计算机体系结构Lab5

实验目的

- 熟悉Tomasulo模拟器和cache一致性模拟器（监听法和目录法）的使用
- 加深对Tomasulo算法的理解，从而理解指令级并行的一种方式-动态指令调度
- 掌握Tomasulo算法在指令流出、执行、写结果各阶段对浮点操作指令以及load和store指令进行什么处理；给定被执行代码片段，对于具体某个时钟周期，能够写出保留站、指令状态表以及浮点寄存器状态表内容的变化情况。
- 理解监听法和目录法的基本思想，加深对多cache一致性的理解
- 做到给出指定的读写序列，可以模拟出读写过程中发生的替换、换出等操作，同时模拟出cache块的无效、共享和独占态的相互切换

实验要求

一.Tomasulo算法模拟器

使用模拟器进行以下指令流的执行并对模拟器截图、回答问题

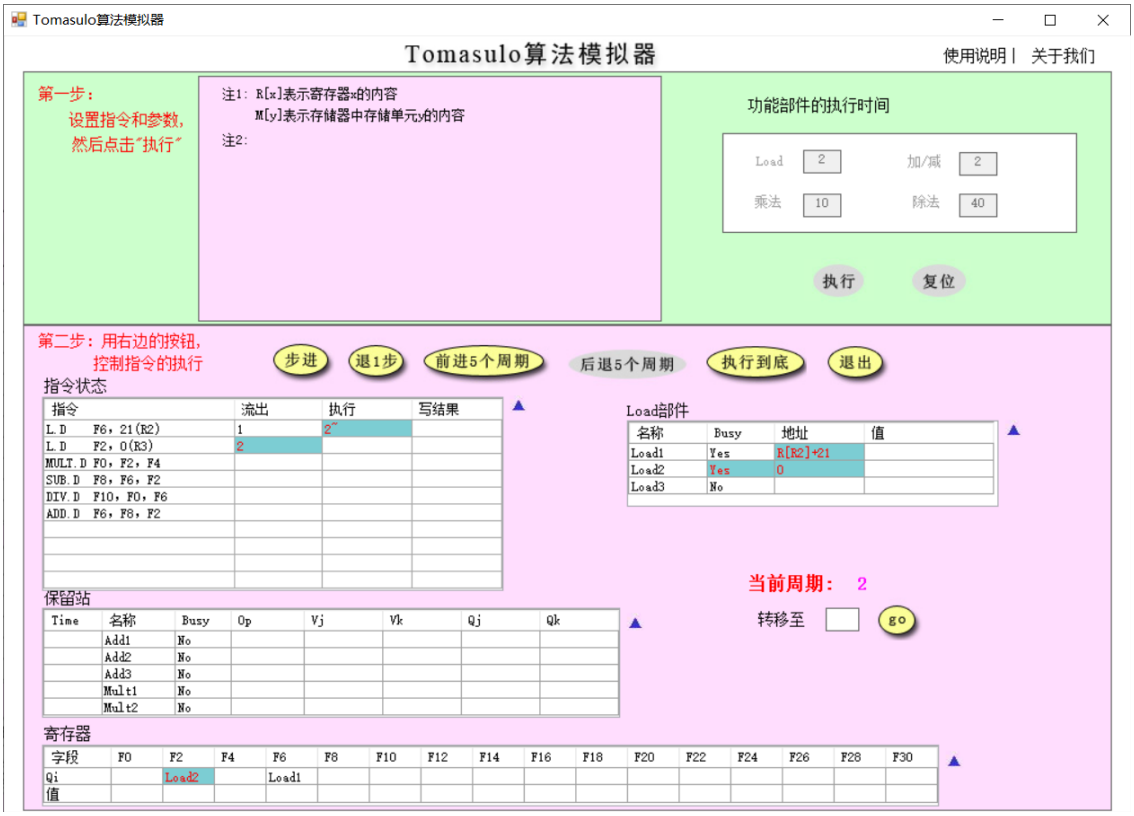
```
L.D  F6, 21（R2）
L.D  F2, 0（R3）
MUL.D  F0, F2, F4
SUB.D  F8, F6, F2
DIV.D  F10, F0, F6
ADD.D  F6, F8, F2
```

假设浮点功能部件的延迟时间：加减法2个周期，乘法10个周期，load/store2个周期，除法40个周期。

1. 分别截图（当前周期2和当前周期3），请简要说明load部件做了什么改动

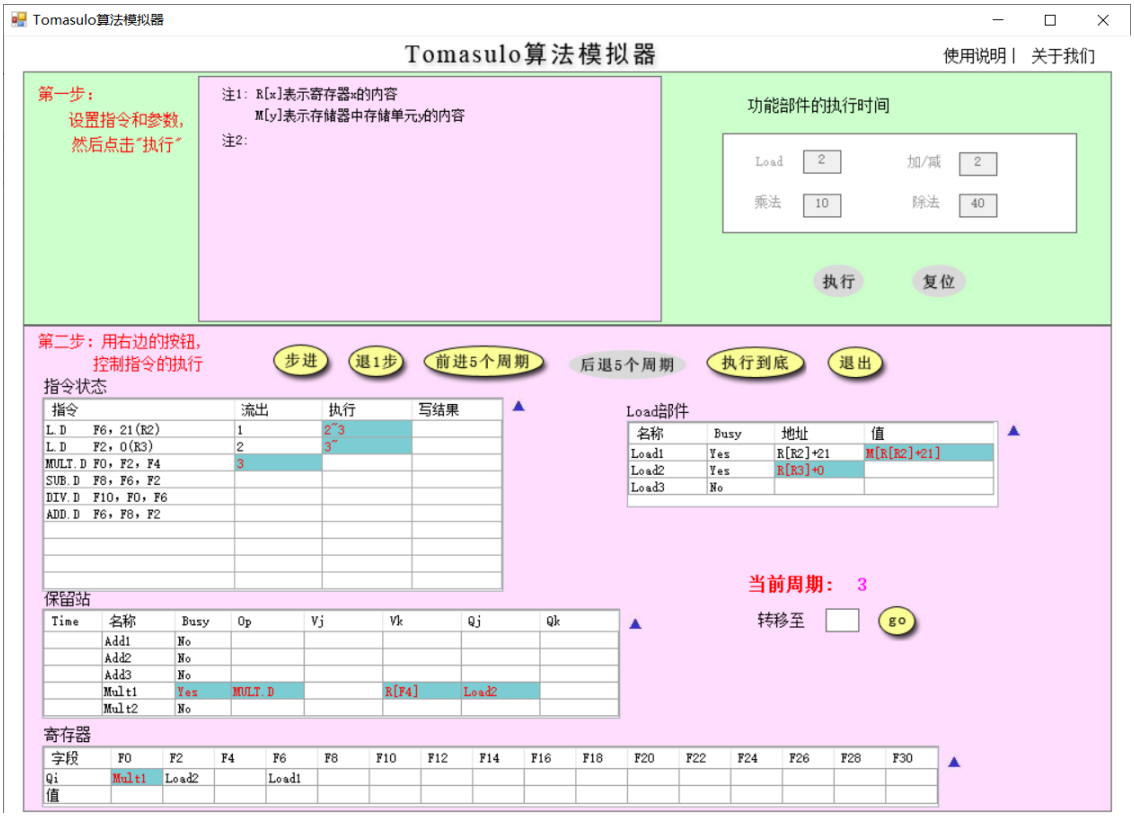
   周期2：

   load部件：load1保存R2地址；load2 busy置yes



周期3：

load部件：load1保存R2对应地址的值；load2保存R3地址



2. 请截图（MUL.D刚开始执行时系统状态），并说明该周期相比上一周期整个系统发生了哪些改动（指令状态、保留站、寄存器和Load部件）

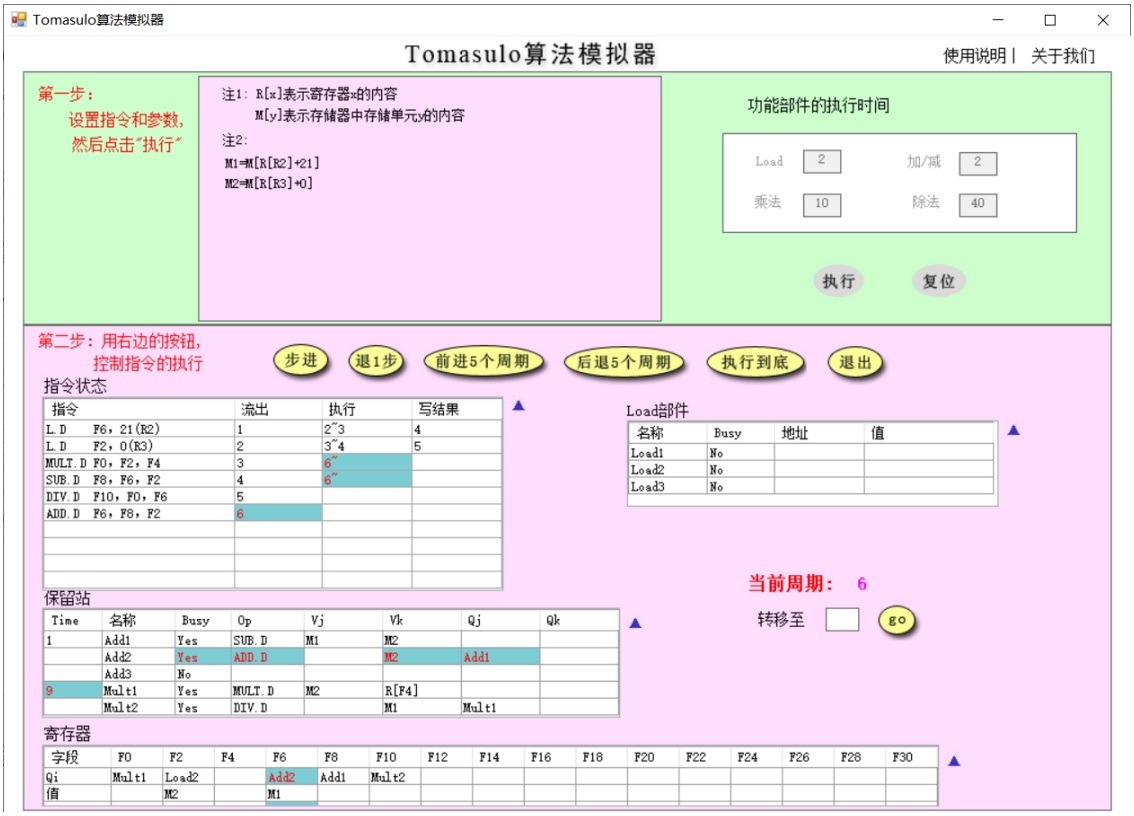
此时为周期6：

指令状态：发射指令`ADD.D F6, F8, F2`

保留站：Add2保留站被占用，执行的MULT.D指令开始倒计时

寄存器：Qi(F6) = Add2

load部件：无改动

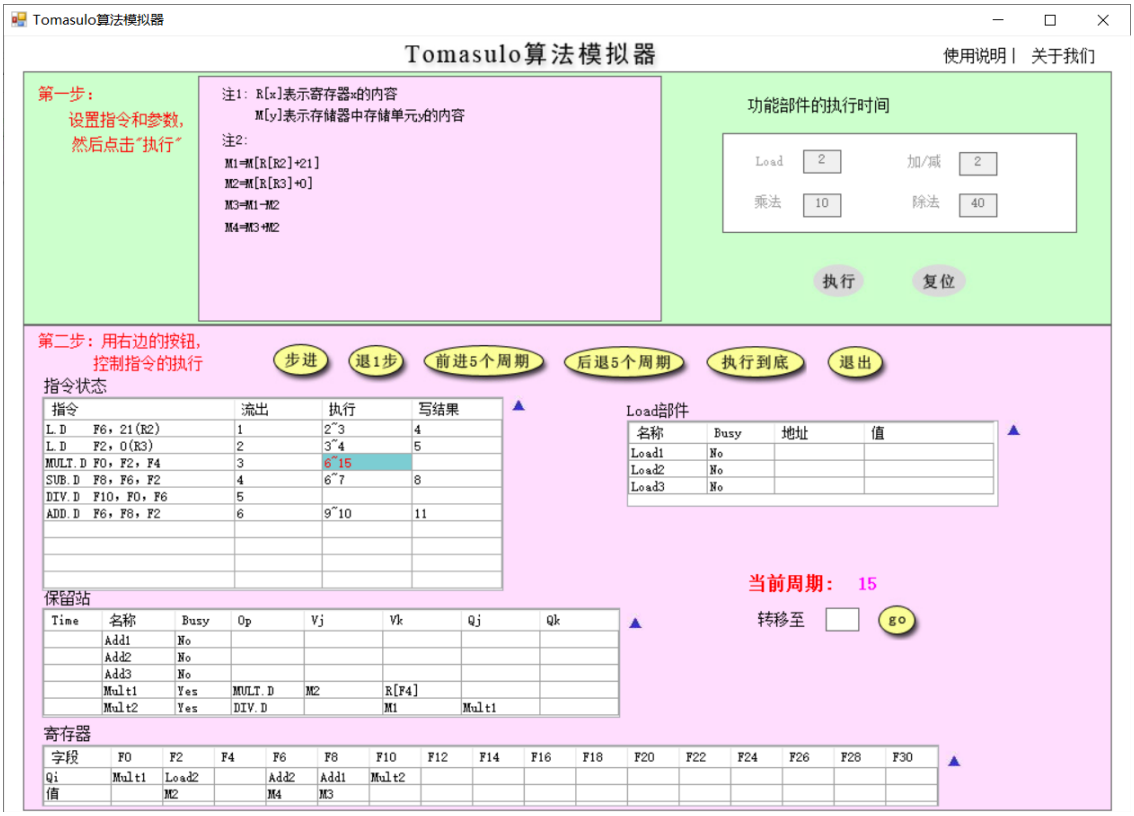


3. 简要说明是什么相关导致MUL.D流出后没有立即执行

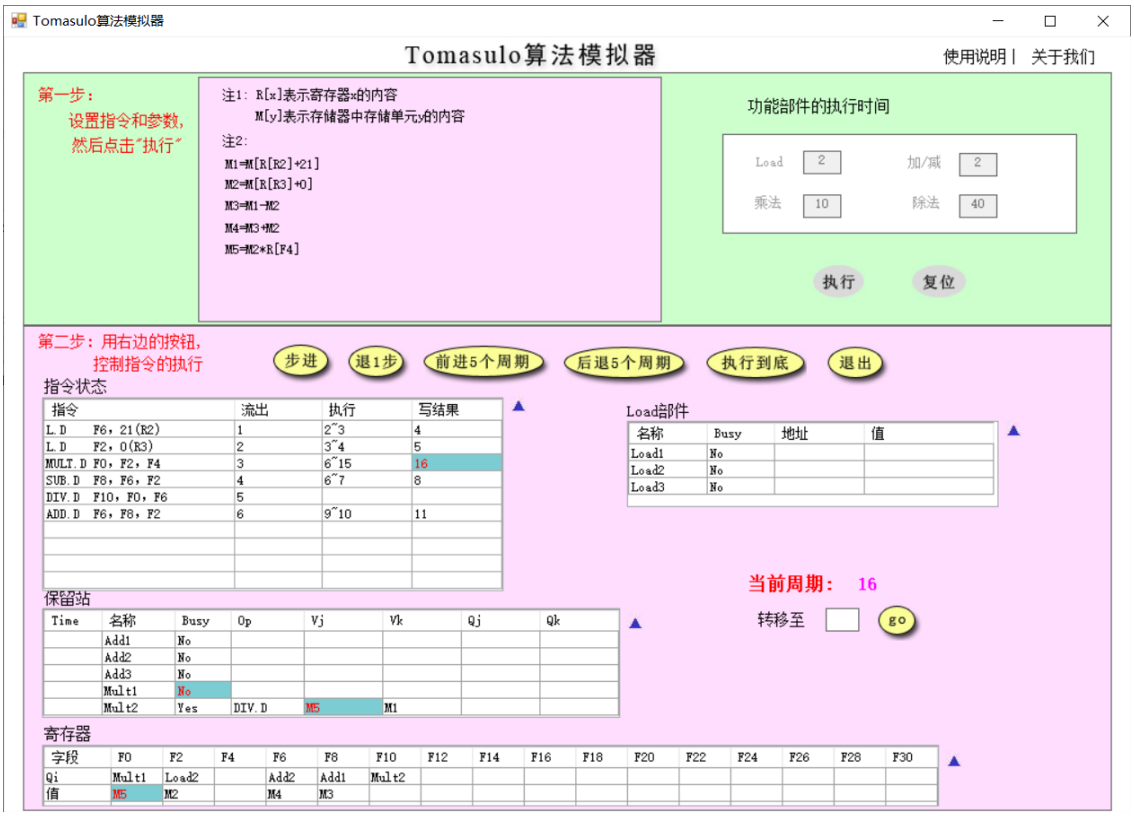
   F2在上一条L.D指令中还未写入

4. 请分别截图（15周期和16周期的系统状态），并分析系统发生了哪些变化

   周期15：



周期16：



周期15->周期16

MULT.D指令执行完，写回结果，释放保留站，将结果广播到M5对应的保留站

5. 回答所有指令刚刚执行完毕时是第多少周期，同时请截图（最后一条指令写CBD时认为指令流执行结束）

57个周期

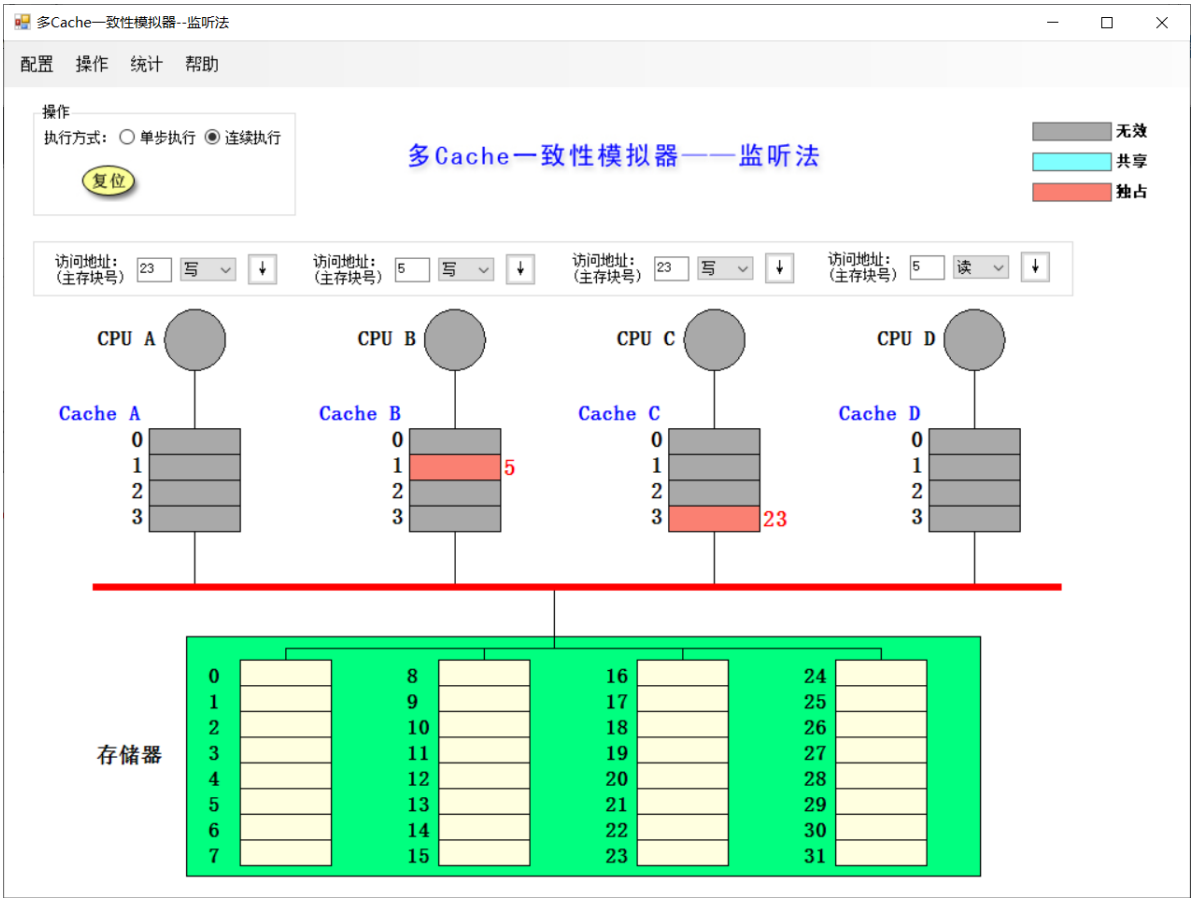
周期57：

二.多cache一致性算法-监听法

1.利用模拟器进行下述操作，并填写下表

所进行的访问	是否发生了替换?	是否发生了写回?	监听协议进行的操作与块状态改变
CPU A 读第5块	替换了CacheA的块1		CacheA发出读不命中，存储器传输第5块到CacheA的块1，该块从空闲变为共享
CPU B 读第5块	替换了CacheB的块1		CacheB发出读不命中，存储器传输第5块到CacheB的块1，该块从空闲变为共享
CPU C 读第5块	替换了CacheC的块1		CacheC发出读不命中，存储器传输第5块到CacheC的块1，该块从空闲变为共享
CPU B 写第5块			CacheB发出作废，CacheA的块1、CacheC的块1从共享变为空闲，CacheB的块1从共享变为独占
CPU D 读第5块	替换了CacheD的块1	写回CacheB的块1	CacheD发出读不命中，CacheB写回块1，该块从独占变为共享，存储器传输第5块到CacheD的块1，该块从空闲变为共享
CPU B 写第21块	替换了CacheB的块1		CacheB发出写不命中，存储器传输第21块到CacheB的块1，该块从共享变为独占
CPU A 写第23块	替换了CacheA的块3		CacheA发出写不命中，存储器传输第23块到CacheA的块3，该块从空闲变为独占
CPU C 写第23块	替换了CacheC的块1	写回CacheA的块3	CacheC发出读不命中，CacheA写回块3，该块从独占变为空闲，存储器传输第23块到CacheD的块3，该块从空闲变为独占
CPU B 读第29块	替换了CacheB的块1	写回CacheB的块1	CacheB写回第21块，发出写不命中，存储器传输第29块到CacheB的块1，该块从独占变为共享
CPU B 写第5块	替换了CacheB的块1		CacheB发出写不命中，存储器传输第5块到CacheB的块1，该块从共享变为独占，CacheD的块1从共享变为空闲

2. 请截图，展示执行完以上操作后整个cache系统的状态

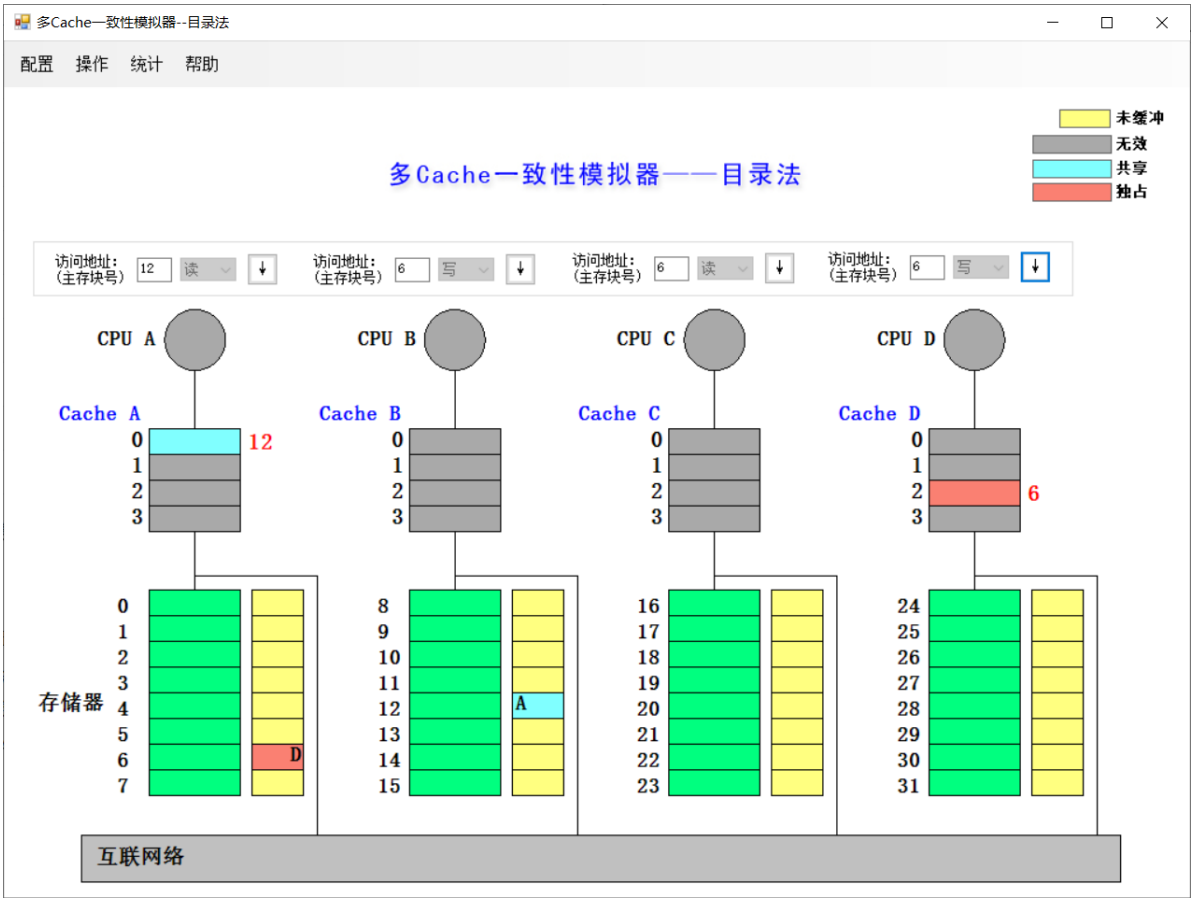


三.多cache一致性算法-目录法

1.利用模拟器进行下述操作，并填写下表

所进行的访问	监听协议进行的操作与块状态改变
CPU A 读第6块	CacheA发出读不命中到本地宿主，宿主节点传输第6块到CacheA的块2，第6块从未缓冲变为共享，共享集合{A}，CacheA的块2从空闲变为共享
CPU B 读第6块	CacheB发出读不命中到第6块的宿主，宿主节点传输第6块到CacheB的块2，第6块共享集合{A, B}，CacheB的块2从空闲变为共享
CPU D 读第6块	CacheD发出读不命中到第6块的宿主，宿主节点传输第6块到CacheD的块2，第6块共享集合{A, B, D}，CacheD的块2从空闲变为共享
CPU B 写第6块	CacheB发出写命中到第6块的宿主，宿主节点发出作废6到CacheA、CacheD，CacheA、CacheD的块2从共享变为空闲，CacheB的块2从共享变为独占，第6块从共享变为独占，共享集合{B}
CPU C 读第6块	CacheC发出读不命中到第6块的宿主，宿主节点发出取数据块6到块6的远程节点，CacheB传输块2到宿主节点，宿主节点传输第六块到CacheC的块2，第6块从独占变为共享，共享集合{B, C}，CacheB的块2从独占变为共享，CacheC的块2从空闲变为共享
CPU D 写第20块	CacheD发出写不命中到第20块的宿主，宿主节点传输第20块到CacheD的块0，第20块从未缓冲变为独占，共享集合{D}，CacheD的块0从空闲变为独占
CPU A 写第20块	CacheA发出写不命中到第20块的宿主，宿主节点发出取数据块20和作废20到块20的远程节点，CacheB传输块0到宿主节点，宿主节点传输第20块到CacheA的块0，第20块共享集合{A}，CacheA的块0从空闲变为独占，CacheC的块2从独占变为空闲
CPU D 写第6块	CacheD发出写命中到第6块的宿主，宿主节点发出作废6到CacheB，CacheB的块2从共享变为空闲，CacheD的块2从空闲变为独占，第6块从共享变为独占，共享集合{D}
CPU A 读第12块	CacheA发出写回到第20块的宿主，第20块从独占变为未缓冲，共享集合{}，CacheA的块0从独占变为空闲； CacheA发出读不命中到第12块的宿主，宿主节点传输第12块到CacheA的块0，第12块从未缓冲变为共享，共享集合{A}，CacheA的块0从空闲变为共享

2. 请截图，展示执行完以上操作后整个cache系统的状态



四.综合问答

1. 目录法和监听法分别是集中式和基于总线，两者优劣是什么？（言之有理即可）

   目录法：

   > 优点
   >
   > 易拓展，对总线带宽占用少
   >
   > 缺点：
   >
   > 处理器多时存储开销大

   监听法：

   > 优点：
   >
   > 保证cache一致性
   >
   > 缺点：
   >
   > 总线竞争，不易拓展

2. Tomasulo算法相比Score Board算法有什么异同？（简要回答两点：1.分别解决了什么相关，2.分别是分布式还是集中式）（参考第五版教材）

   Tomasulo算法：

   > 解决WAW、RAW、WAR、结构相关
   >
   > 分布式

   Score Board算法：

   > 解决WAR、WAW、结构相关
   >
   > 集中式

3. Tomasulo算法是如何解决结构、RAW、WAR和WAW相关的？（参考第五版教材）

结构相关：有结构冲突时不发射

RAW相关：只在取操作数阶段读数，只在不冲突时读

WAR、WAW相关：使用 RS 中的寄存器值或指向 RS 的指针代替指令中的寄存器-寄存器重命名