Problem 1: Mutual Exclusion
Write the assembly/pseudo code for a lock with the given atomic primitives:

```
test&set
lock(l)

swap
lock(l)

unlock(l)
unlock(l)
```

Problem 2: Snoopy Cache Protocol Derivation
Let’s re-derive the snoopy cache coherence protocols (no peeking at lecture slides). For the sake of simplicity let’s assume there are only two processors: P1 and P2. They each have their own private cache, and are connected by a common shared memory. Draw the transition arcs on the next page as if you are P1. For every state, be sure to consider:

- P1 reads
- P1 attempts to (and later) writes
- P2 reads
- P2 attempts to (and later) writes

Also don’t forget if P1 has a cache miss (read or write).

Problem 3: Scaling
How well will the snoopy protocols scale as the number of cores (and caches) grows? How would you make it scale better?
A) MSI (Modified, Shared, Invalid) Protocol

B) Make Your Own 4-State Protocol
Consider a system where it is possible (and much faster) for caches to send data to each other rather than all the way out to memory (they can also still exchange data with memory). Can you think of a 4-state protocol that could handle this better than MSI. (Hint: there are things other than MESI).