

Homework 3 (not graded)

Consider a critical section with two locks (`lock1` and `lock2`) which can be acquired in any order and do not have to be both acquired. Design a spin-lock (i.e., show the locking code using a `test_and_set` instruction) that satisfies the following requirements:

- Deadlock is prevented (i.e., it is never the case that a holder of `lock1` is waiting for `lock2` while a holder of `lock2` is waiting for `lock1`).
- No thread ever spins while holding a lock, as this will prevent another thread that needs only that lock from doing useful work.
- The lock has good performance in terms of latency, delay, and bandwidth consumption on an SMP machine with snooping caches.

You can assume that the contention for locks is low (a well designed concurrent program typically will not have high contention)—that is, there are very rarely more than 2 threads spinning.

Justify with a couple of sentences why your lock works well.

Disclaimer: This HW is not license to kill: don't use spin-locks in projects or exams. Spin-locks are architecture-specific—for all of your regular locking needs you should use the locking primitives provided by a thread library (like `pthread_mutex_lock` in Pthreads implementations).