Lab5 Tomasulo和cache一致性模拟器使用

5.1 Tomasulo算法模拟器

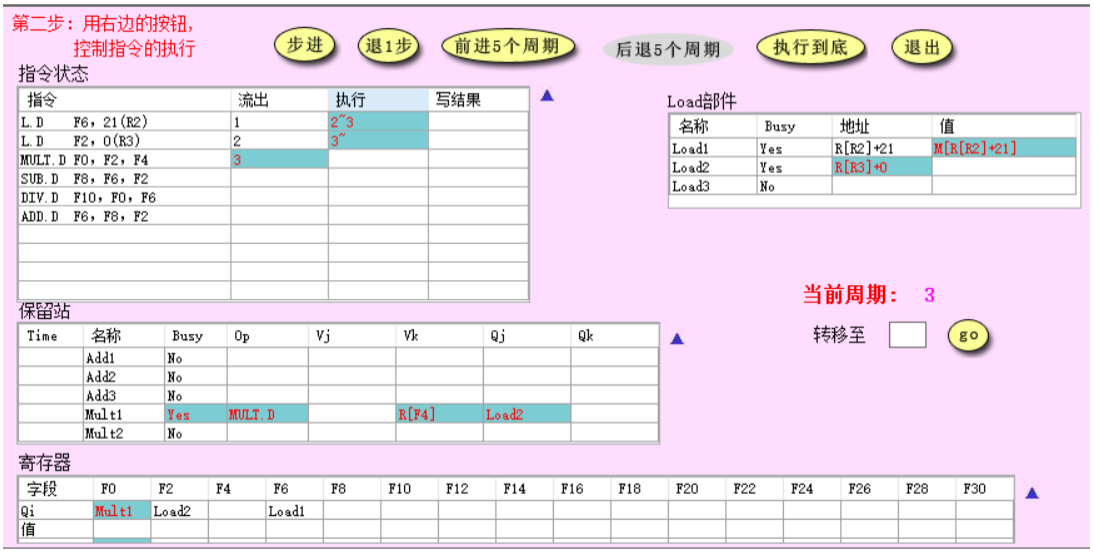
使用Tomasulo模拟器执行以下指令

```
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
```

1.截图: 当前周期2和当前周期3

- 当前周期2

- 当前周期3



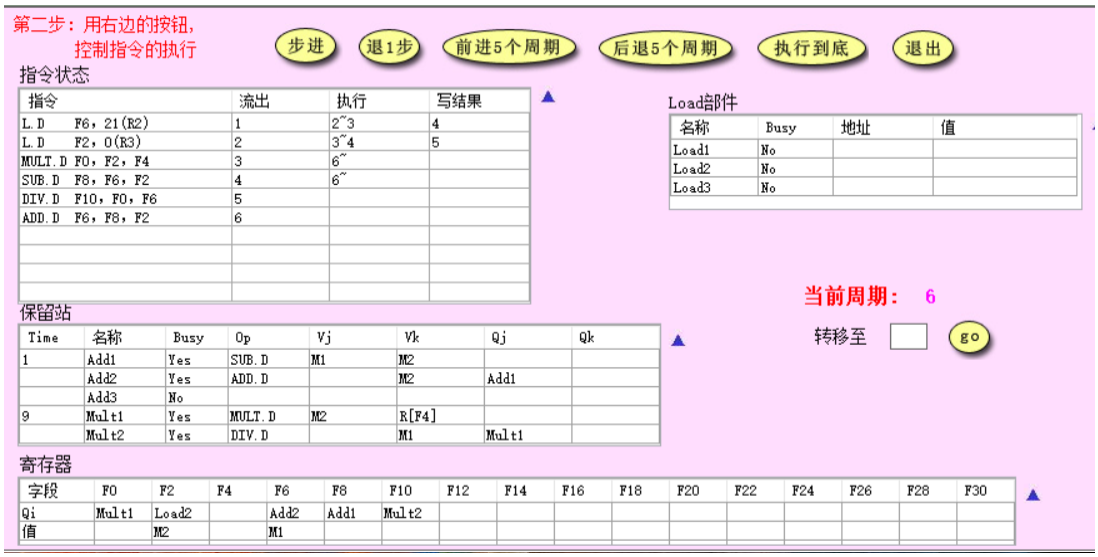
- 简要说明 load 部件做了什么改动

  第一个load指令执行完毕，得到结果，保存在缓冲区还没有在总线上广播

  第二个load指令开始执行，得到储存器地址

2.截图: MUL.D 刚开始执行时系统状态

- MUL.D 刚开始执行



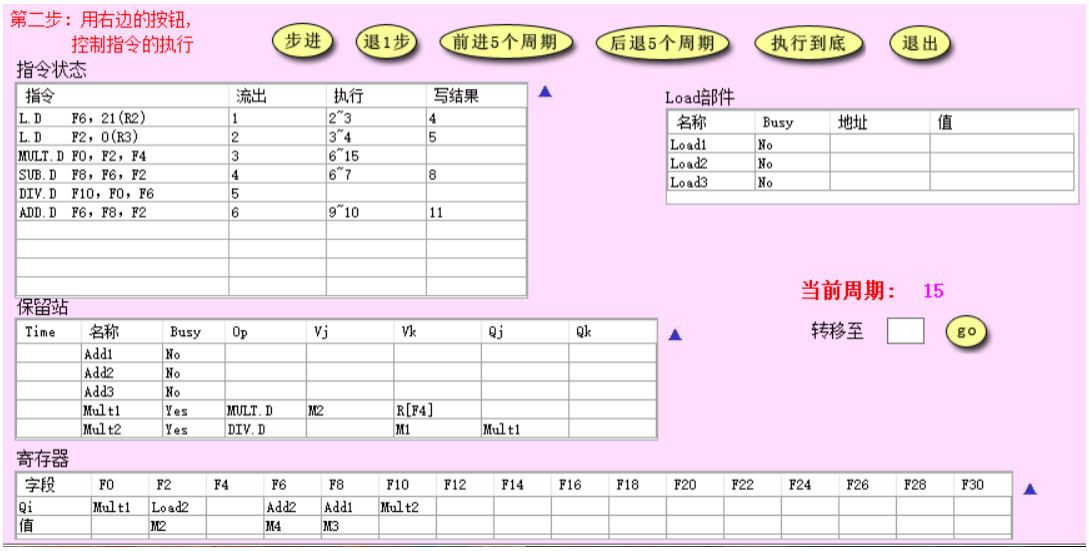
- 说明该周期相比上一周期整个系统发生了哪些改动
  - 指令状态: MULT.D与SUB.D开始执行，ADD.D流出
  - 保留站: 因为ADD.D指令流出, 加减保留站的2号位置Add2处增添这条指令，相应状态位发生变化。Busy=Yes, Op=ADD.D, 但是Vj未知，由Qj=Add1确定，Vk=M2
  - 寄存器: 因为新流出的指令ADD.D修改了F6寄存器，故将寄存器F6的目标值获取地址设置为保留站Add2
  - Load部件: 无变化，因为Load指令全部执行完毕

3.是什么相关导致 MUL.D 流出后没有立即执行

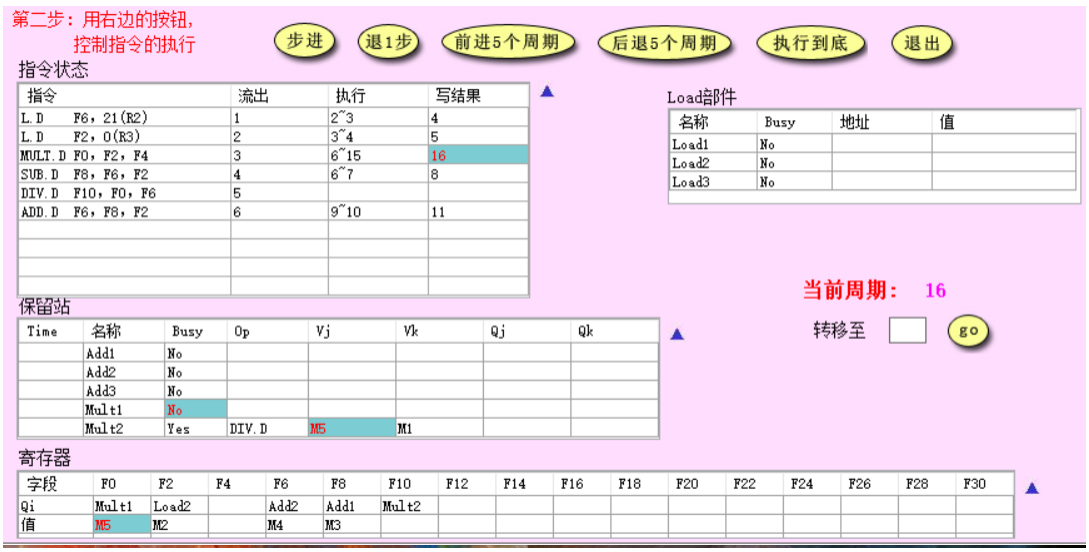
- 是RAW相关，因为MUL.D指令的其中一个操作数在F2寄存器中，而F2寄存器需要先使用Load指令从主存加载，所以MUL.D指令需要等待F2寄存器中值可用之后才能执行

4.截图: 15 周期和 16 周期的系统状态

- 15周期



- 16周期



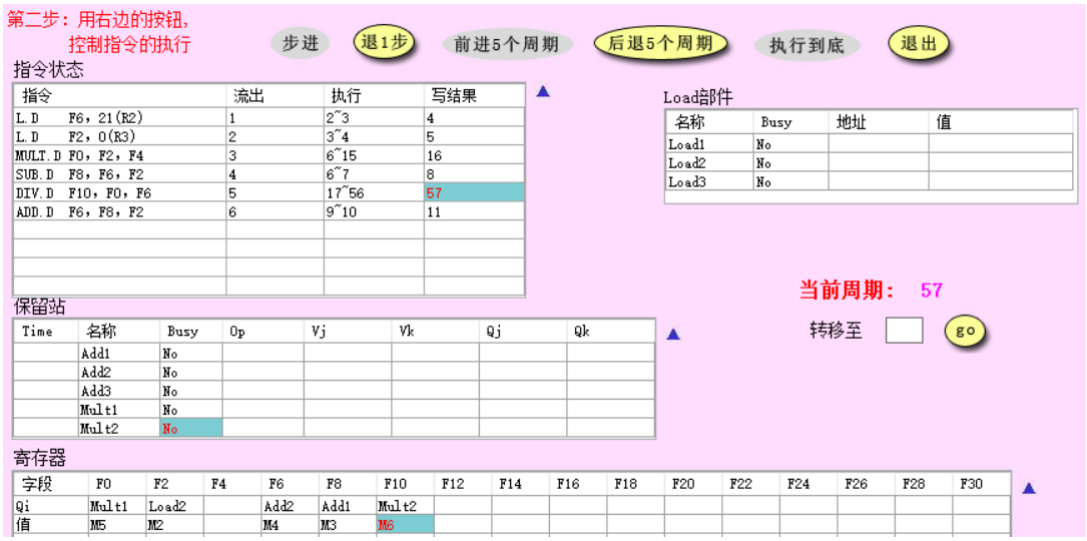
- 分析15-16周期系统发生了哪些变化
  - 指令状态: MULT.D指令执行完毕，写结果
  - 保留站: Mult1的Busy变成No; Mult2的Vj值得到(M5)
  - 寄存器: F0寄存器的值得到为M5
  - Load部件: 无变化，因为Load指令全部执行完毕

5.截图: 所有指令执行完毕

- 所有指令刚刚执行完毕时是第多少周期

  第57周期

- 第57周期截图

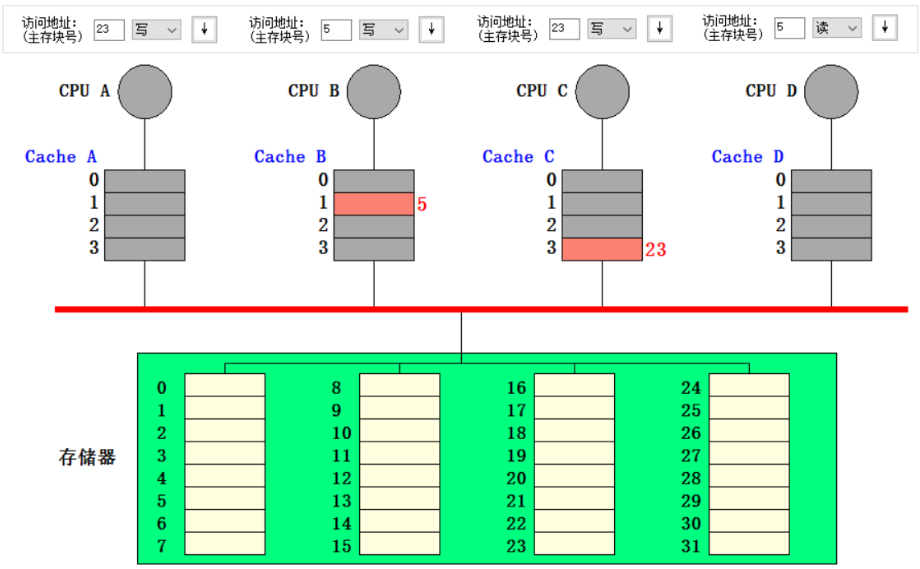


5.2 多cache一致性算法-监听法

1.利用模拟器进行如下操作，并填表

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	否	否	向总线发读不命中，从mem读取，共享
CPU B 读第 5 块	否	否	向总线发读不命中，从mem读取，共享
CPU C 读第 5 块	否	否	向总线发读不命中，从mem读取，共享
CPU B 写第 5 块	否	否	CPU B 写命中,向总线发作废，其他CPU中的第5块均无效，B的第5块修改为独占
CPU D 读第 5 块	否	Y	向总线发读不命中，B的5写回，变成共享，D从mem中读取
CPU B 写第 21 块	Y	否	向总线发写不命中，从mem读21，替换掉第5块, CPU B 的第21块修改为独占, 并对其写入
CPU A 写第 23 块	否	否	向总线发写不命中，从mem读23，CPU A的第23块修改为独占, 并对其写入
CPU C 写第 23 块	否	Y	向总线发写不命中, CPU A的第23块写回mem, CPU C从mem读取第23块, 写后为独占
CPU B 读第 29 块	Y	否	向总线发读不命中，从mem读29，共享
CPU B 写第 5 块	Y	否	向总线发写不命中, 从mem读5,其他的5均失效

2.截图: 执行完以上操作后整个 cache 系统的状态

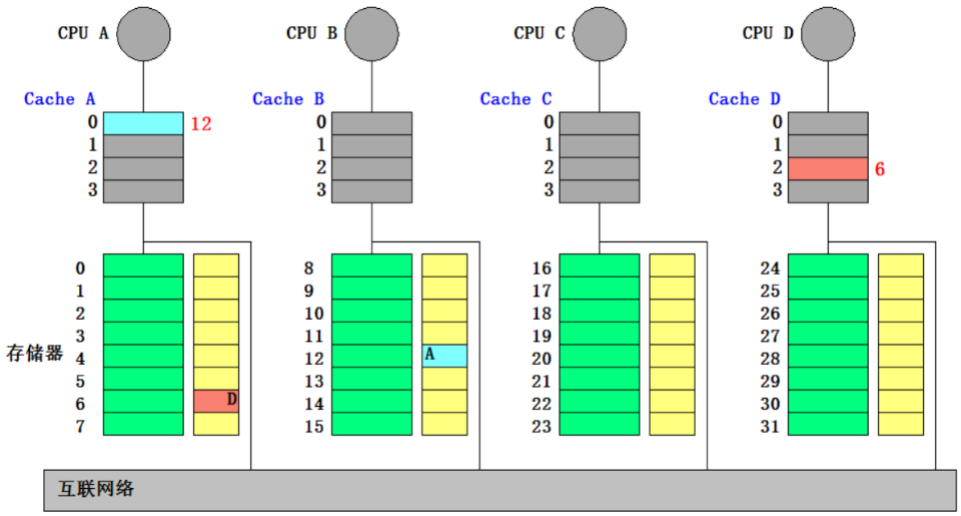


5.3 多cache一致性算法-目录法

1.利用模拟器进行如下操作，并填表

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	读不命中，从本地mem A读取，A目录共享(6,A)
CPU B读第6块	读不命中，从mem A读取, A目录共享(6,AB)
CPU D 读第6块	读不命中，从mem A读取, A目录共享(6,ABD)
CPU B 写第6块	写命中，从A目录获知ABD共享，作废AB cache 6, A目录独占(6,B), B cache 独占6
CPU C 读第6块	读不命中，从A目录获知在cache B, 改成共享，读到C, A目录共享(6,BC)
CPU D写第20块	写不命中，从C中取，C目录改为独占(20,B)
CPU A写第20块	写不命中，从C目录获知B独占，取并作废，C目录改为独占(20,A)
CPU D写第6块	写不命中，从A目录获知BC共享，作废6(BC), 读并独占，A目录改成独占(6,D)
CPU A 读第12块	读不命中，写回并修改共享集(A,20), 读不命中(A,20). 读取

2.截图: 执行完以上操作后整个 cache 系统的状态



5.4 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?

- 目录法拥有一个中央目录控制的角色。原来向总线发送的的消息改为向目录持有者发送消息。而目录持有者负责改变每个处理器上缓存数据的状态。这样可以减少总线的使用。 但实现较复杂。

- 监听法实现方便。但一般只用于共享总线结构,可扩展性较差

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）

	解决了什么相关	**分布式还是集中式**
Tomasulo	用寄存器重命名避免WAR和WAW	分布式
Score Board	用stall来避免WAR和WAW	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

  仅在操作数可用时才执行
- WAR 相关

  用保留站重命名寄存器来避免
- WAW 相关

  用保留站重命名寄存器来避免