CS 537: Intro to Operating Systems (Summer 2017)

Worksheet 7 - Hardware Locks

July 19th, 2017 (Wednesday)

Assume we have a new instruction called the **LoadAndStoreZero (LASZ)**, and it does the following atomically (here is C pseudo-code):

```
int LoadAndStoreZero(int *addr) {
    int old = *addr;
    *addr = 0;
    return old;
}
```

Build a working spin lock using the new instruction **LoadAndStoreZero (LASZ)**.

NOTE: For full points, your code should adhere to the C syntax.

```
typedef struct __lock_t {



} lock_t;


void init(lock_t *lock) {



}

void acquire(lock_t *lock) {







}

void release(lock_t *lock) {



}
```

```c
// Lock with Queues, Test-And-Set, Yield, and Wakeup

typedef struct __lock_t {
    int         flag;   // state of lock: 1=held, 0=free
    int         guard;  // use to protect flag, queue
    queue_t     *q;     // explicit queue of waiters
};


void lock_init(lock_t *lock) {
    lock->flag = lock->guard = 0;
    lock->q    = queue_init();
}


void lock(lock_t *lock) {
    while (xchg(&lock->guard, 1) == 1)
        ; // spin
    if (lock->flag == 0) { // lock is free: grab it!
        lock->flag = 1;
        lock->guard = 0;
    } else {               // lock not free: sleep
        queue_push(lock->q, gettid());
        lock->guard = 0;
        park();            // put self to sleep
    }
}


void unlock(lock_t *lock) {
    while (xchg(&lock->guard, 1) == 1)
        ; // spin
    if (queue_empty(lock->q))
        lock->flag = 0;
    else
        unpark(queue_pop(lock->q));
    lock->guard = 0;
}
```