EECS 570 Midterm Exam - SOLUTIONS
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 (avg. x minutes per problem).
- There are x 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]

Overhead due to synchronization operations; load imbalance among threads; sequential portion of the algorithm creating an Amdahl bottleneck, etc.

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]

OLTP. Ranaganathan’s results show that a considerable fraction of OLTP execution time is spent on synchronization.

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

Two transactions access the same memory location and at least one of the accesses is a write.

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

Advantage: HTM can achieve higher performance.

Disadvantage: difficult to support unbounded transaction sizes in hardware.

e) Give an advantage and a disadvantage of maintaining inclusion in multi-level cache hierarchies in a multiprocessor system. [8 points]

Inclusion allows snoop results to be reported by examining only the lowest-level cache.

However, it may impose restrictions on the blocks that may be simultaneously cached in the L1, and reduces the effective aggregate cache capacity (e.g., relative to exclusive caches).
2) Parallel Programming [12 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]

The code contains an unannotated data race on the "flag" variable. The behavior of racy programs is undefined in C++11.

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]

Some OoO processors might complete the flag=true assignment before the x = new X() assignment. Then, the x-> dereference could dereference a null pointer.

c) How can the program be fixed? [4 points]

Declare the flag variable "atomic".
3) **Synchronization [26 points]**

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

   It is better to spin if the expected wait time for the lock is less than double the context switch time.

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

   It is better to block if expected wait time for the lock is long and there are other runnable threads available in the system.

c) Explain the advantage of a list-based lock over a ticket lock. [4 points]

   In a list based lock, each contending acquirer spins on a separate cache block, so that the lock can be handed off using point-to-point messages instead of an invalidation storm.
c) Under what circumstances might a test-and-test-and-set lock be preferable to a ticket or array/list based lock? [4 points]

When a lock is rarely/never contended, the test-and-test-and-set lock minimizes the overhead of lock acquisitions and releases.

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
```

Consider the situation where one or more threads are waiting at the barrier (on line 6) and the last arriving thread executes lines 3 and 4, resetting count to P and proceeding past the barrier. That thread might continue executing to the next use of the same barrier, where it decrements P before all threads have proceeded past line 6. These threads are then "stuck" at the previous barrier, and the system is deadlocked (count != P and will never be decremented to 1 since some threads cannot proceed out of the first barrier).

The barrier can be fixed by having waiting threads spin on a separate boolean variable, "global_sense". Each thread maintains a local variable "local_sense" that is toggled upon its arrival at the barrier. It then spin until "global_sense" matches "local_sense". The last-arriving thread (detected via count) toggles global_sense. The complete implementation appears in Lecture 7 Slide 33.
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
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?

Private blocks do not require coherence. Private blocks can be tracked at page granularity. Extend the OS page table to indicate a page is private by recording the CPU that access it. Don't bother contacting directory for blocks on private pages. If you access a page that is private to a different CPU, force a cache flush. More elegant designs are possible.
(additional space for question #5)