Lab6 实验报告

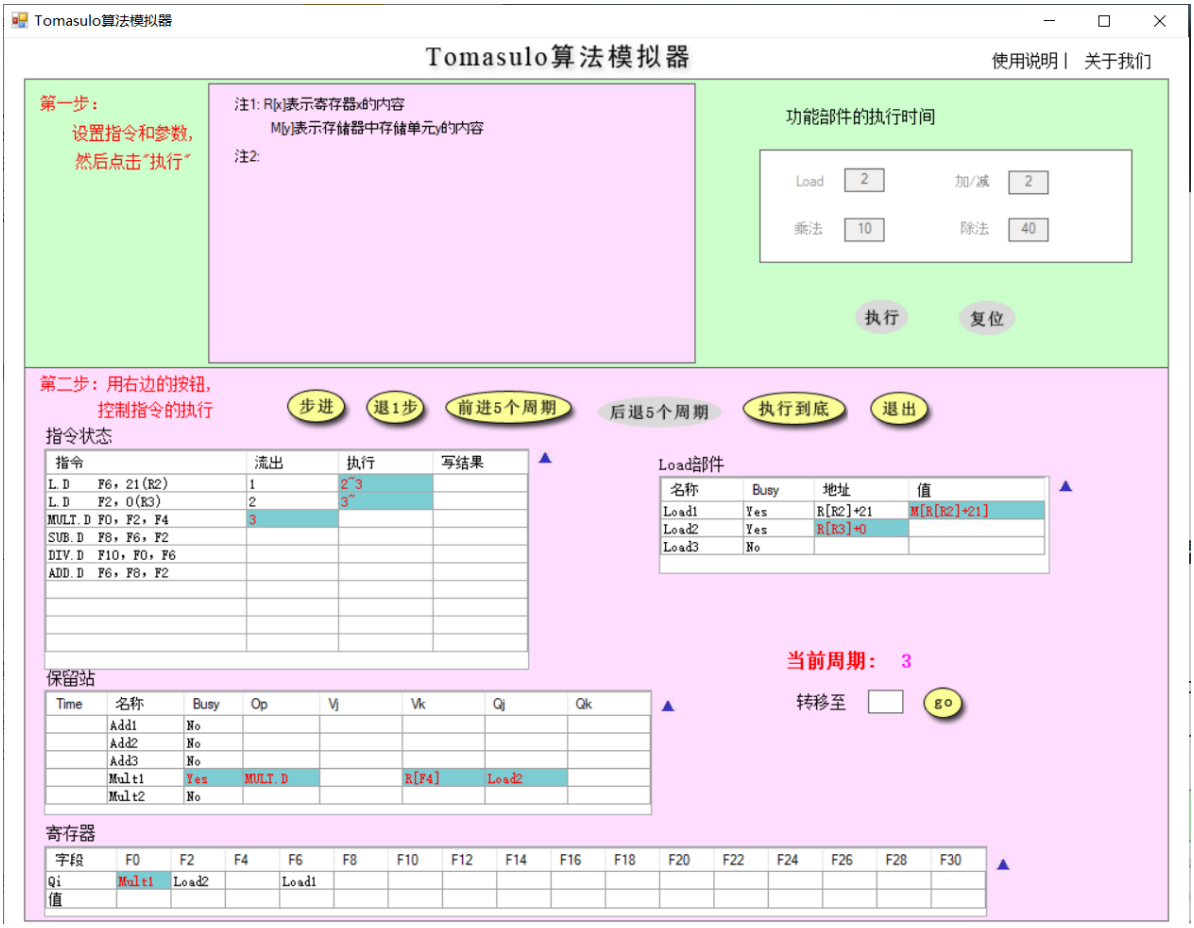
PB20000072 王铖潇

一、Tomasulo算法模拟器

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的截图如下：

周期3的截图如下：

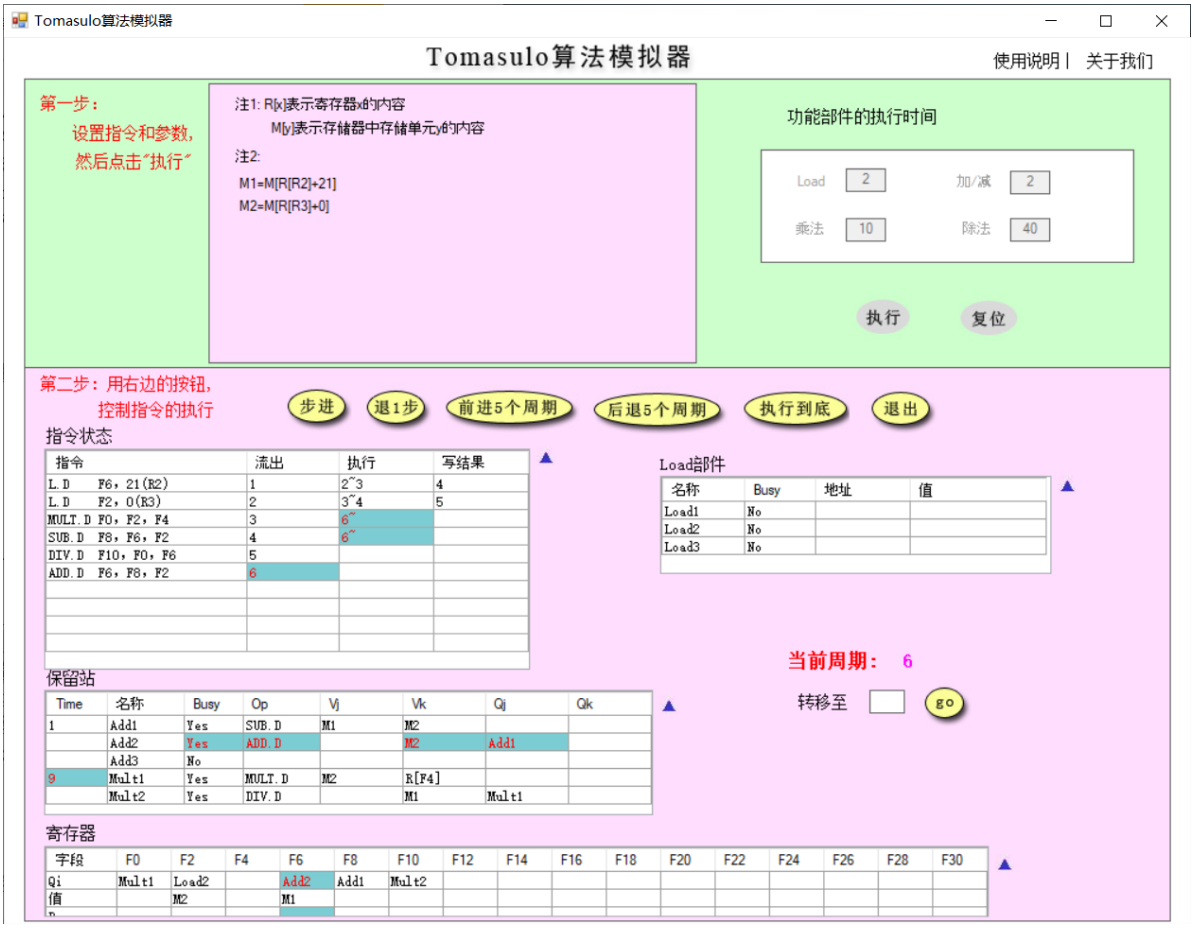


周期2：Load1和Load2部件的状态设置成Busy，Load1地址设置成需要取的值的地址。

周期3：Load1的值设置成Load指令取到了的内存中的值。Load2地址设置成需要取的值的地址。

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）

`MUL.D` 刚开始执行时系统状态截图如下：



发生的改动有：

1. 指令状态：`ADD.D`指令流出，`MUL.D`和`SUB.D`都开始执行。
2. 保留站：

   由于`MUL.D`开始执行，`Mult1`部件的`Time`值设置成9，说明还需要9个周期。

   由于`ADD.D`指令流出，`Add2`部件的`Busy`值设置成`Yes`，操作是`ADD.D`。第一个操作数不可用，需要等待`Add1`的结果，因此`Qj`是`Add1`；第二个操作数可用，是`M2`，从寄存器表可以看到`M2`是之前执行的`Load2`的结果。
3. 寄存器：

   由于`ADD.D`流出，`F6`是它的目的寄存器，所以`F6`的`Busy`字段设置成`Yes`。这条指令使用的`Add2`部件会把输出结果放在F6中，所以F6的`Qi`字段设置成`Add2`。
4. Load部件：

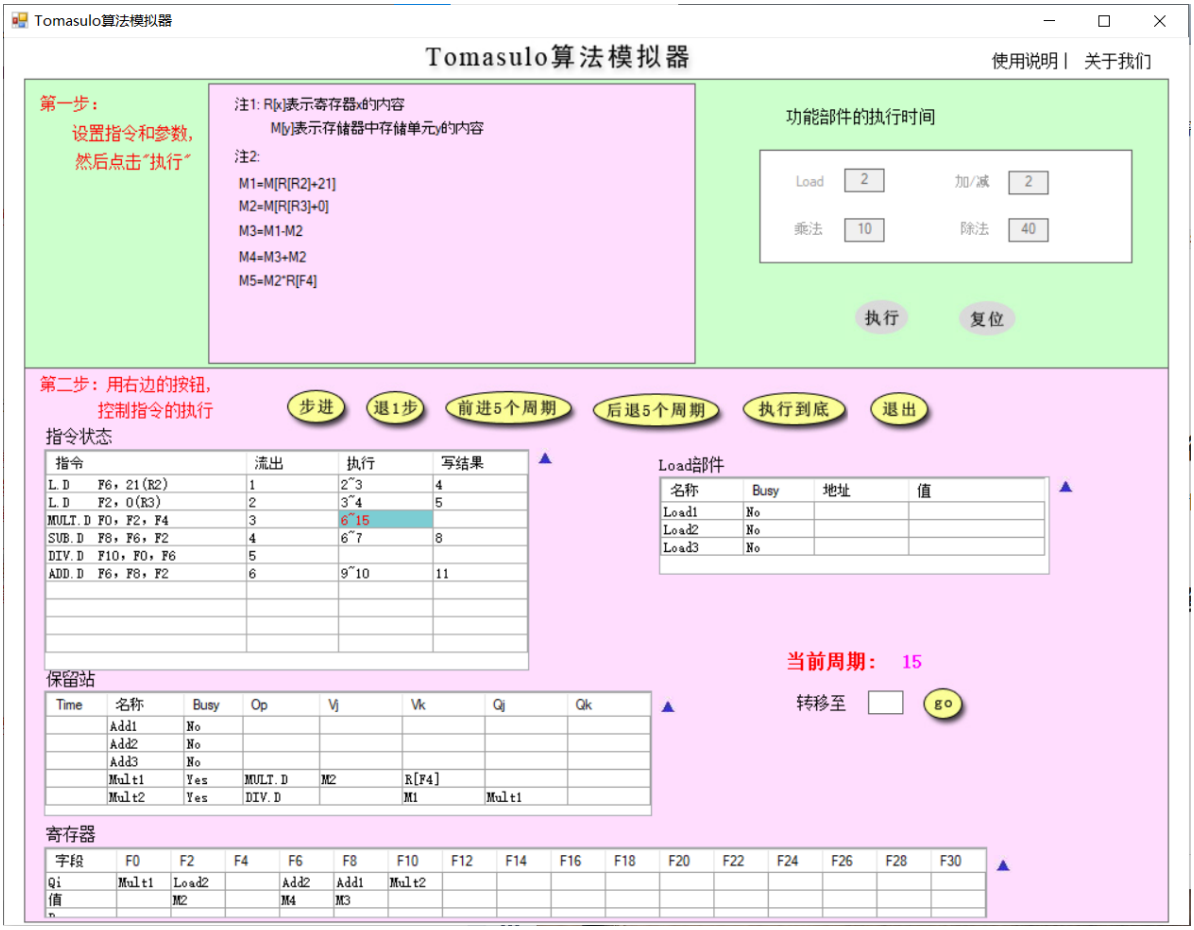
   无变化。

简要说明是什么相关导致MUL.D流出后没有立即执行

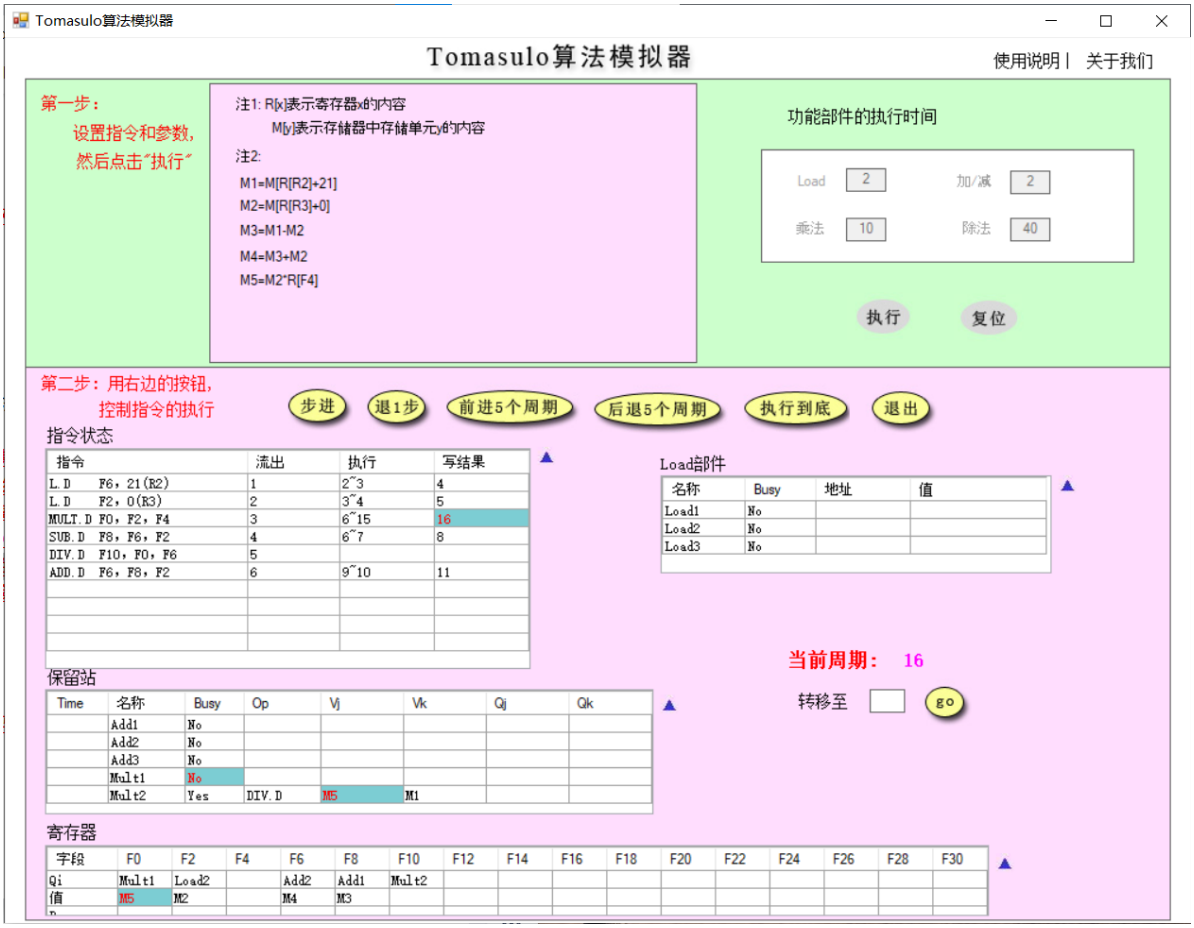
数据相关。`MUL.D`执行需要用到源寄存器`F2`里的值，但是它在第5个周期前都还没有Load出来。

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

第15周期系统状态如下：



第16周期系统状态如下：



变化：

- 第15周期：`MUL.D`指令执行完毕。
- 第16周期：

  指令状态：`MUL.D`结果写回；

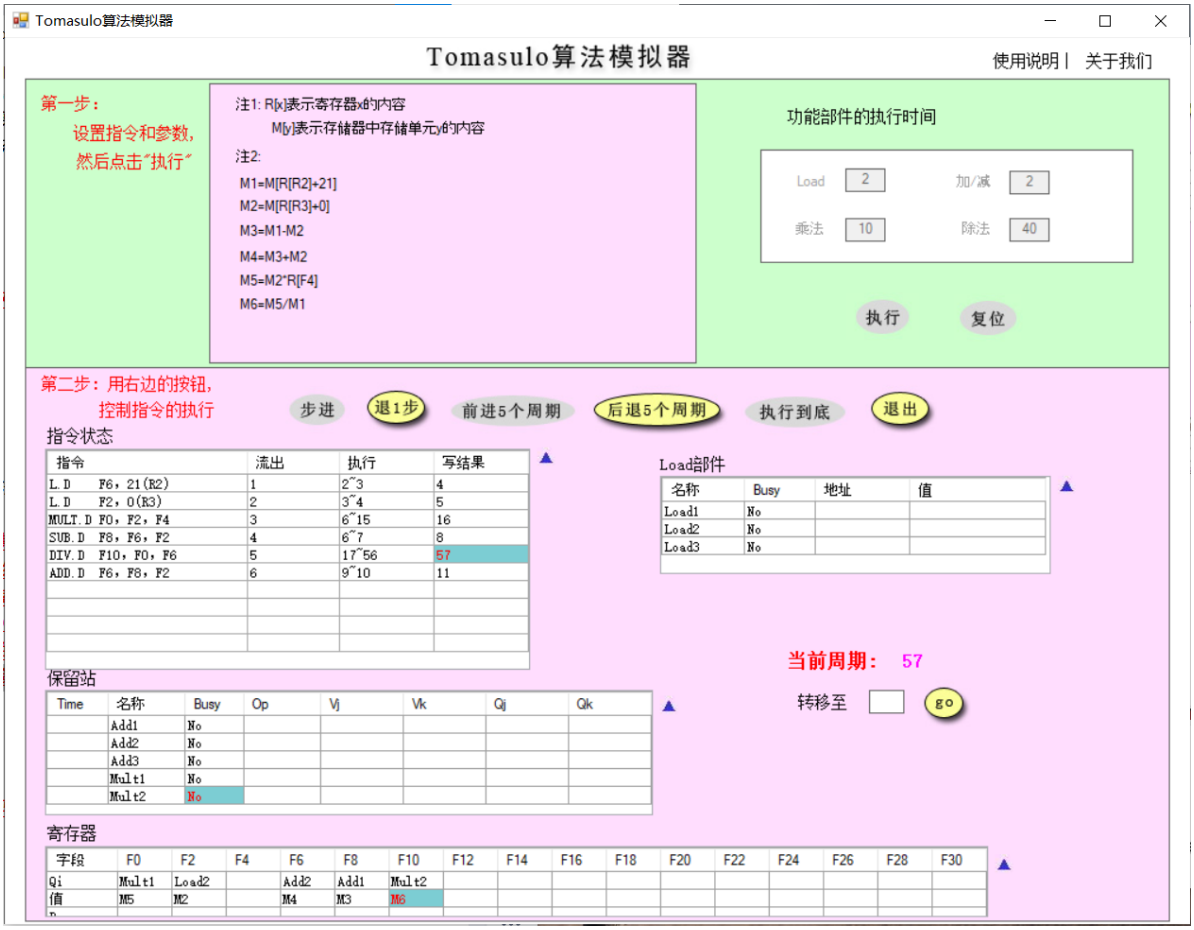
  保留站：`Mult1`的`Busy`设置成`No`；`Mult2`的第一个寄存器可用，值是`M5`。

  寄存器：`Mult1`的结果值标记为`M5`，记录在`F0`的`值`字段中。

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。

截图如下：



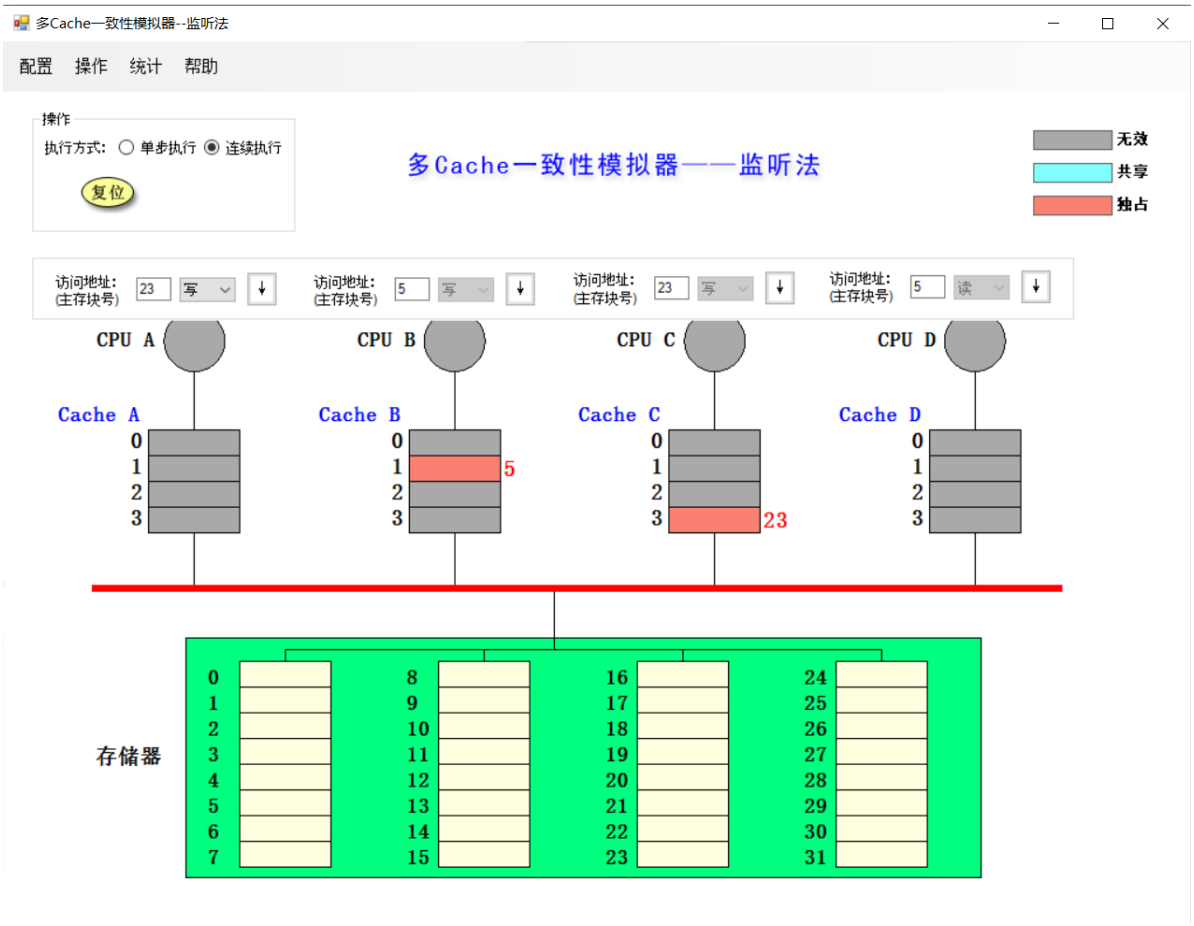
二、 多cache一致性算法-监听法

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	块状态改变
CPUA 读第 5 块	是，替换 CacheA 的块 1	否	CacheA给Bus传递读不命中信号 存储器把第5块传给CacheA	CacheA 第1块：I -> S
CPUB 读第 5 块	是，替换 CacheB 的块 1	否	CacheB 给Bus传递读不命中信号 存储器把第5块传给CacheB	CacheB 第1块：I -> S
CPUC 读第 5 块	是，替换 CacheC 的块 1	否	CacheC 给Bus传递读不命中信号 存储器把第5块传给CacheC	CacheC 第1块：I -> S
CPUB 写第 5 块	否	否	CacheB 给Bus传递作废信号	CacheA 第1块：S -> I CacheC 第1块：S -> I CacheB 第1块：S -> M
CPUD 读第 5 块	替换 CacheD 的块 1	是，CacheB 的块 1 写回	CacheD 给Bus传递读不命中信号 CacheB 写回第 5 块 存储器把第5块传给CacheD	CacheB 第1块：M -> S CacheD 第1块：I -> S
CPUB 写第 21 块	替换 CacheB 的块 1	否	CacheB 给Bus传递写不命中信号 存储器把第21块传给CacheB	CacheB 第1块：S -> M
CPUA 写第 23 块	替换 CacheA 的块 3	否	CacheA 给Bus传递写不命中信号 存储器把第21块传给CacheA	CacheA 第3块：S -> M
CPUC 写第 23 块	替换 CacheC 的块 3	是，CacheA 的块 3 写回	CacheC 给Bus传递写不命中信号 CacheA 写回第 23 块 存储器把第23块传给CacheC	CacheA 第3块：M -> S CacheC 第3块：I -> M
CPUB 读第 29 块	替换 CacheB 的块 1	是，CacheB 的块 1 写回	CacheB 写回第 21 块 CacheB 给Bus传递读不命中信号 存储器把第29块传给CacheB	CacheB 第1块：M -> S

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	块状态改变
CPUB 写第 5 块	替换 CacheB 的块 1	否	CacheB 给Bus传递写不命中信号 存储器把第5块传给CacheB	CacheB 第1块：S -> M CacheD B 第1块：S -> I

截图展示执行完以上操作后整个cache系统的状态



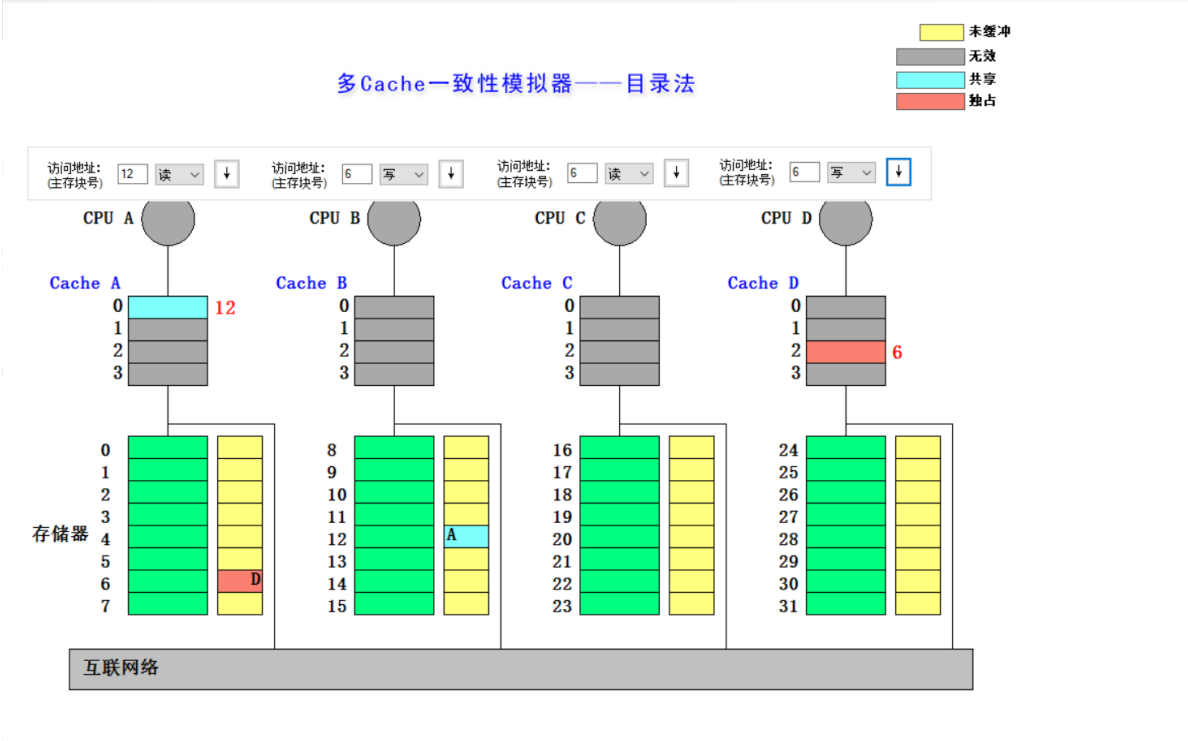
多cache一致性算法-目录法

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	块状态改变
CPUA 读第 6 块	CacheA 向宿主结点发送读不命中(A, 6)消息 宿主结点把块发送给本地结点	CacheA 第2块：I -> S 存储器A 第6块：U -> S 共享集合：{A}
CPUB 读第 6 块	CacheB 向宿主结点发送读不命中(B, 6)消息 宿主结点把块发送给本地结点	CacheB 第2块：I -> S 共享集合：{A, B}
CPUD 读第 6 块	CacheD 向宿主结点发送读不命中(D, 6)消息 宿主结点把块发送给本地结点	CacheD 第2块：I -> S 共享集合：{A, B, D}
CPUB 写第 6 块	CacheB 向宿主结点发送写(B, 6)命中消息 宿主向远程结点 Cache A 和 Cache D 发送作废数据块(6)的消息CacheA 第2块：I -> S	CacheA 第2块：S -> I CacheD 第2块：S -> I CacheB 第2块：S -> M 存储器A 第6块：S -> M 共享集合：{B}
CPUC 读第 6 块	CacheC 向宿主结点发送读不命中(C, 6)消息 给远程结点CacheB发取数据块(6)的消息 远程结点把数据块(6)传递给宿主结点 宿主结点把数据块(6)传递给本地结点	CacheB 第2块：M -> S CacheC 第2块：I -> S 存储器A 第6块：M -> S 共享集合：{B, C}
CPUD 写第 20 块	CacheD 向宿主结点发送写不命中(D, 20)消息 宿主把数据块送给本地结点	CacheD 第0块：I -> M 存储器C 第20块：U -> M 共享集合：{D}
CPUA 写第 20 块	CacheA 向宿主结点发送写不命中(A, 20)消息 宿主结点向远程结点发送取并作废(20) 的消息 远程结点把数据块送给宿主结点，并作废对应块 宿主把数据块送给本地结点	CacheD 第0块：M -> I CacheA 第0块：I -> M 共享集合：{A}

所进行的访问	监听协议进行的操作	块状态改变
CPUD 写第 6 块	CacheD向宿主结点发送写不命中(D,6)消息 宿主结点向远程结点CacheB与 CacheC发送作废(6)消息 远程结点把数据块送给宿主结点并将将对应块作废 宿主结点把数据块送给本地结点	CacheB 第2块：S -> I CacheD 第2块：I -> M 存储器A 第6块：S -> M 共享集合：{D}
CPUA 读第 12 块	CacheA向被替换块的宿主结点发送写回并修改共享集(A, 20)消息 CacheA向宿主结点发送读不命中(A, 12)消息 宿主结点把数据块送给本地结点	CacheA 第0块：M -> I CacheA 第0块：I -> S 存储器C 第20块：M -> U 共享集合：{}； 存储器B 第12块：U -> S 共享集合：{A}

截图展示执行完以上操作后整个cache系统的状态



综合问答

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

1. 目录法
    - 优点：
        - 可以支持大型体系，可扩展性好。
    - 缺点：
        - 存储器接口通信压力大，存储器速度会造成限制。并且当处理器数目增多的时候，目录的存储开销会变得很大。
2. 监听法
    - 优点：
        - 在小型体系中，总线压力较小，成本低。
    - 缺点：
        - 总线的可扩放性收到一定限制：总线上能够连接的处理器数目有限，共享总线存在竞争使用问题，在由大量处理器构成的多处理器系统中监听带宽是瓶颈。
        - 在非总线或环的网络上监听是比较困难的：必须将一致性相关信息广播到所有处理器，这是比较低效的。

Tomasulo算法相比Score Board算法有什么异同

1. 分别解决了什么相关：

   都解决了三种相关，但是方式不完全相同。
    - 结构相关：有结构冲突时指令不流出。
    - RAW相关：操作数可用时才进入执行阶段。
    - WAR相关和WAW相关：Tomasulo通过寄存器重命名解决，Score Board通过流水线停顿等待的方式解决。
2. 分别是分布式还是集中式：

   Tomasulo是分布式

   Score Board是集中式

Tomasulo算法是如何解决结构、RAW、WAR和WAW相关的

1. 结构相关

   有结构冲突时指令不流出。
2. RAW相关

   操作数可用时才进入执行阶段，否则等待需要的操作数的结果算出来。
3. WAR相关

   寄存器重命名。
4. WAW相关

   寄存器重命名。

实验总结

一学期的实验终于告一段落，和Verilog的相处也可以暂告一段落了。

从实现最基本的五级流水线到缓存和分支预测的实现，再到最后对GPU的Cuda编程和对Tomasulo和Cache一致性的理解，我对计算机体系结构有了更深入的理解。曾经最令我头大的Verilog编程这学期看起来也逐渐清晰可爱。之前被数电实验和计算机组成原理实验折磨的时候一直说自己以后坚决不会做系统方向的研究，经过一学期的实验这一点态度似乎也有所改观，突然觉得体系结构也很有意思，或许以后也可以尝试着向这个方向做一做呢。

谢谢助教一学期以来的辛苦付出！~~如果没有助教给出的友好的实验框架，我对Verilog的态度大概毕业之前都不会改变。~~