Lab5 prelab

1. **Why are transient states necessary?**
   In real hardware the state transitions along with their associated actions do not happen instantaneously and atomically due to interconnection network layers. If we do not have transient states, it might violate the single-writer, multiple-reader invariant and the data-value invariant. For example, say core 1 executes a read request for block B and the cache is in invalid state. To transition from invalid to shared state, a set of actions needs to be taken. During the time window, another request might have arrived at the directory and been serviced. If Core 1 had transitioned to S state immediately, then the system would have violated the single write/multiple readers invariant. To resolve this issue, transient states are introduced. The block stays in a transient state until all the associated actions of transition are completed and can safely transition to a stable state.

2. **Why does a coherence protocol use stall?**
   Stalls are used to make sure the read/write requests are handled correctly by the system. In most cases, read/write requests cannot be processed if there are pending coherence requests (i.e. the cache block is in transient state). If a cache block is in transient state, it might not have enough information to respond to read/write requests as the state is not stable. For example, if a load is issued to a block that is transiting to a shared state, the cache controller can only respond to it once the data is received in the cache. So, it needs to stall the load instruction.

3. **What is deadlock and how can we avoid it?**
   Deadlock is the phenomenon where no forward progress is made in a system because of two inter-dependent events. For example, if an event A can only occur after event B completes, and event B can only complete after event A occurs, then none of these two events will ever take place. This cyclic dependence between A and B indefinitely deadlocks the system. The system avoids deadlocks by having separate resources for different message types. For the purpose of an MSI protocol, different virtual networks are used to make sure that different types of requests are served in different queues. (request message, response message and forward message)

4. **What is the functionality of Put-Ack (i.e., WB-Ack) messages? They are used as a response to which message types?**
   Put-Ack is used to acknowledge cache eviction (i.e., replacement). They are used as a response to messages of type **request**. Specifically, Put-Ack is used to respond to PutS-NotLast, PutS-Last, PutM from Owner, and PutM from Non-owner such that the caches can complete their transitions from M or S to I.

5. **What determines who is the sender of a data reply to the requesting core? Which are the possible options?**
   The directory state determines the sender of the data. If the directory is in invalid state, the directory will request the data from the memory and send it to the requesting core. If

the directory state is shared, the directory's copy is up to date. So, the directory will forward the copy. If the directory is in modified state, the directory will forward the get message to the owner of the cache block. The owner will send the data.

6. **What is the difference between a Put-Ack and an Inv-Ack message?**
   Put-ack is issued by directory controller to acknowledge a transition from M or S to I state (i.e. cache eviction). Inv-ack is issued by cache controller to notify the requestor that the sharers has invalidated their copies in state S and has transitioned acknowledge another cache's transition from S to M sharers are invalidated to I state. The requestor can procced to upgrade to M state safely.