CSC 258/458 Written Assignment

Due Monday, February 24, 2014

1. (10 points) We say that write-after-read and write-after-write dependencies are *not* true dependencies. Why? Provide sufficient details to justify your answer.

2. (10 points) A cache coherence protocol may invalidate a cached data item at processor $\mathcal{X}$ that is being modified by another processor $\mathcal{Y}$. Alternatively, the cache coherence protocol can update the cached copy while keeping it valid. What is the advantage of updating over invalidation? What is the disadvantage of updating?

3. (10 points) Consider the following execution of two parallel programs on a shared-memory multiprocessor.

```
                /* Initially flag1 = flag2 = 0 */

/* P1 */                            /* P2 */
flag1 = 1;                          flag2 = 1;
turn = 2;                           turn = 1;
while (flag2 && turn==2) ;          while (flag1 && turn==1) ;

/* ... critical section */          /* ... critical section */

flag1 = 0;                          flag2 = 0;
```

   Assume that the processor and compiler does not reorder any memory operations. However, the multiprocessor has processor-local caches **without** cache coherence. Does the above program guarantee mutually exclusive executions of the critical sections (e.g., the processes do not enter their respective critical sections at the same time)? Explain your answer.

4. (10 points) Consider the following execution of three parallel programs on a shared-memory multiprocessor.

```
                        /* Initially A = B = 0 */

/* P1 */                /* P2 */                                /* P3 */
A = 1;                  while (A == 0) ; /* wait */             if (B == 1)
                        B = 1;                                      output(A);
```

   If the multiprocessor supports sequential memory consistency, what are possible outputs for P3?

5. The atomic instruction `test_and_set` assigns a value to a location and returns the old value of the location. Consider the lock/unlock routines below that protect a critical section (acquire lock before entering; release lock after leaving).

```
acquire_lock(L *location) {
    while (test_and_set(location, locked) == locked) {
        while (*location == locked) ;
    }
}

release_lock (L *location) {
    *location = unlocked;
}
```

(a) (10 points) If the multiprocessor supports sequential memory consistency, show that the above lock/unlock routines ensure mutual exclusion of protected critical sections.

(b) (10 points) Dr. Foobar says that the second while loop in `acquire_lock` is unnecessary for correct synchronization. Is Dr. Foobar right? Explain your answer.

(c) (10 points) Dr. Foobar also says that the second while loop in `acquire_lock` may help improve the synchronization performance. Is Dr. Foobar right? Explain your answer.

6. (10 points) One of the earliest "read-modify-write" atomic instructions is `compare_and_swap`. The instruction takes three operands: the location to be modified, a value that the location is expected to contain, and a new value to be placed there if (and only if) the expected value is found. The instruction returns an indication of whether it succeeded. Show that `compare_and_swap` can be used to emulate `test_and_set` which atomically assigns a value to a location and returns the old value of the location.