LLD designs are of two types: Machine coding where you actually build something. And 'Design' where you mention the classes and inheritance/composition relations.

Machine coding is similar to HLD where you clarify requirements of the object you are building, and then move on to code it. It requires some show to executable code.

1. Readability
    1. First draw the interaction diagram, and make sure the requirements are clear. You first tell what you'll do, then code it.
    2. Walk your interviewer through the code while you write it.
    3. Clarify requirements as you go along.
    4. Don't refrain from using the board. It's better to invest 2 minutes for a smooth flow.
2. Correctness
    1. Before you write a major function, define what it does.
    2. Set 1-2 tests for every major function. This makes you confident of correctness. Try to make the tests runnable (even some examples running in the main function help).
    3. Is concurrency an issue? Most of these cases can't be tested in the limited time, so try to remove such issues. Can you use ordering? Code locks? Data Locks?
    4. Pick one feature at a time. Write a test to confirm.
3. Performance
    1. Look for bottlenecks in your code. You need to be very familiar with time complexities. Code with O(n^2) time complexity is not acceptable.
    2. Most performance problems related to search can be solved using inverse mapping or priority queues. Look at how a map and queue can work together.

Tips:

a) Clarify requirements

b) Code in a structured manner and talk them through the process

c) Attack the most critical parts first.

Code Link:
https://github.com/coding-parrot/Low-Level-Design/tree/master/distributed-cache