IS2202 - Computer Systems Architecture
Multiprocesors and Memory Ordering
Bonus assignment 2

April 7, 2014

Instructions

- The assignment should be solved *individually*.
- Solutions should be properly *motivated*, a few short sentences is usually enough.
- Answer the question in your own words. Copying answers from other sourses such as text books is not acceptable. You may qote other sources. In thtat case, make sure to include the source and say how you interpret the quoted text.
- Solutions should be given in English (as we have non-Swedish speaking TA:s).
- Solutions need to be handed in as PDF document *before the deadline* at the appropriate place on the course web (https://www.kth.se/social/course/IS2202).
- You need to core at least 10 points to qualify for a bonus point at the exam.
- Deadline April 17, 2014

1 Cache Coherence

1.1 False-sharing, 3 points

1. What is a *false-sharing miss*?
2. How can *false-sharing* be avoided? Give *two* examples, one using only software techniques and one where hardware is changed.

1.2 MSI-coherence, 2 points

A simple snooping based cache coherence protocol is described on page 214 (4th edition) or page 360 (5th edition) in *Computer Architecture A Quantitative Approach*. The protocol uses three states, *Exclusive*, *Shared* and *Invalid*. The *Exclusive* state in the book is normally called *Modified*, so we'll call it that! Hence, the protocol will be called MSI.

A cache line enters the *Modified* state whenever a write occurs on that cache line. If the cache line was in the *Invalid* state, data have to be fetched and all other caches have to invalidate their copies. If the cache line was in the *Shared* state, no data have to be fetched, but all other caches still have to invalidate their copies. We call the transition from *Shared* to *Modified* an *upgrade*.

Table 1: MSI Transaction table

step	CPU0	CPU1	CPU2	CPU3	on bus	data source
0	wr 0				RTW	MEM
1			rd 0			
2				wr 0		
3			wr 0			
4		wr 0				
5			rd 0			
6			wr 0			

3. Table 1 contains a set of serialized transaction such that step n happens before $n + 1$. Each memory access in the table references the same cache line. Fill in the *on bus* column with the bus request caused by the each access. Use *RTS* for *read to share*, *RTW* for *read to write*, *INV* for *invalidate* or none if no bus activity is required. Also, indicate the source of data.

1.3 MOSI-coherence, 1 point

The MOSI protocol, as discussed in on-line lecture, is similar to the MSI protocol, but contains an additional *Owner* state.

4. Describe what the *MOSI* protocol tries to optimize compared to the *MSI* protocol.

2 Memory consistency, 7 points

The code below will be used in the following questions. Prior to executing the code below, `a=1` and `flag=0`.

Listing 1: CPU0's code

```
a = 2;
flag = 1;
```

Listing 2: CPU1's code

```
while {flag != 1)
    ;                          // Wait until flag is 1
cout << a;                     // Print a
```

5. What value will CPU1 print when executed on. . .

   (a) . . . a *sequentially consistent* machine?

   (b) . . . a machine implementing *Total Store Order*?

   (c) . . . a machine implementing *Release Consistency*?

   Note: Only one of the answers 1, 2 or *timing dependent* is correct for each sub problem. Do not forget to motivate your answer.

6. Give at least one reason why a multiprocessor machine should implement *sequential consistency* instead of *release consistency*.
7. Give at least one reason why a multiprocessor machine should implement *total store order* instead of *sequential consistency*.
8. Why would a computer architect chose to implement any of the weaker memory models?
9. Does a weaker memory order affect the correctness of correctly synchronized applications? Assume that the application uses pthreads. Motivate.