实验报告

一．实验题目：

高速缓存一致性

二．实验内容：

缓存一致性模拟系统

三．实验环境：

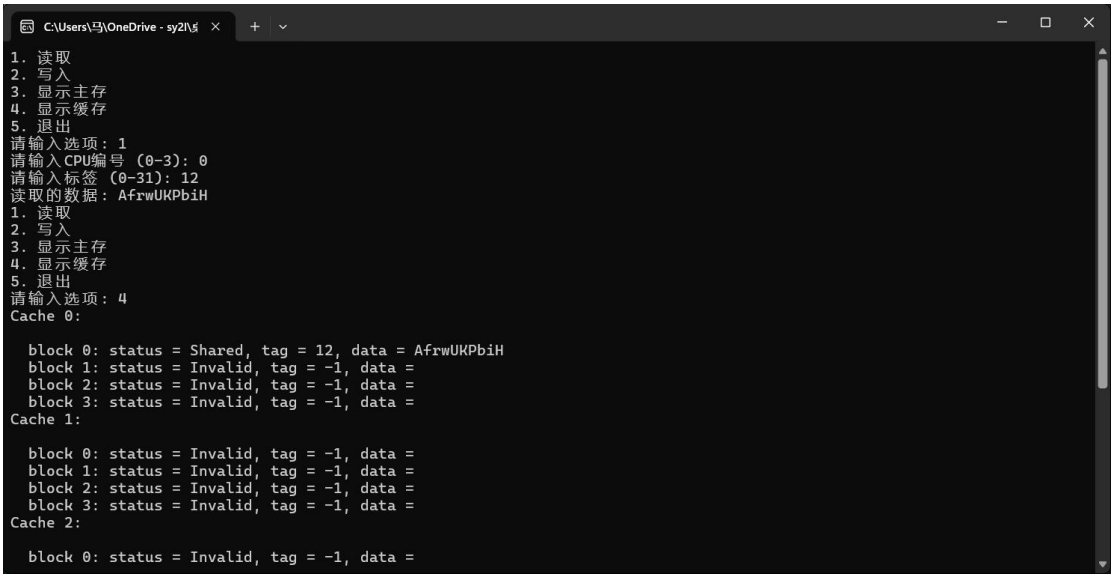
环境：Windows11

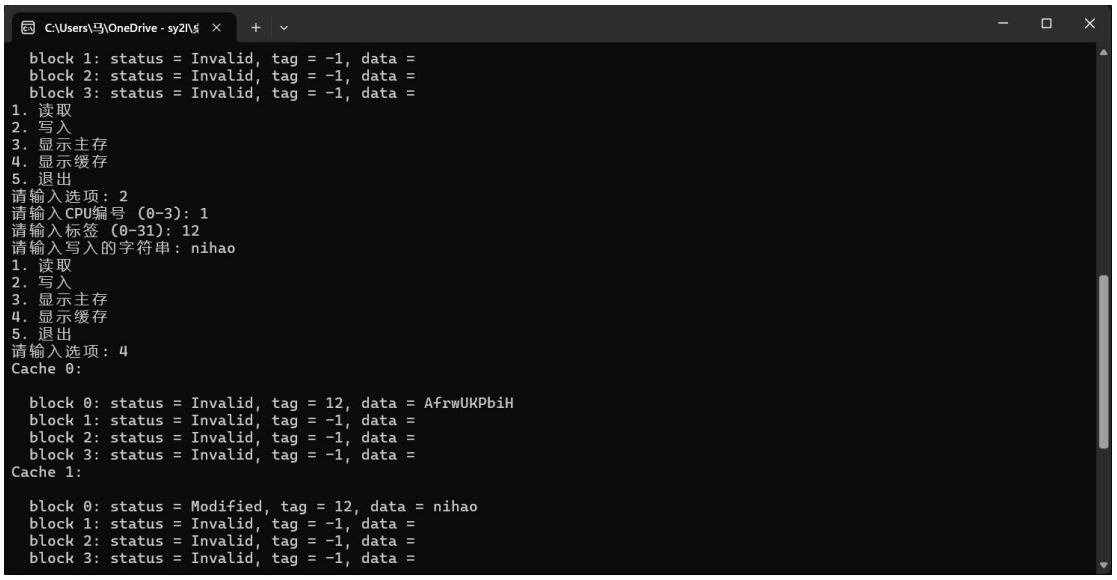
语言：c++

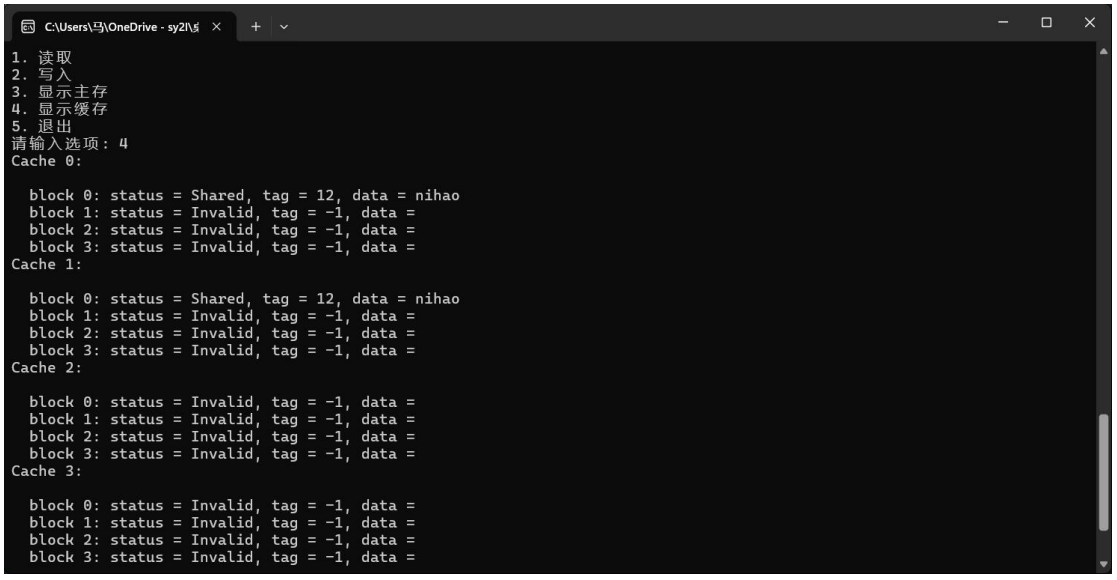
四．实验设计

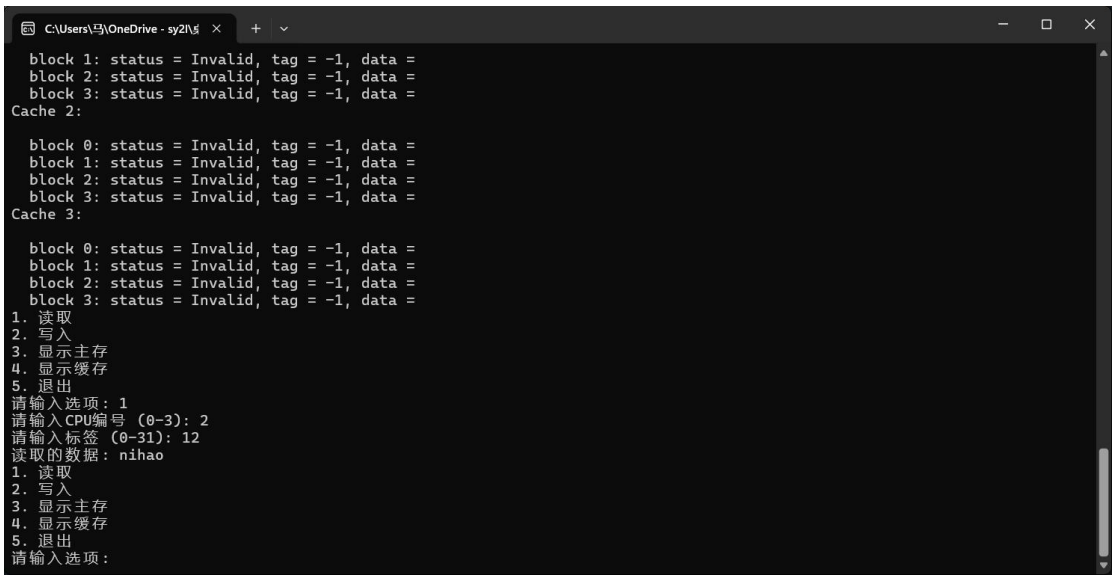
上述代码实现了一个简单的缓存一致性模拟系统。主要结构包括主存（32个块）和四个缓存（每个缓存包含四个块），通过随机生成的数据初始化主存块。缓存块具有状态（Invalid、Modified、Shared）来标识其是否有效以及数据是否已经被修改或共享。写入操作（`cpuwrite`函数）根据缓存块的状态来管理数据的更新和缓存一致性，优先考虑共享块和已修改块的替换，最后再考虑随机替换。读取操作（`cpuread`函数）同样需要考虑多个缓存之间的状态一致性，如果数据在其他缓存中被修改，则先写回主存后再进行读取，确保数据的正确性。整体设计通过管理缓存状态和标签来实现对数据的读取和写入操作，模拟了多核系统中的缓存一致性问题处理机制。

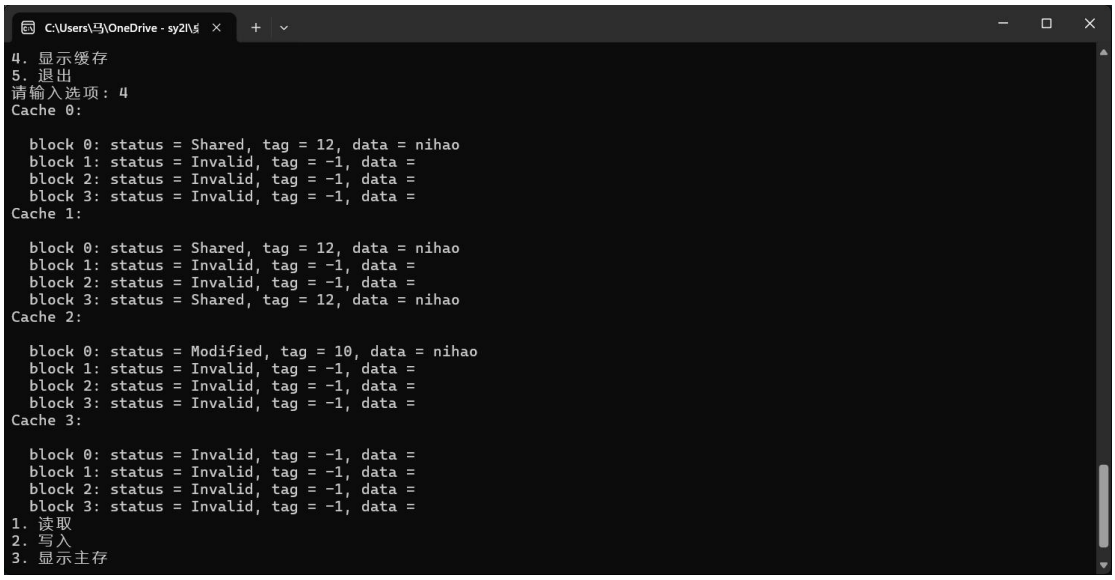
五．运行过程截图

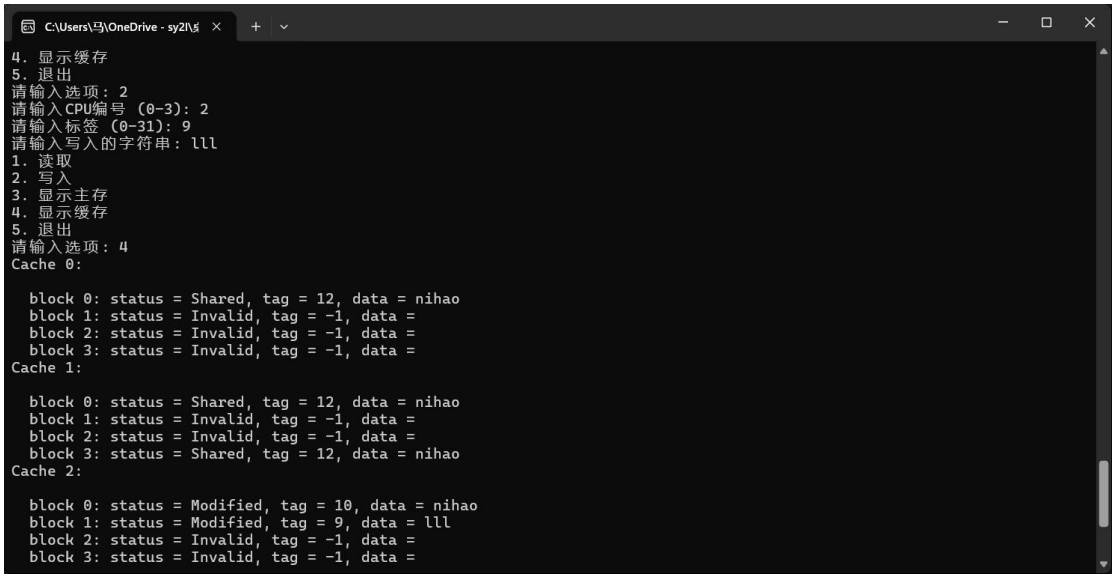



C:\Users\马\OneDrive - sy2l\桌
block 1: status = Invalid, tag = -1, data =
block 2: status = Invalid, tag = -1, data =
block 3: status = Invalid, tag = -1, data =
1. 读取
2. 写入
3. 显示主存
4. 显示缓存
5. 退出
请输入选项: 2
请输入CPU编号 (0-3): 1
请输入标签 (0-31): 12
请输入写入的字符串: nihao
1. 读取
2. 写入
3. 显示主存
4. 显示缓存
5. 退出
请输入选项: 4
Cache 0:
block 0: status = Invalid, tag = 12, data = AfrwUKPbiH
block 1: status = Invalid, tag = -1, data =
block 2: status = Invalid, tag = -1, data =
block 3: status = Invalid, tag = -1, data =
Cache 1:
block 0: status = Modified, tag = 12, data = nihao
block 1: status = Invalid, tag = -1, data =
block 2: status = Invalid, tag = -1, data =
block 3: status = Invalid, tag = -1, data =


C:\Users\马\OneDrive - sy2l\桌
1. 读取
2. 写入
3. 显示主存
4. 显示缓存
5. 退出
请输入选项: 4
Cache 0:
block 0: status = Shared, tag = 12, data = nihao
block 1: status = Invalid, tag = -1, data =
block 2: status = Invalid, tag = -1, data =
block 3: status = Invalid, tag = -1, data =
Cache 1:
block 0: status = Shared, tag = 12, data = nihao
block 1: status = Invalid, tag = -1, data =
block 2: status = Invalid, tag = -1, data =
block 3: status = Invalid, tag = -1, data =
Cache 2:
block 0: status = Invalid, tag = -1, data =
block 1: status = Invalid, tag = -1, data =
block 2: status = Invalid, tag = -1, data =
block 3: status = Invalid, tag = -1, data =
Cache 3:
block 0: status = Invalid, tag = -1, data =
block 1: status = Invalid, tag = -1, data =
block 2: status = Invalid, tag = -1, data =
block 3: status = Invalid, tag = -1, data =


C:\Users\马\OneDrive - sy2l\桌
block 1: status = Invalid, tag = -1, data =
block 2: status = Invalid, tag = -1, data =
block 3: status = Invalid, tag = -1, data =
Cache 2:
block 0: status = Invalid, tag = -1, data =
block 1: status = Invalid, tag = -1, data =
block 2: status = Invalid, tag = -1, data =
block 3: status = Invalid, tag = -1, data =
Cache 3:
block 0: status = Invalid, tag = -1, data =
block 1: status = Invalid, tag = -1, data =
block 2: status = Invalid, tag = -1, data =
block 3: status = Invalid, tag = -1, data =
1. 读取
2. 写入
3. 显示主存
4. 显示缓存
5. 退出
请输入选项: 1
请输入CPU编号 (0-3): 2
请输入标签 (0-31): 12
读取的数据: nihao
1. 读取
2. 写入
3. 显示主存
4. 显示缓存
5. 退出
请输入选项:





六. 实验结论

通过实验，我成功地完成了一个简单的缓存一致性模拟系统的编写和测试。在实验过程中，我学习了如何设计和实现缓存一致性协议，包括处理缓存块状态、数据写入和读取的逻辑。这个系统能够模拟多个 CPU 核心同时访问共享主存数据时可能出现的问题，如数据一致性和缓存块的替换策略。然而，实验中也发现了一些不足之处。首先，缓存一致性协议的实现还比较简单，没有涵盖复杂的高级协议，如 MESI 或 MOESI 等，这些协议可以更精确地控制缓存块的状态转换和共享数据的管理。其次，当前的实现并未考虑并发访问带来的性能和一致性挑战，实际应用中可能需要更复杂的同步和协调机制来保证数据的正确性和效率。

七. 实验感想

这里可以写实验过程中遇到的困难、如何调试，也可以写实验感想。随意发挥。