CS350 Operating Systems

In-Class Problems: `ll` and `sc`

The following assembly language pseudo-code shows how the load linked (`ll`) and store conditional (`sc`) instructions can be used together to test-and-set a lock. In this code, `&lock` represents the address of the lock variable. The comments remind you how the `ll` and `sc` instructions behave.

```
// load the value 1 into register R1
li R1,1
// load the value of the lock variable into register R0 */
ll R0,&lock
// if the value of the lock variable has not changed since the ll
// instruction, store the value in R1 into the lock variable and
// set the value in R1 to 1 to indicate success. Otherwise,
// do not change the value of the lock variable and set the value
// of R1 to 0 to indicate failure.
sc R1,&lock
```

Suppose that a thread `T` executes these instructions as part of a call to spinlock acquire. Immediately after `T` executes the `sc` instructions, there are four possible situations, depending on the values in the registers `R0` and `R1`.

The table below lists these four possible situations. For each situation, indicate which of the following statements is true:

- `T` holds the lock.
- Some thread other than `T` holds the lock.
- No thread holds the lock.
- Not possible to determine whether the lock is held.

Indicate your answers by writing the correct statement in each box. The same statement may appear in more than one box.

Value of R0	Value of R1	Statement
0	0	
0	1	
1	0	
1	1	