CSC 258/458 Written Assignment

Due Sunday, February 20, 2011

1. We say that write-after-read and write-after-write dependences are *not* true dependences. Why? Provide sufficient details to justify your answer.

2. Consider multiprocessors with shared bus and write-through caches. Under the bus snooping-based coherence protocol, each processor monitors the bus for writes. If there is a local cached copy of the write target location, the local copy is invalidated.

   (a) Show that the bus snooping protocol satisfies cache coherence. Provide sufficient details to complete your proof.

   (b) The above cache coherence protocol invalidates 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. 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, is it possible for P3 to output 0? If so, please explain how? Otherwise, please prove that it is impossible.

4. 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 used for mutual exclusion.

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

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

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

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

(c) 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.

5. 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` is *universal*, in the sense that it can be used to emulate any other read-modify-write instruction that atomically reads a memory location, computes a new value as a function of the old one, and writes the new value back.