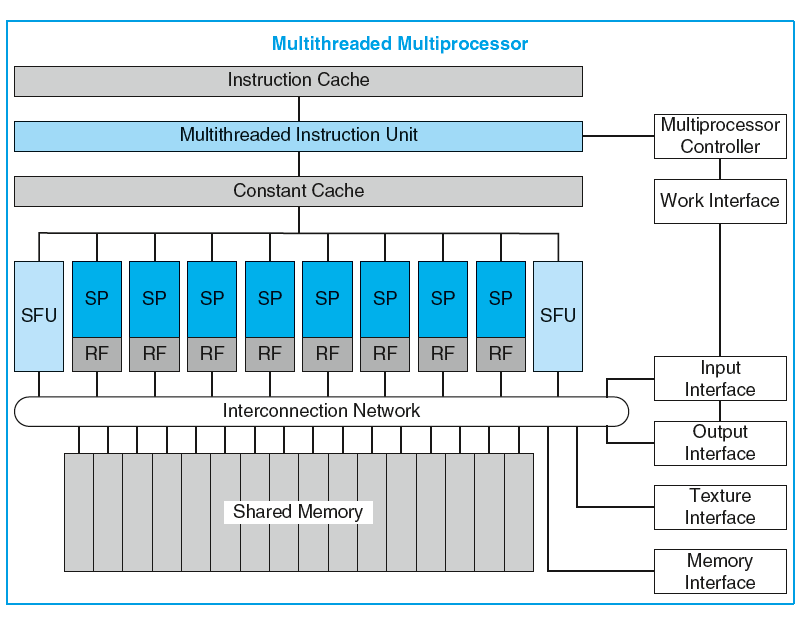
1. What is the diameter, degree and bisection width of 4x6 torus network with bidirectional traffic?
2. The main problem with a crossbar switch is:
   1. We cannot connect all the processors to all the destinations all the time.
   2. The propagation delay in the network.
   3. It is expensive when we have lots of processors.
3. Assume that **Full Map Directory cache coherence protocol with Centralized Directory Invalidate is implemented.** Assume that directory for address X contained all 0s at the beginning. Assume that there are 8 processors and that directory has dirty bit and presence bits. Fill the following table for the following sequence of instructions:

|  |  |  |
| --- | --- | --- |
| Time instant | Operation | Content of the directory for X |
| 1 | Processor 0 – read X |  |
| 2 | Processor 5 – read X |  |
| 3 | Processor 0 – writes to X |  |

1. (4 points each, total 12)

Multithreaded multiprocessor with eight scalar processor cores that makes one unit of the GPU is shown in the figure below. The eight SP cores each have a large multithreaded register file (RF) and share an instruction cache, multithreaded instruction issue unit, constant cache, two special function units (SFUs), interconnection network, and a multibank shared memory.

1. Why is this called single instruction multiple thread architecture and how different that is from SIMD?
2. How is the group of parallel threads executed on this architecture? How is the group of parallel threads called in OpenCl and/or in CUDA terminology?
3. How large is the register file in comparison with CPU register file? How large is the main GPU memory in general?



1. (6 points each – total 18) We would like to perform mathematical operations on matrices using a single processing system with a cache. The cache line is 16 bytes (4 words). The matrix is stored in the memory row by row (eg. A11, A12, ..., A1N, A21, A22, ..., A2N, ... AN1, AN2, ...,ANN) . Each element of the matrix is an integer (4 bytes=1 word).
2. Which of the two programs below would run faster and why?
3. Estimate the total number of cache misses if N is larger than the size of the cache.

|  |  |
| --- | --- |
| S=0;  For (i=1:N)  For (j=1:N)  S=S+Aij  End  End | S=0;  For (j=1:N)  For (i=1:N)  S=S+Aij  End  End |

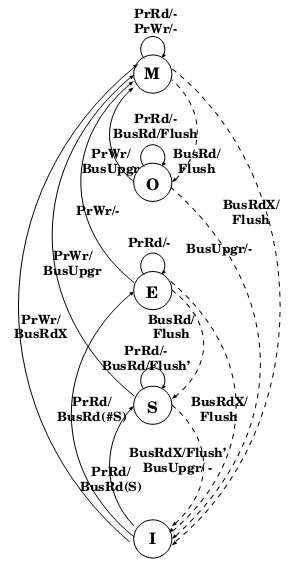
1. The program on the left is implemented on a shared memory system with 4 processors. Each processor is handling addition of N/4 numbers where N/4 is larger than the size of each cache. The partial sums computed on each processor need to be summed in the end. Rewrite this program so that it can run on multiple processors. Add synchronization primitives (Lock, Unlock, Barrier,…) if needed.
2. (20 points) Consider a shared memory system with two processors A and B. Each processor has a cache that is two way set-associative and has total size of 4096 bytes with block size of 16 bytes (4 words). The cache is initialized with all zeros.

The following sequence is executed

1. Processor A writes a word 44 to the address 0x4B0
2. Processor B reads a word from the address 0x4B8
3. Processor A writes a word 22 to the address 0x4B0
4. Processor B writes a word 11 to the address 0x4B8
5. Processor A reads a word from the address 0x4B8

Give the state of the cache and the contents of the caches and the memory (*x*) after each step, if MOESI write-back invalidation protocol is used.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | State of A’s cache | Content of x in A’s cache | State of B’s cache | Content of x in B’s cache | *Content of memory locations 0x4B0-0x4BF* |
| 1. Processor A writes a word 44 to the address 0x4B0 |  |  |  |  |  |
| 1. Processor B reads a word from the address 0x4B8 |  |  |  |  |  |
| 1. Processor A writes a word 22 to the address 0x4B0 |  |  |  |  |  |
| 1. Processor B writes a word 11 to the address 0x4B8 |  |  |  |  |  |
| 1. Processor A reads a word from the address 0x4B8 |  |  |  |  |  |



1. (20 points)

Consider a future 16-way CMP operating in a power-constrained environment. The CMP can automatically configure itself to run as a 1-, 2-, 4-, 8-, 16-way core CMP but always using a fixed power budget. For instance, it can run as a single-core processor by grabbing the power from the other 15 cores by putting them to sleep and using the additional power to increase its frequency. Assume that sleep and wakeup times are zero, and that power and frequency have a square relationship. For instance, if one core uses the power of all 16 cores, its frequency can increase four-fold. We call this an EPI-throttled CMP.

Consider a partially parallel application. This application starts as a single-threaded application and spends 5% of the time in sequential mode. During the following 60% of the time, the application has 16 threads, and only four threads for the next following 20% of the execution time. During the remaining execution time, the application has only one thread.

1. What is the speedup of this application when it runs on this future CMP compared to running on a single-core machine that uses the same power but operates at a higher frequency using the square relationship?
2. What is the speed of this application when it runs on this future CMP compared to running on a traditional 16-way CMP (again using the same power budget) that does not provide reconfiguration capability?
3. What is DFVS? What are the other techniques used in heterogeneous processors to controlling power and frequency when the load is high or low?