计算机体系结构Lab5

PB18071463 朱映 2021.6.7

计算机体系结构Lab5
- 实验目的
- 实验要求
  - 一.Tomasulo算法模拟器
  - 二.多cache一致性算法-监听法
  - 三.多cache一致性算法-目录法
  - 四.综合问答
- 五.收获与建议

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

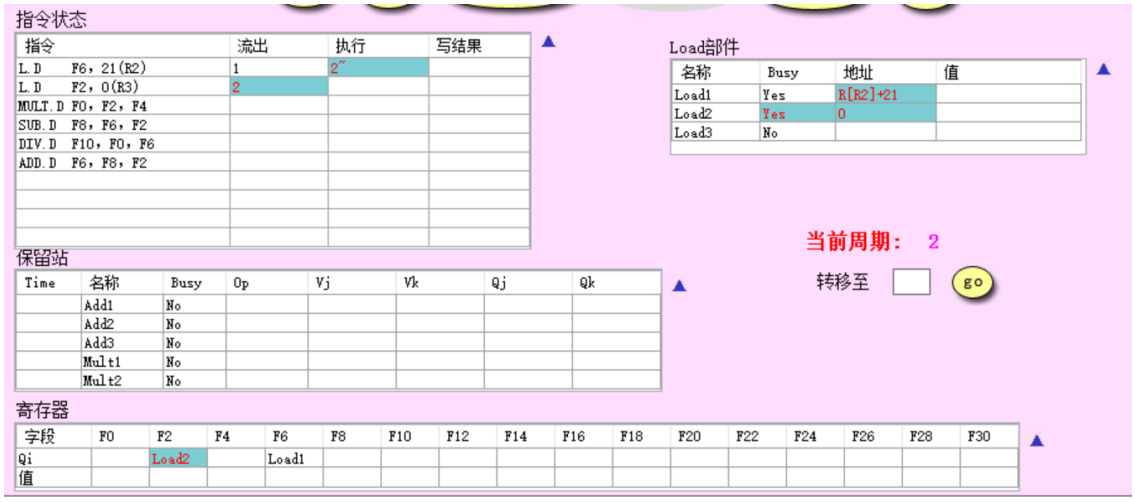
使用模拟器进行以下指令流的执行并对模拟器截图、回答问题

```
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**

   a）周期2：



指令状态

指令	流出	执行	写结果
L.D F6, 21(R2)	1	2~	
L.D F2, 0(R3)	2		
MULT.D F0, F2, F4			
SUB.D F8, F6, F2			
DIV.D F10, F0, F6			
ADD.D F6, F8, F2			

Load部件

名称	Busy	地址	值
Load1	Yes	R[R2]+21	
Load2	Yes	0	
Load3	No		

当前周期： 2

转移至 go

保留站

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

寄存器

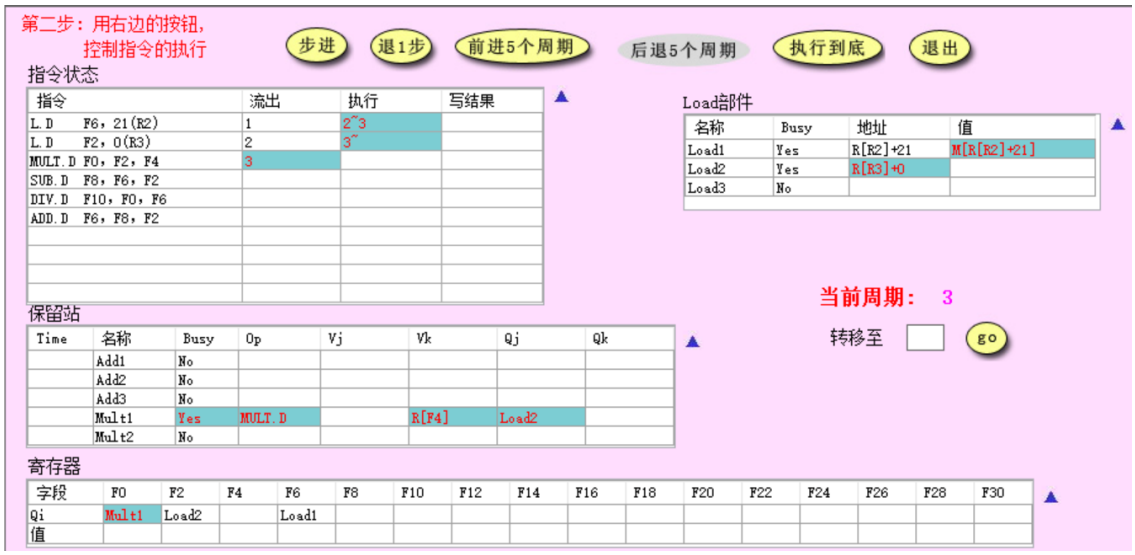
字段	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qi		Load2		Load1												
值																

两条load指令占用两个Load部件。

第一条load指令开始访存，对应的Load1部件变为busy，存储访存地址R[R2]+21 。

第二条load指令暂无动作， 对应的Load2部件变为busy，但并未记录内容。

b) 周期3：



第二步：用右边的按钮，控制指令的执行

步进 退1步 前进5个周期 后退5个周期 执行到底 退出

指令状态

指令	流出	执行	写结果
L.D F6, 21(R2)	1	2~3	
L.D F2, 0(R3)	2	3~	
MULT.D F0, F2, F4	3		
SUB.D F8, F6, F2			
DIV.D F10, F0, F6			
ADD.D F6, F8, F2			

Load部件

名称	Busy	地址	值
Load1	Yes	R[R2]+21	M[R[R2]+21]
Load2	Yes	R[R3]+0	
Load3	No		

当前周期： 3

转移至 go

保留站

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D		R[F4]	Load2	
	Mult2	No					

寄存器

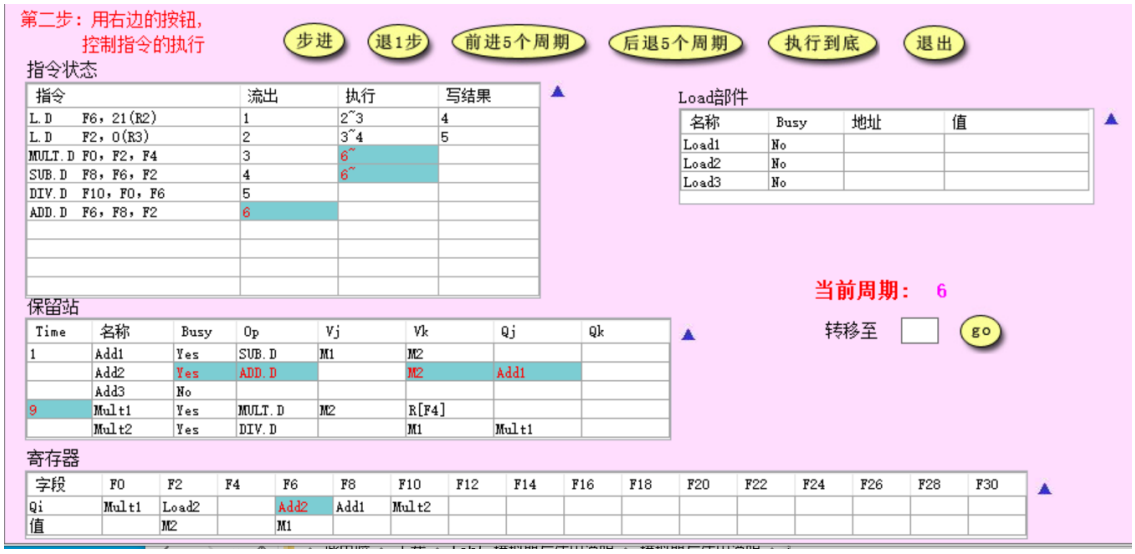
字段	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qi	Mult1	Load2		Load1												
值																

两条load指令占用两个Load部件。

第一条load指令访存完成，Load1部件仍为busy，取出的对应的值存放在Load1部件的值中，值为M[R[R2]+21] 。

第二条load指令开始访存，Load2部件仍为busy，存储访存地址R[R3]+0 。

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）**



MUL.D刚开始执行时的系统状态如上图所示。

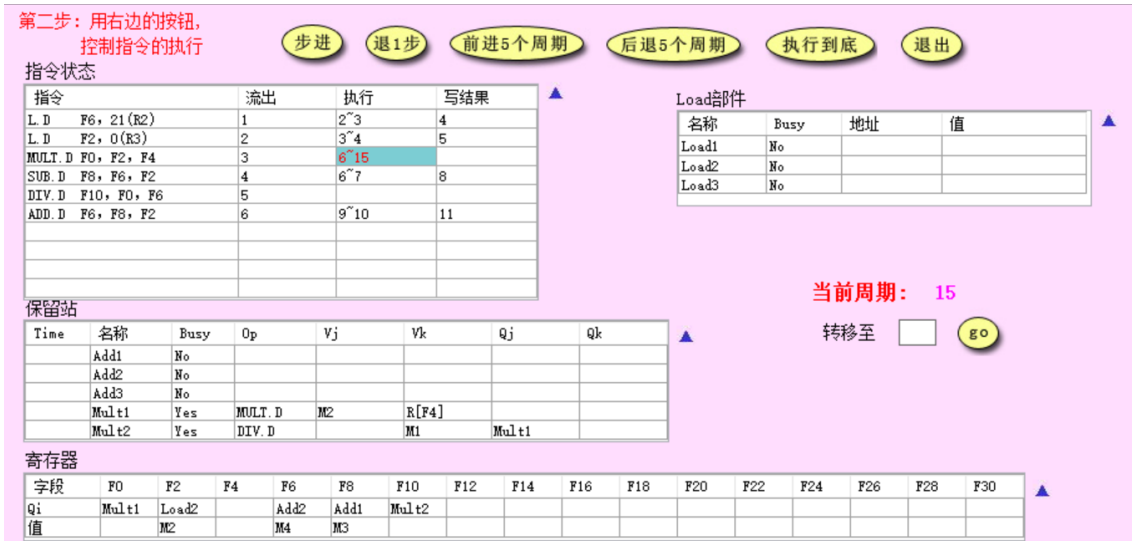
该指令刚开始执行时是第 6 周期，和上一周期相比，变动有：

1. ADD 流出，MULT 和 SUB 都开始执行；
2. 保留站中 SUB 和 MULT 指令开始倒计时，ADD指令占用 Add2 部件，等待 Add1 返回计算结果。
3. ADD 指令指示寄存器 F6 的值将来自于自己，并进行标注。
4. 没有访存，因此 Load 部件没有变化；

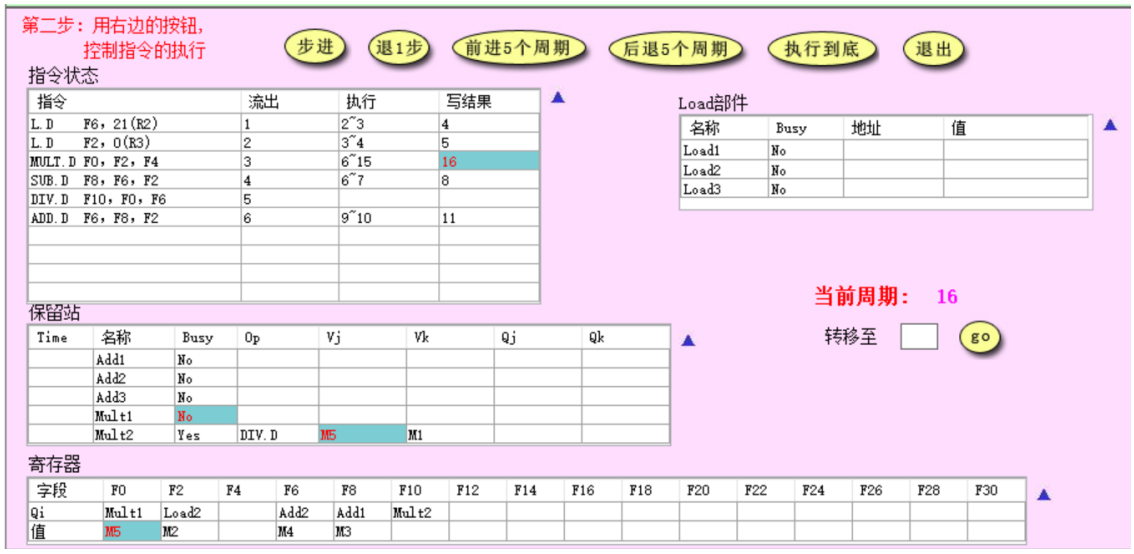
3. **简要说明是什么相关导致MUL.D流出后没有立即执行**

F2 RAW写后读相关导致 MUL 指令需要等待 LOAD 指令写入寄存器后才能继续执行。

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**



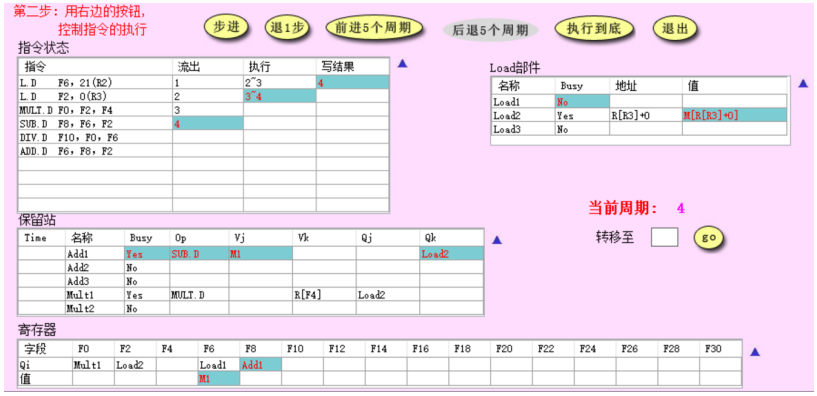
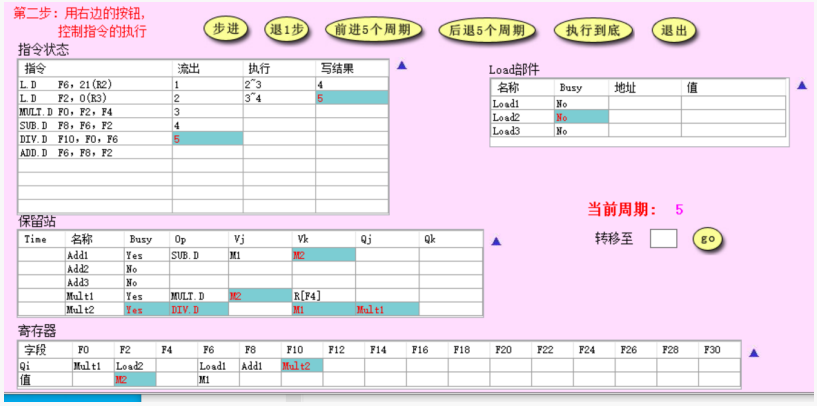
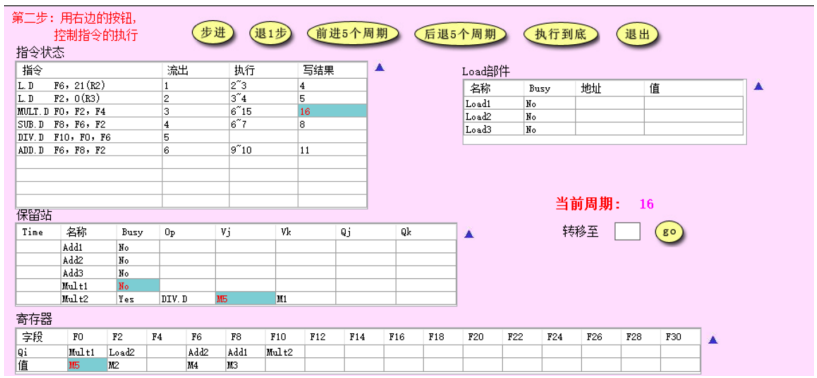
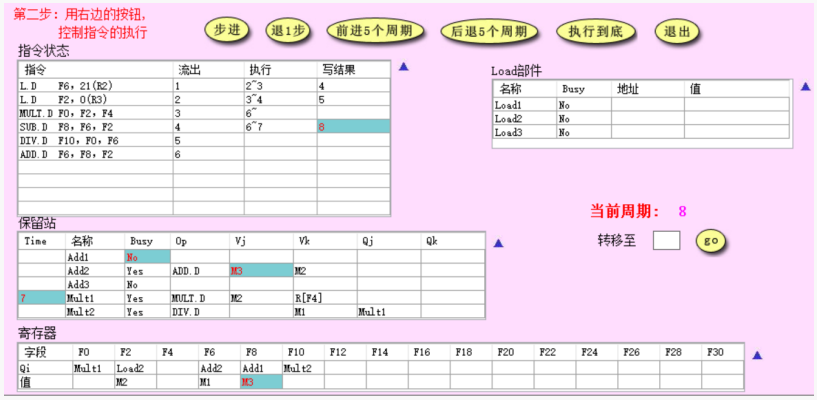
第15周期：MULT.D 指令执行完成。

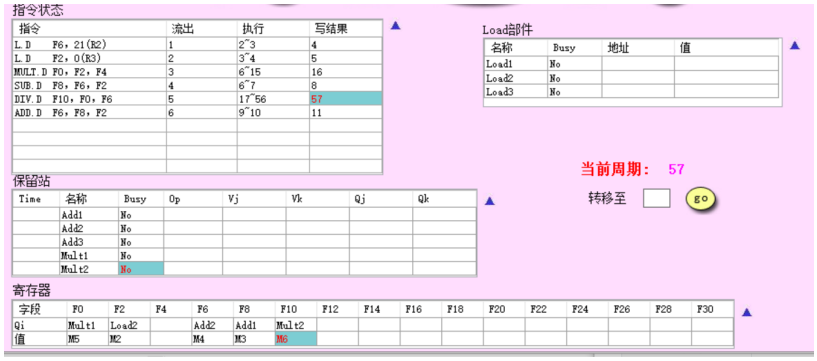
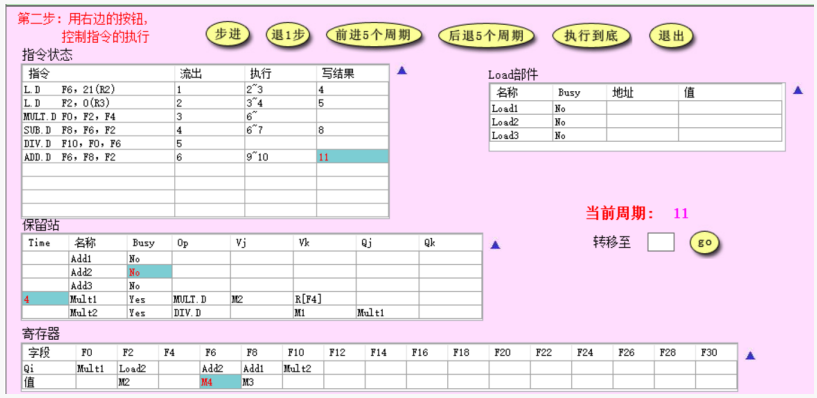


第16周期：

MULT.D指令写结果，保留站Mult1资源被释放，Mult2资源获得来自 MULT 指令的结果 M5 作为第一个操作数，F0 写入值 M5。

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）**

指令	截图	结束周期
L.D F6, 21（R2）		4
L.D F2, 0（R3）		5
MUL.D F0, F2, F4		16
SUB.D F8, F6, F2		8

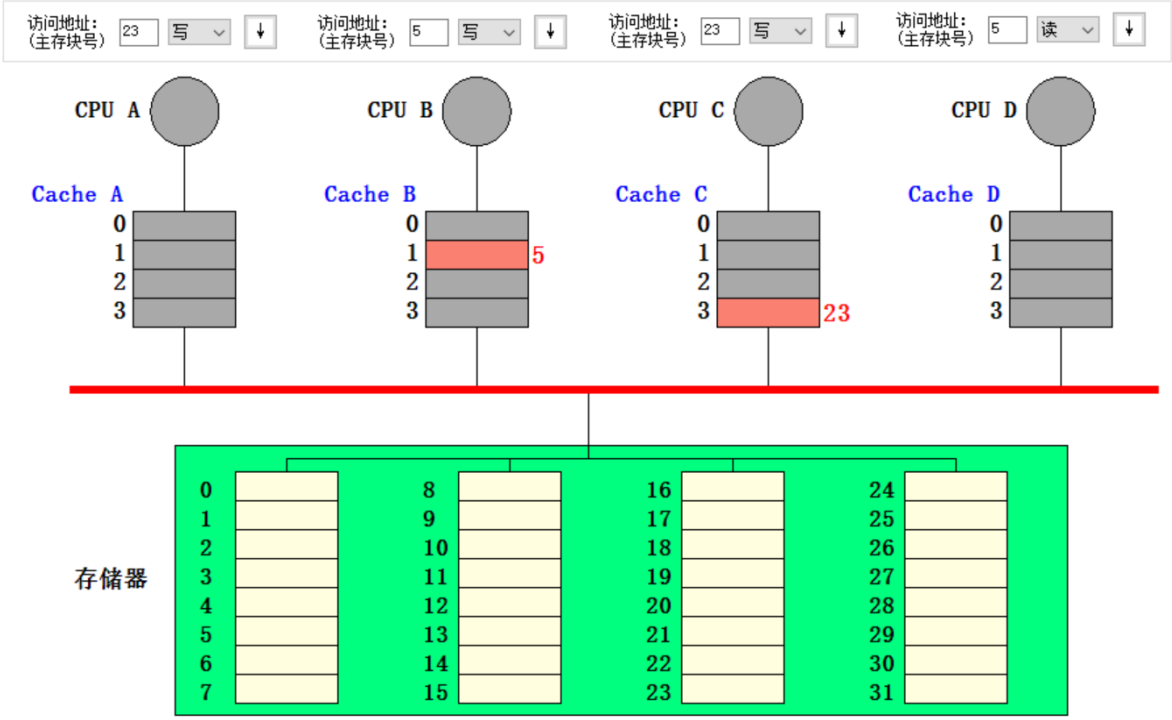
指令	截图	结束周期
DIV.D F10, F0, F6		57
ADD.D F6, F8, F2		11

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	N	N	读不命中，主存第5块进入CPU A 的缓存 1 中，状态为共享
CPU B 读第5块	N	N	读不命中，主存第5块进入CPU B 的缓存 1 中，状态为共享
CPU C 读第5块	N	N	读不命中，主存第5块进入CPU C 的缓存 1 中，状态为共享
CPU B 写第5块	N	N	CPU B 写命中，总线告知其他处理器作废，A 和 C 中1号缓存存放的第5块作废，B 将新值写入第5块，状态为独占
CPU D 读第5块	N	Y	读不命中，CPU B将第5块写回存储器，状态变为共享；第5块从主存进入CPU D的缓存 1 中，状态为共享
CPU B 写第21块	Y	N	写不命中，第 21 块从主存进入 B 1号缓存并作废第五块内容，将新值写入，状态为独占
CPU A 写第23块	N	N	写不命中，第 23 块从主存进入 B 3号缓存，将新值写入，状态为独占
CPU C 写第23块	N	Y	写不命中，第 23 块从CPU A 3号缓存写回主存并作废内容，第23块从主存进入 CPU C 3号缓存并将新值写入，状态为独占
CPU B 读第29块	Y	Y	读不命中，第 21 块从 CPU B 1号缓存写回主存并作废内容，第29块从主存进入 CPU B 1号缓存，状态为共享
CPU B 写第5块	Y	N	写不命中，第 5 块从主存写入 CPU B 1号缓存并作废原有的第 29 块内容，将新值写入，状态为独占；同时 D 中 1号缓存的第五块内容作废

2. 请截图，展示执行完以上操作后整个cache系统的状态

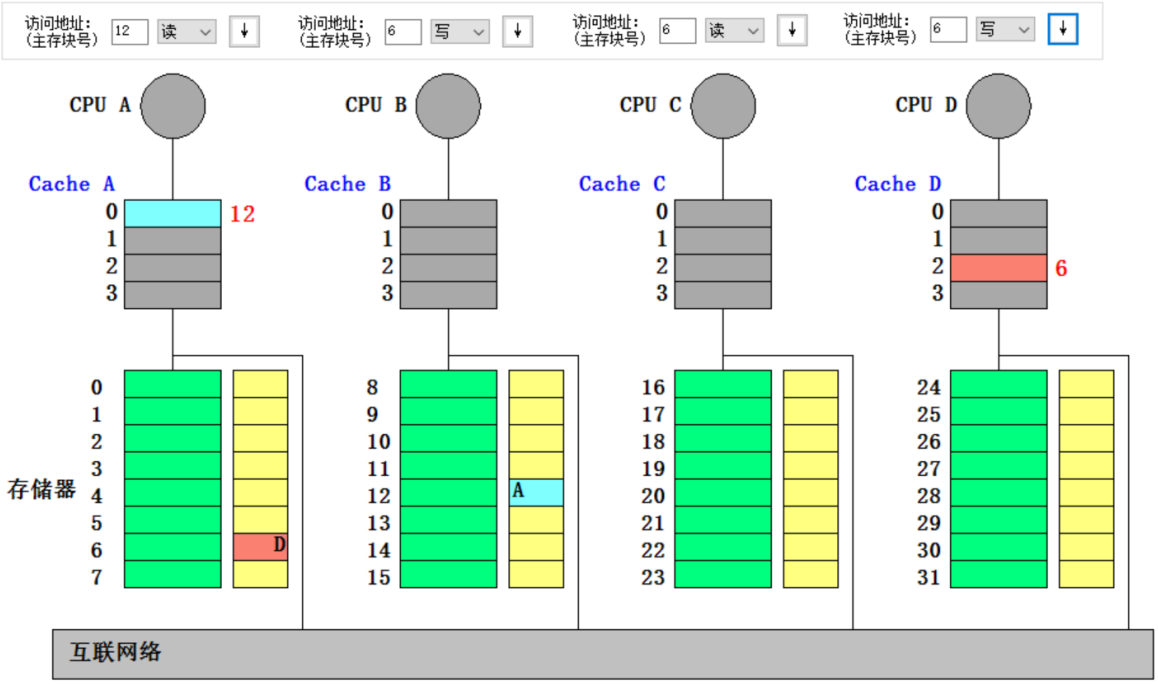


三.多cache一致性算法-目录法

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	A向宿主发送读第六块请求，宿主返回数据并将共享集合设为A，A将数据存入第2号缓存，状态为共享
CPU B 读第6块	B通过网络向宿主发送读第六块请求，宿主返回数据并将共享集合设为AB，B将数据存入第2号缓存，状态为共享
CPU D 读第6块	D向宿主发送读第六块请求，宿主返回数据并将共享集合设为ABD，D将数据存入第2号缓存，状态为共享
CPU B 写第6块	B通过网络向宿主发送第六块写命中请求，宿主向第六块的共享集合中除B外的处理器发送作废命令，共享集合变为B； A、D中2号缓存的第6块变为无效， B将新值写入第二号缓存，状态为独占
CPU C 读第6块	C通过网络向宿主发送读失效请求，宿主向B发送读请求，B中的值进入宿主进行更新，宿主再将第6块的值返回给C，更新共享集合为BC；B、C中2号缓存的第6块状态变为共享
CPU D 写第20块	D通过互联网络向宿主发送写不命中请求，宿主将20块发送给D，更新共享集合为 D；D将块20新值写入0号缓存，状态为独占
CPU A 写第20块	A通过互联网络向宿主发送写不命中请求，宿主向D发送读请求，D中的值进入宿主进行更新，宿主再将第20块的值返回给A，更新共享集合为A；D中0号缓存的第20块状态变为无效，A中0号缓存的20块状态为独占
CPU D 写第6块	D通过网络向宿主发送第六块写不命中请求，宿主向第六块的共享集合中的处理器发送作废命令，共享集合变为D；B、C中2号缓存的第6块变为无效， D将新值写入第2号缓存，状态为独占
CPU A 读第12块	A 第12块应在的位置已经有状态为独占的第20块，所以将第20块写回主存，然后发送读不命中请求；宿主传递第12块的值到 A 中第0号缓存并修改共享集合为A， A可以读取，状态为共享

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）**

   监听法：

   优点是总线交流次数少，通信开销低，对规模较小的体系有优势。

   缺点是CPU数量较多时，需要很频繁的总线工作，压力较大，此时性能会受到较大影响。

   目录法：

   优点是通过宿主保存一部分信息，可以有目的的进行通信，降低了总线压力，对于CPU数量较多的大型体系有优势。

   缺点是操作序列多写多读时宿主与缓存的沟通次数非常多，需要频繁发出作废信号，导致性能下降。

2. **Tomasulo算法相比ScoreBoard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）**

   相同之处：

   两者都通过动态调度的方式来消除RAW相关，避免流水线停顿过久。都解决了结构相关、RAW、WAR 和 WAW。

   不同之处：

   1. Tomasulo是分布式，ScoreBoard是集中式；
   2. 解决 WAW 相关的方式不同。Tomasulo在Issue阶段对寄存器换名；而ScoreBoard 在 Issue 阶段检测是否有其他活动的指令与要发射指令的目的寄存器相同，空闲才发射指令；
   3. 写结果的方式不同。Tomasulo可以直接将结果写入保留站中供其他指令使用，而ScoreBoard必须将结果先写入寄存器，其他指令要使用则要从寄存器中读入数据。

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：当有对应资源可用时才发射指令；

   RAW：指令执行阶段中，如果操作数没有准备好则暂停，一直检测数据总线直到前一条写指令执行完成，得到操作数为止；

   WAR、WAW：寄存器换名。

五.收获与建议

耗时：2h，主要在截图和作答上。

收获：

1. 模拟器动画效果非常直观，学习效率好于简单的 PPT 演示。
2. 通过对指令执行进行观察，了解了 cache 一致性的实现以及 Tomasulo 算法的原理，加深了理解。

建议：

使用的软件版本偏老，希望能有更新版本。