CSC 258/CSC 458
Written Assignment 1
February 10th, 2020
Due in class on Monday, February 17th, 2020

Please be sure to show all your work and write down any assumptions you make.

1. Load linked/store conditional is a pair of instructions that can be used for synchronization. Load-link returns the current value of a memory location, while keeping track of the address. A subsequent store-conditional to the same memory location will store the new value supplied only if no updates have occurred since the address was load-linked (determined by the hardware keeping track of activity on the address).

   (a) Show that `ll/sc` 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. E.g., how can it be used to emulate compare-and-swap?

   (b) `ll/sc` is said to avoid compare-and-swap's ABA problem (the problem that a value in a variable switches from A to B and then back to A, with the change going undetected by compare-and-swap). Why might this be a problem to begin with and why does ll/sc not suffer from it? Are there other possible problems with implementations of `ll/sc`?

2. Consider a snoopy bus-based protocol similar to the MESI protocol we reviewed in class. Suppose you were designing a coherence protocol that favored a migratory access pattern, in the sense that it was designed to minimize bus activity for migratory data. Data is said to exhibit a migratory access pattern when it bounces around, or migrates, from one processor to another, with each processor writing (and usually reading) the data one or more times, before it is accessed by the next (i.e., at any given time, the data is being read and written by exactly one processor). Draw the finite state machine for such a protocol, assuming that you were concerned only with the stable states. While you do not have to (but are encouraged to) optimize for multiple sharing patterns, you should make sure not to increase traffic for the common cases of private (accessed only by a single process/processor) read-only or read-write data and allowing caching of read-shared data in multiple caches.