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 snooping the bus, when would it change the blocks state based on messages snooped off of the bus?

2. Show actions for each situation:
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

1st:

```
Proc 1  Proc 2
 A S     A I
```

```
Main Mem
 A
```

```
Proc 1  Proc 2
 A M     A I
```

```
Main Mem
 A
```

2nd:

```
Proc 1  Proc 2
 A S     A I
```

```
Main Mem
 A
```

```
Proc 1  Proc 2
 A M     A I
```

```
Main Mem
 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

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 the number of processors?