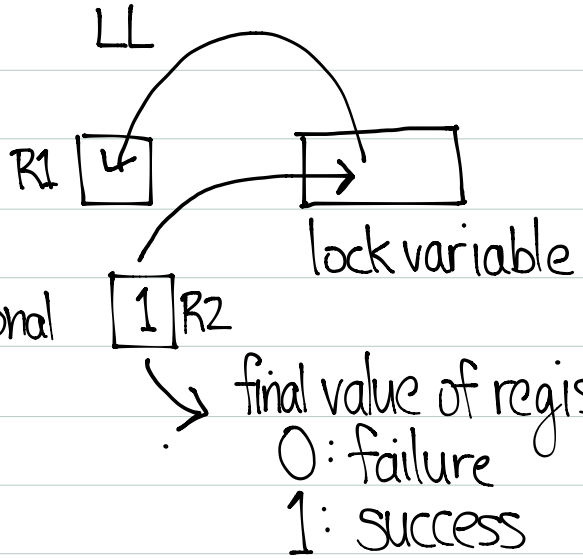
- No tested set for mips
- use
  - ↳ LL → load link
  - ↳ SC → store conditional



Locks

- spinlocks vs locks
  - ↳ use spinlocks to implement locks
  - ↳ seem very similar
    - when to use either
      - ↳ spinlocks
        - small, quick, critical sections
      - ↳ locks
        - larger more complicated
        - other thread won't just stay spinning

Semaphore

- can solve mutual exclusion
- P and V are atomic

Semaphore P implementation WRONG

```
P();
```

```
        spinlock_acquire(&sem->sem_lock);
while   if(sem->sem_count==0) {
            /* block */
            wchan_lock(sem->sem_wchan);
            spinlock_release(&sem->sem_lock);
            wchan_sleep(sem->sem_wchan);
            spinlock_acquire(&sem->sem_lock);
        }
        sem->sem_count--;
        spinlock_release(&sem->sem_lock);
```

cs350_submit -
cs350_runtests |& tee runtests.out
↳takes 15 mins on unloaded servers

How LL SC works.

- SC R M

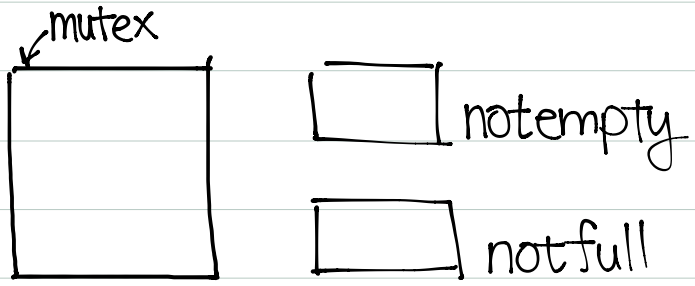
① WRITE contents of R into M if no other update to M since LL
otherwise, do not change M

② Set value of R to 0 or 1 to indicate success or failure of the store

Condition Variables

- kind of like wait channels

- signal - wakeone
- broadcast — wakeall
- need to be used with a lock
  ↳ can only be used within the critical section that is producted by a lock.



- if a thread gets waken on notempty/notfull they may not have priority in the thread queue in the current simple implementation

Banded Buffer Producer/Consumer with mutual exclusion

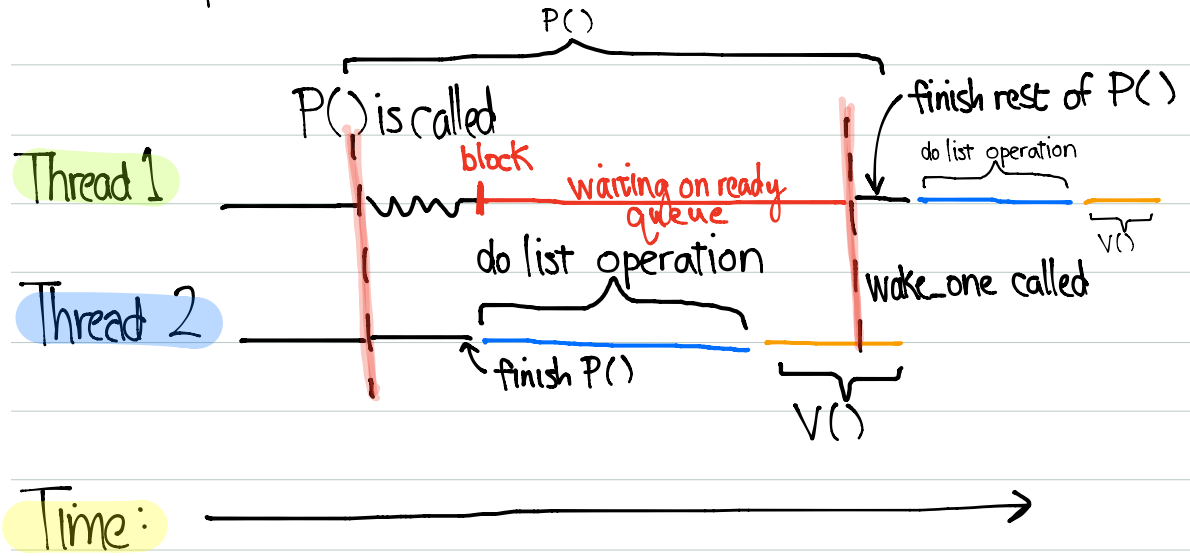
Producer

```
P(empty)   // mutex can only be 1 or 0
P(mutex)
// add item
V(mutex)
V(full)
```

Consumer

```
P(full)
P(mutex)
//rename item
V(mutex)
V(empty)
```

How semaphores avoid conflict



higher granularity → higher contention
↘ lower overhead

Necessary condition for deadlock is "hold and wait"

Deadlock detection algorithm - run periodically to make sure system OK, kernel's job.