1. Assume a write-invalidate cache-coherence protocol.
   a) For the processor containing a memory block in its cache, when would it send “invalidate messages” to the bus?

   b) For the processor containing a memory block in its cache and snoopying the bus, when would it change the blocks state based on messages snooped off of the bus?

2. Show actions for each situation: (assume write-allocate in the 2nd and 4th cases)
   1st: assume write-invalidate protocol with Proc 2 having a read miss on A
   2nd: assume write-allocate protocol with Proc 2 having a write miss on A
   3rd: assume write-invalidate protocol with Proc 2 having a read miss on A
   4th: assume write-allocate protocol with Proc 2 having a write miss on A
Assumptions:
- \( n \) is a shared variable initialized to the number of processors,
- \( count \) is a shared variable initialized to 0,
- \( arrive \) is a shared spin-lock variables which is initially unlocked
- \( depart \) is a shared spin-lock variables which is initially locked

```c
procedure synch()

lock(arrive);
    count := count + 1; /* count the processors as
    if count < n /* they arrive at barrier
        then unlock(arrive)
    else unlock(depart);

lock(depart);
    count := count - 1; /* count the processors as
    if count > 0 /* they leave barrier
        then unlock(depart)
    else unlock(arrive);
```

3. On which spin-lock variable are processors “waiting” at the “barrier” for all processors to arrive?

4. What is the value of \( count \) when all processors have arrived at the “barrier”?

5. How does \( count \) get reset to equal 0?

6. For a synch() within a loop as in the vector summation example, what prevents a “fast” processor from leaving the barrier and getting through the next barrier before all the other processors leave the barrier?