Jared Raiola
CS 4290: Advanced Computer Organization
Homework 3 - 100 points

Question 1 - Multicore 15 points

A. List out the categories of Flynn's Taxonomy and give an example architecture for each category.

a. SISD: Single I, Single D Stream, traditional uniprocessor
    i. Most conventional computers have this architecture.
b. SIMD: Single I, Multiple D Stream, can execute the same instructions but works on different data streams
    i. Cray's vector processing machines
c. MISD: Multiple I, Single D Stream, systems operate on the same data stream and must agree on the result
    i. Space shuttle flight control computer
d. MIMD: Multiple I, Multiple D Streams, multiple instructions performed on multiple data sets
    i. IBM's Symmetric Multi-Processing System

B. Compare and contrast the two types of memory models a multiprocessor system could implement.

a. Message-Passing:
    i. Pros:
        1. A processor can directly address local memory only, meaning there is all communication must be explicit
        2. Each computer will have its own processor and memory
        3. Simpler and cheaper hardware to make
        4. Explicit communication can be used to directly tell what memory operations are the most costly
    ii. Cons:
        1. Explicit communication is difficult to program
        2. Optimization must be done by hand
b. Distributed Shared Memory:
    i. Pros:
        1. Communication is automatically
        2. Programs are easier to write and optimize
        3. No need to manually distribute data
    ii. Cons:
        1. More hardware support necessary
        2. Programs are easy to write correctly, but may not be efficient

C. **What is the difference between process and threads?**
   a. Process:
      i. A program running on a machine
   b. Thread:
      i. A context within a program
      ii. Shares a virtual address space with other threads
      iii. Used for parallelism in programs

Question 2 - SMT 25 points

A. **Detail the pros and cons of Simultaneous Multi-Threading?**
   a. Pros:
      i. Instructions from different threads allows for less stall time, when one thread is stalled, another goes.
      ii. It fills idle “issue slots” with work from other threads and throughput improves
   b. Cons:
      i. More hardware
         1. For N-way (N threads) we need N sets of registers, N RATs, and N virtual memory spaces
      ii. More complex logic, need to maintain interrupts, exceptions, faults on a per-thread basis
      iii. Can cause performance degradation by doing multiple tasks at the same time

B. **Describe the changes that need to be made to add SMT support to an Out-of-Order superscalar architecture. Make sure to specifically address what units need to be replicated, as well as changes to existing units.**
   a. For an N-way SMT
      i. N registers
      ii. N RATs
      iii. N virtual memory spaces

C. **Describe the cases where one-thread may stall the execution of another thread when performing SMT execution.**
   a. One thread may stall the execution of another thread when both threads are attempting to access/manipulate the same portion of memory. Both threads cannot write to the same portion of memory at the same time otherwise data will be lost.

Question 3 - Cache Coherence 30 points

A. Given a machine with the following details:

- **64-bit machine with a byte-addressable 64GB physical memory space that uses 4KB pages**
- **An 32KB 4-way set-associative write-back, write-allocate cache**
    - **Cache block is 64B**
    - **Cache uses true LRU replacement policy**
    - **Cache is virtually indexed and virtually tagged**
    - **Cache implements MOSI protocol.**

How big (in bits) is the tag store?

a. Tag Store = Number of Lines * (Tag Size + LRU Size + bits for states)
b. Tag Store = (512) * (128 + 2 + 4) = 68608 bits

B. Given a machine with the following details:

- **64-bit machine with a byte-addressable 64GB physical memory space that uses 4KB pages**
- **An 32KB 4-way set-associative write-back, write-allocate cache**
    - **Cache block is 64B**
    - **Cache uses true LRU replacement policy**
    - **Cache is virtually indexed and virtually tagged**
    - **Cache implements MOESI protocol.**

How big (in bits) is the tag store?

a. Tag Store = Number of Lines * (Tag Size + LRU Size + bits for states)
b. Tag Store = (512) * (128 + 2 + 5) = 69120 bits

C. Given a two processor system (P0 and P1), with each processor paired with 2-entry fully associative private cache. The caches are kept coherent by using the MOSI protocol with bus intervention. There are 2 possible addresses in our program: A and B.

If the following stream of accesses is seen:
P0 Read A, P1 Read A, P0 Read A, P1 Write A, P0 Read A, P0 Write A

Then, assuming that the caches are initially empty, what is the miss rate of P0?

a. 33% miss rate

D. Given the same details from Part C, say that the hit time of the local cache is 4 cycles, accessing data from a neighbor cache is 10 cycles, and accessing data from memory is 100 cycles, then how long will it take for P0 to complete its memory accesses?

a. 100 for first read because it's not in the cache + 4 for the second read because it's in the local cache + 10 for the third read because it was written to it's neighbor + 4 for the write because it's in local cache = 100 + 4 + 10 + 4 = 118 cycles

E. Given a four processor system (P0, P1, P2, and P3), where each processor has a 4-entry fully associative private cache that uses the MESI protocol. Initially, P0'S private cache contains the following state:

	A	B	C	D
State	**E**	**S**	**I**	**M**

If the following stream of requests is seen, how will the cache state change after each request?

P1 GetS A, P2 GetS A, P2 GetM B, P3 GetS D, P1 GetM A, P3 GetM D

a. A -> S
b. A -> S
c. B -> I
d. D -> S
e. A -> I
f. D -> I

F. What are mechanisms to reduce the metadata overhead for a directory-based coherence system? Make sure to mention what protocol changes may be need to be made.

a. In order to reduce metadata overhead for a directory-based coherence system, processors can be grouped into nodes. This allows for broadcasts to happen to a singular node, which then broadcasts to all that processors in the node at once. It also allows for a snoopy protocol to happen within specific nodes. We can also limit our pointer scheme and store the I node-ids and only broadcast when there is an overflow.

Question 4 - Consistency 15 points

A. What is the difference between coherence and consistency?

a. Coherence is defined as insurance that writes to a particular location will ALWAYS be seen in order.
b. Consistency insures that writes to different locations will be seen in the order that makes the most sense, allowing for reordering.

B. **Given Processor A that implements a weak consistency model, how would we modify a program that requires sequential consistency to run on Processor A?**
    a. Sequential consistency requires all accesses for each processor to be kept in order. If we implemented a weak consistency model, we could keep synchronization accesses the same, but all writes in the system must fully complete before accesses occur and no data accesses are allowed AT ALL until all previous synchronization accesses have been completed.

C. **Given Processor B that implements a release consistency model, how would we modify a program that requires sequential consistency to run on Processor B?**
    a. We would not need to modify a program that requires sequential consistency because a release consistency model upholds sequentially consistent acquire and release fences.

Question 5 - Synchronization 10 points

A. **To implement synchronization operations such as mutex locks or barriers, what considerations should be made in terms of cache coherence to gain the best performance.**
    a. Must consider whether the thread is modifying the data or just reading the data. If both threads are simply trying to read the data from a shared variable, it can be allowed because they are not modifying that memory location.
    b. Casualty of writes must also be taken into consideration. Since all writes will be seen in the same order by all processors, a request queue for the memory must be made, otherwise a new thread could essentially “cut the line” and grab the memory before a process that has been waiting longer, essentially starving it.

B. **If test-and-set is sufficient to create a mutex lock, why is test-and-test-and-set the preferred mechanism?**
    a. It is the most simple mechanism, requires less programming to implement, and will return whether the mutex is busy or not in the fastest time possible.