Advanced Computer Architecture
Bonus Assignment 2 — Multiprocessors and Memory Ordering

Instructions

- The assignment should be solved *individually*.
- Solutions should be properly *motivated*, a few short sentences is usually enough.
- Answer the questions in *your* own words. Copying answers from other sources, such as the text book, is not acceptable. You may quote other sources, in that case, make sure to include the source and say how you interpret the quoted text.
- Solutions should be given in *English* or *Swedish*. English is preferred.
- Solutions should be handed in *before* the deadline by:
  - Posting them in *Andreas Sembrant's* mailbox (house 1, floor 2).
  - Sending an email to andreas.sembrant@it.uu.se. The *answers should be in the message body*, solutions attached as Word documents, or similar, will be ignored. If you need to include graphics use one of the following formats: PNG, JPEG, SVG, PDF, EPS, PS
- Fill out and include this cover page when you hand in your solution. If you are submitting by email, make sure to include the same information in the email.
- Unreadable solutions will be treated as incorrect.
- You need to score at least 10 points to get a bonus point on the exam.

Deadline

Solutions handed in after the deadline specified on the course homepage will be marked on a best effort basis and will not result in any bonus on the exam.

Student

Name

Civic registration number (personnummer)

Email address

Date

1 Cache Coherence

1.1 False-sharing

1. What is a *false-sharing miss*? (1)

2. How can *false-sharing* be avoided? Give *two* examples, one using only software techniques and one where the hardware is changed. (2)

1.2 MSI-coherence

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*.

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			

Table 1: MSI transaction table

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. (2)

1.3 MOSI-coherence

The *MOSI* protocol, as discussed in class, 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. (1)

2 Memory Consistency

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 */
printf("%i\n", a);
```

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

   a. ...a *sequentially consistent* machine? (1)

   b. ...a machine implementing *Total Store Order*? (1)

   c. ...a machine implementing *Release Consistency*? (1)

   Note: Only the answer *1*, *2* or *timing dependent* is correct.

6. Give at least one reason why a multiprocessor machine should implement *sequential consistency* instead of *release consistency*. (1)

7. Give at least one reason why a multiprocessor machine should implement *total store order* instead of *sequential consistency*. (1)

8. Why would a computer architect chose to implement any of the weaker memory models? (1)

9. Does a weaker memory order affect the correctness of correctly synchronized applications? Assume that the application uses pthreads. Motivate. (1)