1. What is the cache coherency problem?
   In a multiprocessor system, each processor typically has its own cache. When different processors cache the same memory block, inconsistencies can arise if one processor modifies its cached copy without other caches being updated. This results in different processors working with stale or outdated data, which can lead to incorrect computations and data inconsistencies.
2. Explain the snooping mechanism.
   The snooping mechanism is used in multiprocessor systems with shared memory to maintain cache coherence. In this approach, each cache in the system "snoops" on the bus (a shared communication pathway) to monitor memory accesses (reads or writes) made by other processors. If a cache detects that another processor is accessing or modifying a memory block that it also holds, it takes appropriate action to maintain coherence.
3. What is cache coherency protocol? Explain the protocols below
   a. Write-invalidate.
      - When a processor writes to a memory block in its cache, all other caches holding copies of that block invalidate their copies.
      - After invalidation, any other processor attempting to read that block must fetch the updated value from the main memory or the processor that modified it.
   b. Write-update.
      - When a processor writes to a memory block, it broadcasts the update to all other caches that hold a copy of the block.
      - Instead of invalidating the block, the updated value is written to all caches, keeping them synchronized.
4. Explain the operation of the MESI protocol (Cache coherency).
   The MESI protocol (Modified, Exclusive, Shared, Invalid) is designed to maintain cache coherence in a multiprocessor system by defining the state of each cache line in the system. Here's a detailed look at how it works:

   ## Cache Line States:

   1. Modified (M):
      - Description: The cache line is both dirty (has been modified) and exclusive (not present in any other cache). The data is different from the main memory and is valid only in this cache.
      - Action: If a cache line is in the Modified state, any write to it updates the line in this cache and must also write it back to the main memory when the cache line is evicted.
   2. Exclusive (E):
      - Description: The cache line is clean (matches the main memory) and is exclusive to this cache (not present in any other cache).
      - Action: If a processor writes to a line in the Exclusive state, it transitions to the Modified state. If another cache requests the line, it sends the data, and the line moves to the Shared state in that cache.
   3. Shared (S):
      - Description: The cache line may be stored in multiple caches and is clean (matches the main memory). Multiple caches can have this line.
      - Action: If a processor writes to a line in the Shared state, it must first invalidate other caches that have this line and then transition the line to the Modified state in its own cache.
   4. Invalid (I):
      - Description: The cache line is not valid. It does not contain valid data and must be fetched from the main memory or another cache if needed.

   - Action: If a processor needs the line, it must request it from memory or other caches. If another cache has the line in the Modified state, it will be supplied with the data.

Operation of the MESI Protocol:

1. Cache Miss:
   - When a cache experiences a miss, it sends a request to the bus. Depending on the current state of the cache lines in other caches, the requesting cache will receive the block in the appropriate state:
     - Exclusive if the block is only in one cache.
     - Shared if the block is in multiple caches.
2. Write Operation:
   - Write to Modified: If a processor writes to a line in the Modified state, the data is updated in that cache and written back to the main memory upon eviction. All other caches holding this block are invalidated.
   - Write to Shared: If a write is attempted on a line in the Shared state, all caches holding that line are invalidated, and the writing cache line transitions to the Modified state.
3. Read Operation:
   - Read Miss: When a cache line is read and is not present, the requesting cache checks the bus for other caches that might hold the line. If it is in the Modified state in another cache, the data is sent to the requesting cache, and the state of the line in the other cache transitions to Invalid.
   - Read Hit: If a cache line in the cache is accessed and it is in the Shared or Modified state, the data is returned to the processor.