DAY 22 -111 DAYS VERIFICATION CHALLENGE

Topic: Cache coherence

Skill: Cache memories

DAY 22 CHALLENGE:

1. What is the cache coherency problem?

Cache coherency is a significant challenge in multiprocessor systems where multiple processors have their own local caches. It refers to the consistency of data stored in these caches. The problem arises when each processor has its own copy of the same memory location, and one processor modifies its copy. Without a proper mechanism, the other processors may continue to work with outdated data, leading to incorrect program execution.

Key Concepts

1. **Multiple Copies of Data:**
    - Each processor has its own cache.
    - A memory location might be cached in multiple caches.
2. **Inconsistency:**
    - One processor writes to its cache.
    - Other processors' caches have stale data.

Example Scenario

- **Processor 1** has variable X in its cache.
- **Processor 2** also has X in its cache.
- **Processor 1** updates X.
- **Processor 2** still has the old value of X unless notified of the update.

2. Explain snooping mechanism

The snooping mechanism is a cache coherency protocol used in multiprocessor systems to ensure that all caches in the system have a consistent view of memory. It involves monitoring (or "snooping") the communication bus to keep track of the read and write operations performed by other processors. Here's how it works:

How Snooping Mechanism Works

1. **Bus Monitoring:**
    - Each cache in the system monitors the bus for any transactions (read or write operations) initiated by other processors.
    - This allows each cache to detect when another processor is accessing or modifying data that it might also have.
2. **Cache Controllers:**
    - Each cache has a controller that snoops on the bus.
    - The controller listens for memory transactions and takes appropriate actions to maintain cache coherency.
3. **Actions Taken on Bus Transactions:**
    - **Read Miss:** When a processor wants to read a data block that is not in its cache, it issues a read request on the bus.
        - Other caches snoop this request and, if they have the data, they may provide it to the requesting processor.
        - The state of the data in the caches may change depending on the protocol used (e.g., MESI, MOESI).
    - **Write Miss:** When a processor wants to write to a data block that is not in its cache, it issues a write request on the bus.
        - Other caches snoop this request and, if they have the data, they may invalidate their copies of the data.
    - **Write Hit:** If the data is present in the cache and in a state that allows writing, the processor updates the cache directly and may broadcast an invalidation request to other caches if necessary.

3. What is cache coherency protocol. Explain below protocols: -

A cache coherency protocol is a set of rules and mechanisms that ensures consistency and correctness of data across all caches in a multiprocessor system. These protocols prevent scenarios where different processors have different values for the same memory location in their caches, which can lead to incorrect program execution. Cache coherency protocols manage how and when data is propagated and updated among the caches to maintain a consistent view of memory.

i. write-invalidate

- When a processor writes to a data block, it sends an invalidation signal to other caches.
- Other caches invalidate their copies of the data block.
- The writing processor updates its cache and can then proceed with the write.

Advantages:

- Simple to implement.
- Reduces the number of write operations broadcasted on the bus.

Disadvantages:

- High latency due to invalidation before writing.
- Increased invalidation traffic in systems with high data sharing.

ii. Write-update

- When a processor writes to a data block, it broadcasts the new value to all other caches.
- Other caches update their copies of the data block with the new value.

Advantages:

- Ensures all caches have the most recent data.
- Reduces the number of invalidations.

Disadvantages:

- Can generate high bus traffic due to frequent updates.
- More complex to implement compared to write-invalidate.

4. Explain operation of MESI Protocol (Cache Coherency)

The MESI protocol (Modified, Exclusive, Shared, Invalid) is a cache coherency protocol used in multiprocessor systems to maintain consistency among caches. Each cache line can be in one of four states: Modified (M), Exclusive (E), Shared (S), or Invalid (I). The protocol ensures that any changes to data in one cache are reflected in other caches or the main memory as needed.

States of MESI Protocol

1. **Modified (M):**
    - The cache line is modified from the value in the main memory.
    - The cache is the only one with the modified data.
    - The data must be written back to the main memory before being replaced.
2. **Exclusive (E):**
    - The cache line is consistent with the main memory.
    - The cache is the only one with this data.

- The cache can modify the data without informing other caches.

3. **Shared (S):**
   - The cache line is consistent with the main memory.
   - The data may be present in other caches.
   - Any write to this line must notify other caches to invalidate their copies.
4. **Invalid (I):**
   - The cache line is not valid.
   - The data must be fetched from another cache or main memory before use.