**Test Plan:**

1. **Hit and Miss Test:**
   **1.1 Read Miss**
   - Read an address from the trace file. It will be a compulsory miss.

   **1.2 Read Hit**
   - Read from the same address again. Now it has to be a hit.

   **1.3 Write Miss**
   - Give a different address to write to a cache line. It should be a miss.

   **1.4 Write Hit**
   - Write to the same address. Now it should be a hit.

2. **Testing PLRU:**
   **2.1 Access pattern validation**
   - Read from 16 addresses such that all those point to the same index. All of them will be compulsory misses.
   - Read from same 16 addresses, but now all of them will be hit.
   - Check the PLRU bits, and all the bits should be 1's because the cache line was sequentially filled.

   **2.2 Checking Replacement policy**

   **2.2.1 Checking before any access**
   - Now perform a read with an address such that it goes to the same index but with a different tag. Now it will be a miss.
   - Now way 0 has to be evicted because it was least recently used with respect to PLRU.

   **2.2.2 Checking replacement policy after access.**
   - Now read from way 7 address.
   - Now give a new address with the same index but a different tag.
   - According to PLRU, way 8 should be evicted.

   **2.2.3 Checking after multiple access to the same cache line.**
   - Now read way 13 address.
   - Write to way 13. Read again from Way 13. Perform a write again to way 13.
   - Despite multiple accesses, the PLRU bits should not change after each access.
   - Give an address again with the same index but a different tag. Way 2 should be evicted.

   **2.3 Check for cache clearing**
   - After performing a set of accesses and replacements, PLRU bits a set to a particular state.
   - Now perform a clear cache by giving operation code 8.
   - All PLRU bits should be 0's

3. **Testing Mode 8 and 9:**
    - Give a set of addresses with different operation codes.

    **3.1 Mode 9**
    - Give the command with operation code 9. See if the contents of caches are printed.

    **3.2 Mode 8**
    - To clear the cache, we give command with operation code 8.
    - Now give command with operation code 9 again. Nothing should be printed because we cleared the cache.
4. **Testing the correct implementation of MESI protocol for Processor Read:**

    **4.1 Checking for Invalid->Shared**
    - Give a processor read to some address, such that the last two bits of the 32bit address is 00, and there would be a BUS READ operation.
    - Now, give operation code 9, the state should be in SHARED, and there would be a BUS READ operation.
    - Give a processor read to the same address again, it should still be in SHARED state.

    **4.2 Checking for Invalid->Exclusive**
    - Give a processor read to some address, such that the last two bits of the 32bit address is 10 or 11, and there would be a BUS READ operation.
    - Now, give operation code 9, the state should be in EXCLUSIVE.
    - Give a processor read to the same address again, it should still be in EXCLUSIVE state.
5. **Testing the correct implementation of MESI protocol for Processor Write:**

    **5.1 Checking for Invalid->Modified**
    - Give a processor write to some address, and there would be a BUS RWIM operation.
    - Now, give operation code 9, the state should be in MODIFIED.
    - Give a processor read to the same address again, it should still be in MODIFIED state.
    - Give a processor write to the same address again, it should still be in MODIFIED state.

    **5.2 Checking for Exclusive->Modified**
    - Give a processor write to some address.
    - Now, give operation code 9, the state should be in MODIFIED.
    - Give a processor read to the same address again, it should still be in MODIFIED state.
    - Give a processor write to the same address again, it should still be in

MODIFIED state.

**5.3 Checking for SHARED->Modified**

- Give a processor write to some address, and there would be a BUS INVALIDATE operation.
- Now, give operation code 9, the state should be in MODIFIED.
- Give a processor read to the same address again, it should still be in MODIFIED state.
- Give a processor write to the same address again, it should still be in MODIFIED state.

6. **Testing the correct implementation of MESI protocol for Snooping Read:**

   **6.1 Checking for Modified->Shared**

   - Give a processor write to some address.
   - Now, give operation code 9, the state should be in MODIFIED.
   - Now, give operation code 3, to the same address, the state should be in SHARED.
   - Give operation code 3 to the same address again, it should still be in SHARED state.

   **6.2 Checking for Exclusive->Shared**

   - Give a processor read to some address, such that the last two bits of the address are 10 or 11.
   - Now, give operation code 9, the state should be in MODIFIED.
   - Now, give operation code 3, to the same address, the state should be in SHARED.
   - Give operation code 3 to the same address again, it should still be in SHARED state.

7. **Testing the correct implementation of MESI protocol for Snooping Write:**

   - Give operation code 4 to some address, the state should be in INVALID.

8. **Testing the correct implementation of MESI protocol for Snooping Read With Intent to Modify:**

   **8.1 Checking for Modified->Invalid**

   - Give a processor write to some address.
   - Now, give operation code 9, the state should be in MODIFIED.
   - Now, give operation code 5, to the same address, the state should be in INVALID.
   - Give operation code 5 to the same address again, it should still be in INVALID state.

**8.2 Checking for Exclusive->Invalid**

- Give a processor read to some address, such that the last two bits of the address are 10 or 11.
- Now, give operation code 9, the state should be in EXCLUSIVE.
- Now, give operation code 5, to the same address, the state should be in INVALID.
- Give operation code 5 to the same address again, it should still be in INVALID state.

**8.3 Checking for Shared->Invalid**

- Give a processor read to some address, such that the last two bits of the address are 00.
- Now, give operation code 9, the state should be in SHARED.
- Now, give operation code 5, to the same address, the state should be in INVALID.
- Give operation code 5 to the same address again, it should still be in INVALID state.

9. **Testing the correct implementation of MESI protocol for Snooping Invalidate:**

**9.1 Checking for Exclusive->Invalid**

- Give a processor read to some address, such that the last two bits of the address are 10 or 11.
- Now, give operation code 9, the state should be in EXCLUSIVE.
- Now, give operation code 6, to the same address, the state should be in INVALID.
- Give operation code 6 to the same address again, it should still be in INVALID state.

**9.2 Checking for Shared->Invalid**

- Give a processor read to some address, such that the last two bits of the address are 00.
- Now, give operation code 9, the state should be in SHARED.
- Now, give operation code 6, to the same address, the state should be in INVALID.
- Give operation code 6 to the same address again, it should still be in INVALID state.

## 10. Test for Messages between L2 and L1

### 10.1 Checking for Sendline Message

- Give a processor read to some address, it will be a miss and after it performs a BUS READ, there should be a message from L2 to L1 as SENDLINE.
- Give a processor write to same address, it will be a hit and there should be a message from L2 to L1 as SENDLINE.

### 10.2 Checking for Evictline Message

- Give 16 processor reads to some addresses such that index is same and tag is different for all, it will be 16 misses and after it performs 16 BUS READs, there should be 16 messages from L2 to L1 as SENDLINE.
- Give processor read to the some addres with same index as above, it will be a collision miss.
- According to PLRU way 0 should be evicted, there should be a message from L2 to L1 saying EVICTLINE with address of way 0.
- There will be BUS READ to the address given with processor read, there should be a SENDLINE message from L2 to L1.