一、Spin lock

共享数据被中断上下文和进程上下文访问，该如何保护呢？如果只有进程上下文的访问，那么可以考虑使用 semaphore 或者 mutex 的锁机制，但是现在中断上下文也参和进来，那些可以导致睡眠的 lock 就不能使用了，这时候，可以考虑使用 spin lock。

特点：

1. **是死等的锁机制：发生访问资源冲突时，有两种解决方式：一个是死等，一个是挂起当前进程，调度其他进程执行。spin lock 是一种死等的机制，当前的执行 thread 会不断的重新尝试直到获取锁进入临界区**
2. **只允许一个 thread 进入。semaphore 可以允许多个 thread 进入，spin lock 不行，一次只能有一个 thread 获取锁并进入临界区，其他的 thread 都是在门口不断的尝试。**
3. **执行时间短。由于 spin lock 死等这种特性，因此它使用在那些代码不是非常复杂的临界区（当然也不能太简单，否则使用原子操作或者其他适用简单场景的同步机制就 OK 了）**
4. **因为不睡眠，可以在中断上下文执行**

二、因为 spin lock 需要原子操作，所以在具体实现的时候，有两种方案：

1．Exch R1 0(R2) 交换操作包含了 read 和 write

2 .llsc:

ll:load linked——return the initial value

原子地将一个地址处的值加载到寄存器中，并设置一个锁定位。这个锁定位标志着对该地址的“链接”已经建立。

Sc:store conditional——return 1 if succeed(no other store to same memory location since preceeding load 确认为原子操作) and 0 otherwide

在之前使用 `LR` 指令读取的地址上尝试原子存储操作。如果在期间没有其他处理器修改了该地址的值，则存储条件成功，即将新值写回该地址。否则，存储条件失败，不会写回新值。

通过 LR 和 SC 指令配对使用，程序可以实现原子的加载和存储操作，从而避免了竞态条件和数据的不一致性。