EECS 570 Midterm Exam
Winter 2015

Name: ________________________________ unique name: ______________

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

_________________________________

Scores:

<table>
<thead>
<tr>
<th>#</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/ 28</td>
</tr>
<tr>
<td>2</td>
<td>/ 12</td>
</tr>
<tr>
<td>3</td>
<td>/ 26</td>
</tr>
<tr>
<td>4</td>
<td>/ 14</td>
</tr>
<tr>
<td>5</td>
<td>/ 20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>/ <strong>100</strong></td>
</tr>
</tbody>
</table>

NOTES:

- Closed book, closed notes.
- Calculators are allowed, but no PDAs, Portables, Cell phones, etc.
- Don’t spend too much time on any one problem.
- You have about 90 minutes for the exam.
- There are 8 pages in the exam (including this one). Please ensure you have all pages.
- **Be sure to show work and explain what you’ve done when asked to do so.**
1) **Short Answer [28 points]**

   a) Give an example reason why a parallel program might not achieve linear speedup in a multicore system. (Linear = speedup of p with p processors) [4 points]

   b) Of online transaction processing (OLTP) and decision support system (DSS) database workloads, which is more likely to benefit from hardware mechanisms that accelerate synchronization operations? [4 points]

   c) State the conditions under which a transaction must be aborted/retried in a hardware transactional memory system. [4 points]

   d) State one advantage and one disadvantage of hardware transactional memory as compared to software transactional memory. [8 points]

   e) Give an **advantage** and a **disadvantage** of maintaining inclusion in multi-level cache hierarchies in a multiprocessor system. [8 points]
The following C++11 program snippets have undefined behavior.

```cpp
// Initialization code (runs before either thread)
X* x = null;
bool flag = false;

// Code running on producer thread
x = new X();
flag = true;

// Code running on consumer thread
while (!flag);
x->f++;
```

a) Briefly explain why the behavior of the program is undefined. [4 points]

b) On some modern processors, sometimes the code may run as intended, but sometimes it may misexecute. Explain a scenario for how it might execute incorrectly (i.e., what might hardware do that causes the program to fail). [4 points]

c) How can the program be fixed? [4 points]
3) **Synchronization [26 points]**

a) Explain a scenario where spin based synchronization will be better than blocking based synchronization. [4 points]

b) Explain a scenario where blocking-based synchronization is better than spin-based synchronization. [4 points]

c) Explain the advantage of a list-based lock over a ticket lock. [4 points]
c) Under what circumstances might a test-and-test-and-set lock be preferable to a ticket or array/list based lock? [4 points]

d) Explain why the following barrier implementation is incorrect. Describe briefly how it can be fixed (Note: it is not necessary to write corrected pseudo-code provided your explanation of how to fix the implementation is clear.) [10 points]

1: global (shared) count : integer = P  //P is # of CPUs
2: procedure central_barrier
3:   if fetch_and_decrement(&count) == 1
4:     count := P
5:   else
6:     repeat until count == P
4) **Cache Coherence [14 points]**

a) Complete the following coherence state transition diagram for an MOSI coherence protocol on a symmetric multiprocessor using the action / reaction symbols below. (Do not introduce additional states - you do not need to worry about transient states for this question). Two example transitions have been filled in for you.

- PrRd - Processor Read
- PrWr - Processor Write
- PrEvict - Processor Evict (due to cache replacement)
- BusRd - Bus Read
- BusRdX - Bus read exclusive (i.e., read-and-invalidate)
- BusInv - Bus invalidate
- BusDataReply - Bus cache-to-cache data transfer
- BusWB - Bus Writeback

PrRd / BusRd

PrRd

O

M
5) Selective Cache Coherence [20 points]

Many researchers have observed that maintaining cache coherence for all cache blocks is expensive and may be unnecessary. Cache coherence incurs large overheads in terms of storage, time, and energy for both load and store operations. Design a scheme that maintains cache coherence for only a subset of cache blocks, while still maintaining correctness for arbitrary shared memory programs.

a) Identify a property of a cache block that obviates the need to maintain coherence.

b) Once you have identified such a property, consider the cost to track it. Should it be tracked at cache block granularity? Can it be tracked at some other granularity? What advantage is there to tracking over a different granularity?

c) Based on the property and granularity you have identified, briefly describe a hardware mechanism for identifying and tracking blocks that do not require coherence. How are loads and stores for such blocks handled? Draw a diagram of any new storage/tracking hardware structures, operating system modifications, or additions to the cache tags or coherence directory required by your scheme.

d) Suppose a program violates the property you identified in part (a) (e.g., because a memory location changes behavior). What actions will your system take to ensure correct execution?
(additional space for question #5)