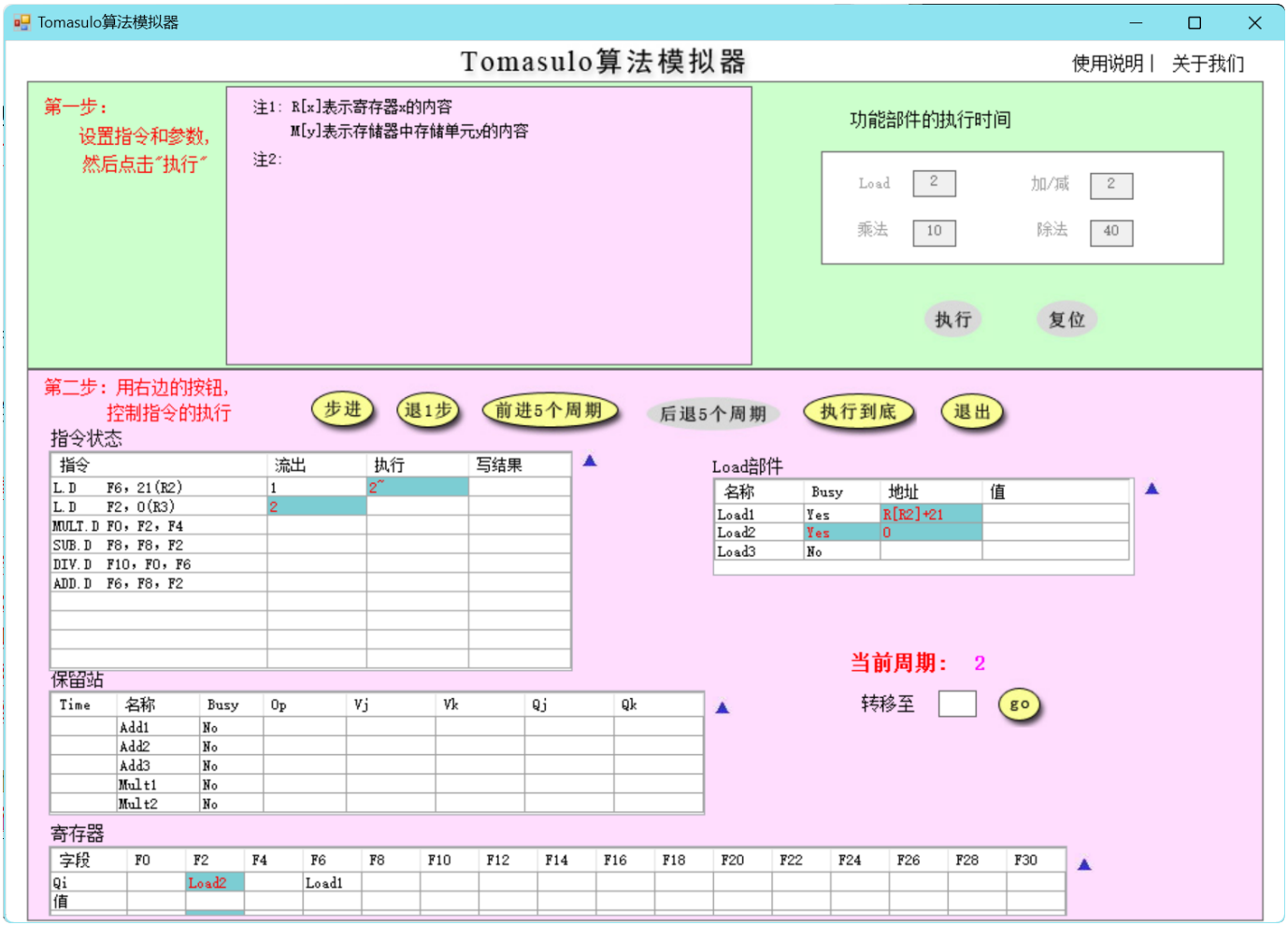
体系结构LAB6

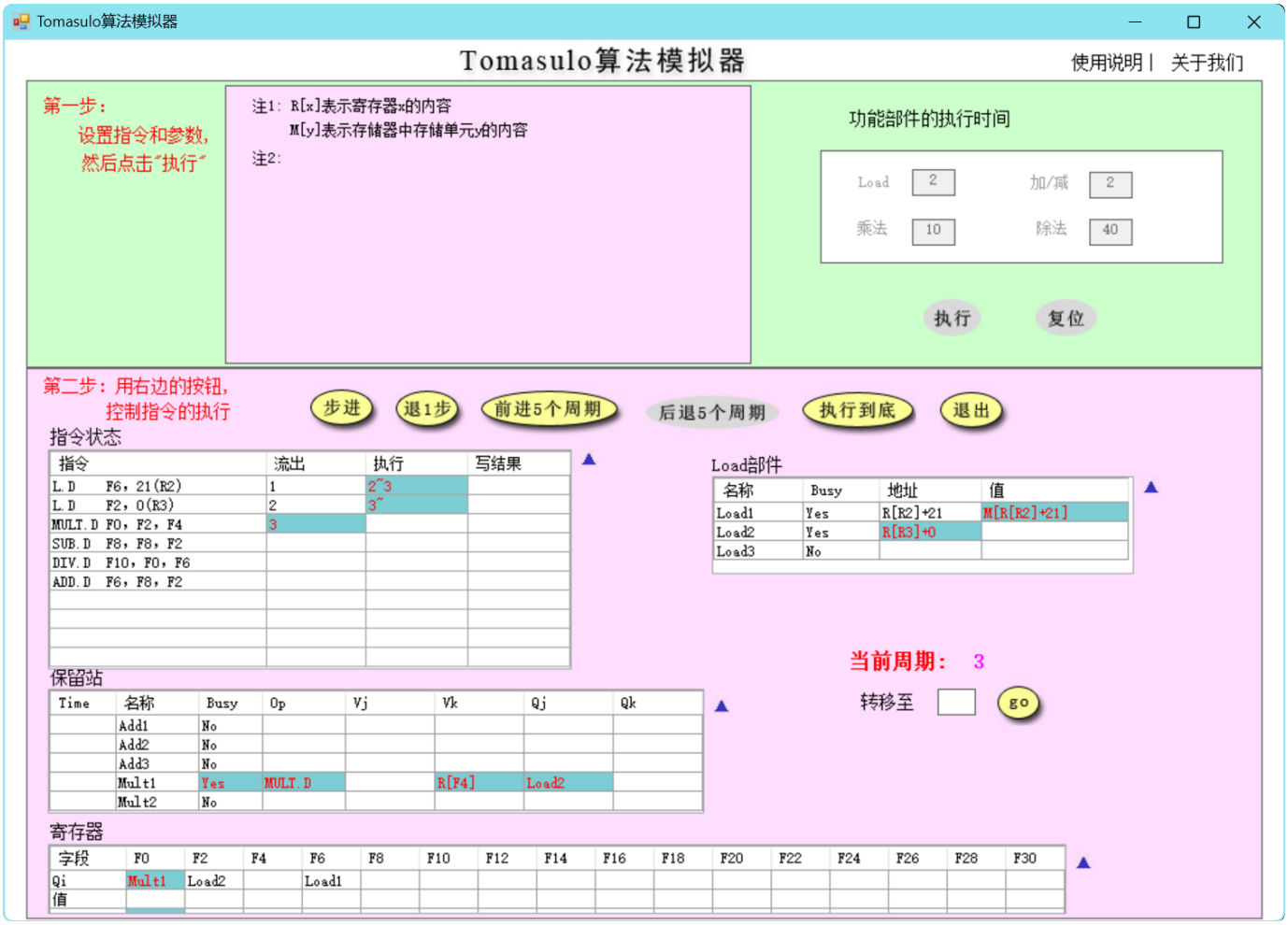
Tomasulo 算法模拟器

1. 当前周期2：



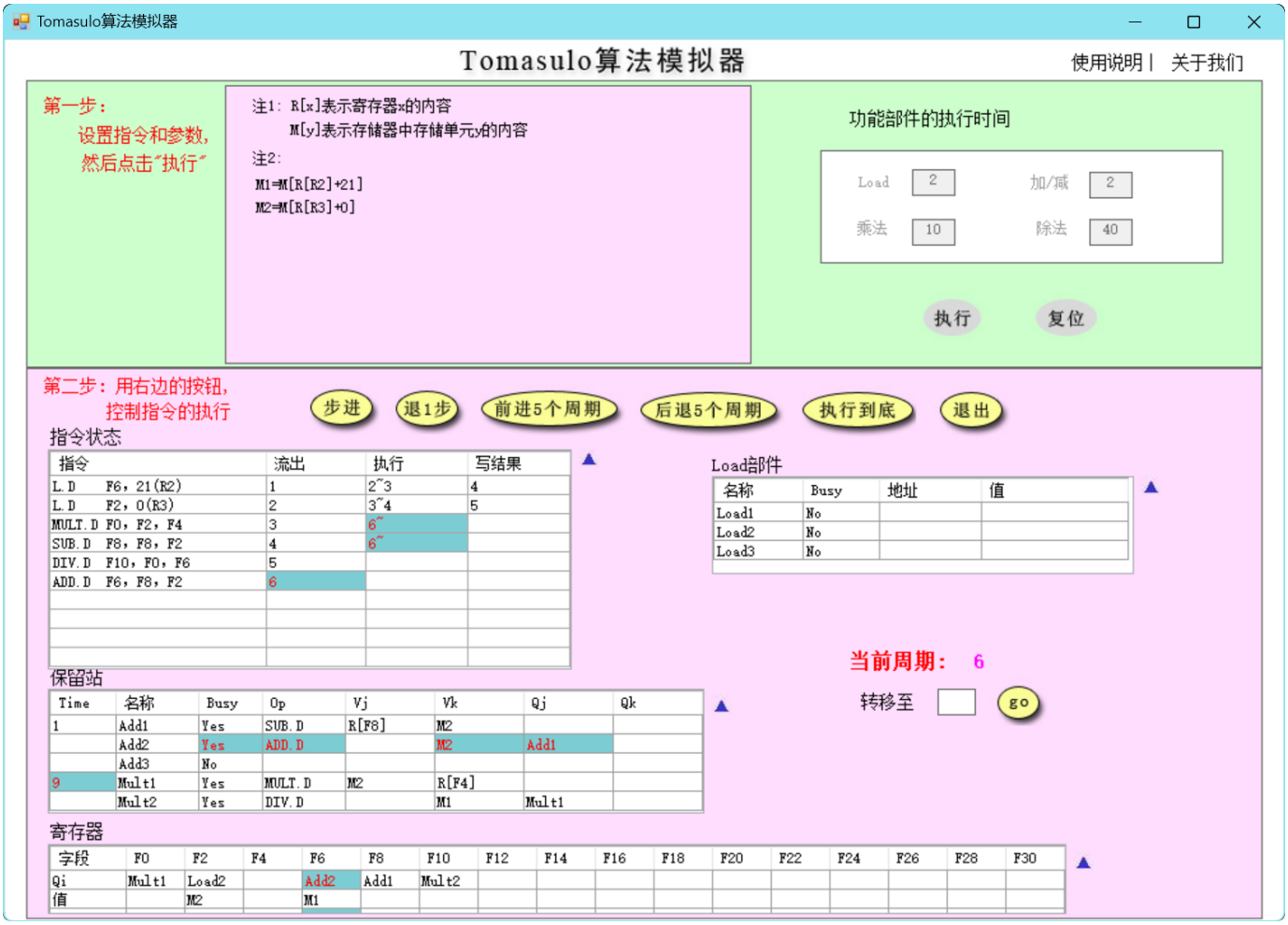
从周期1到周期2：Load1部件地址计算为R[R2]+21；Load2部件变为busy状态，地址暂定为0（下一周期才计算为R[R3]+0）

当前周期为3：



从周期2到周期3：Load1部件地址位置处的值计算出；Load2部件的地址计算得出。

2. 第6周期时，MUL.D刚开始执行



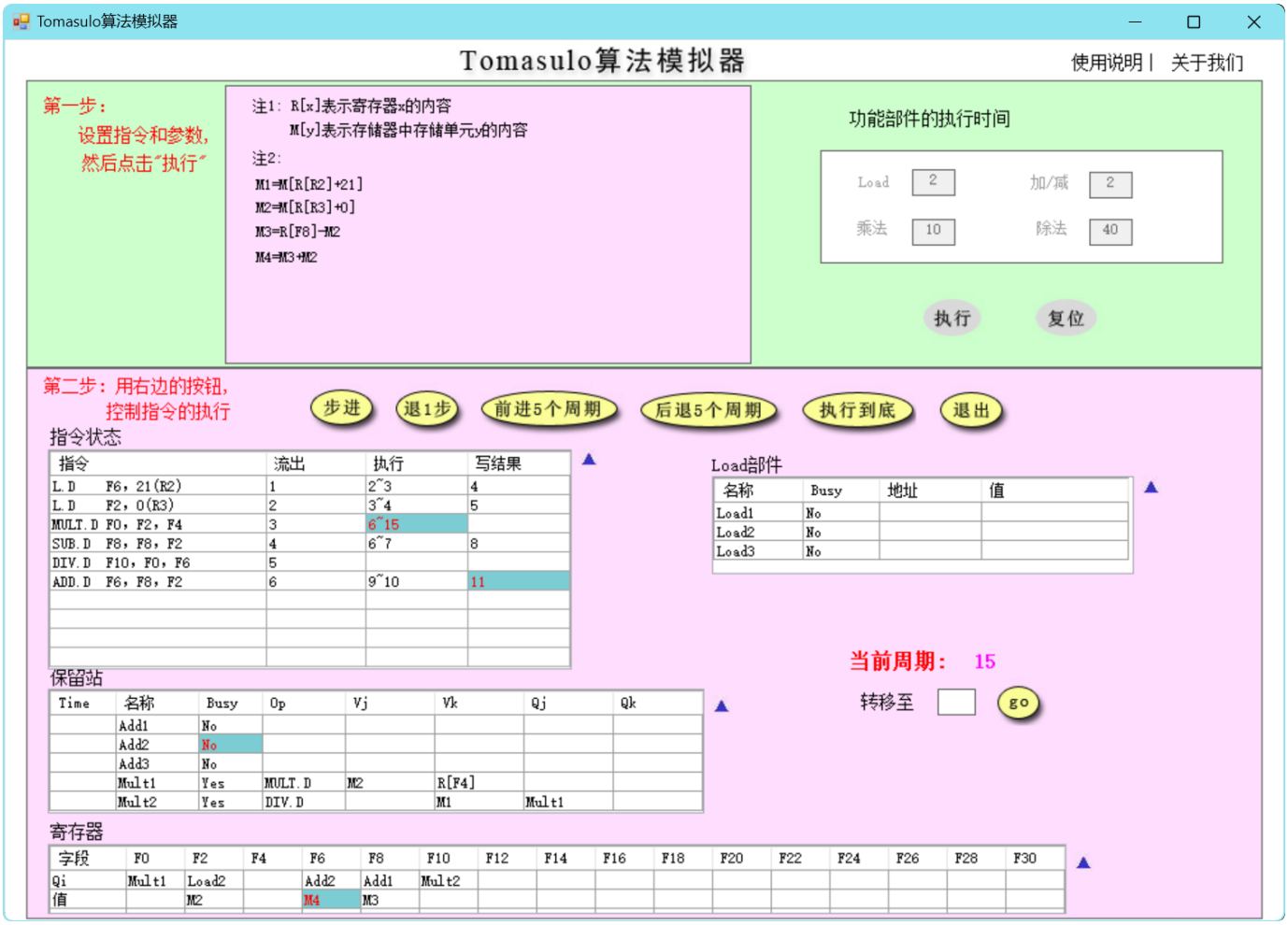
从第5周期到第6周期：

- 指令状态中，
  - MULT.D和SUB.D开始执行，因为在第5周期时，他们在保留站中对应的Vj和Qj准备好了。
  - ADD.D指令流出。
- 保留站中，
  - Time栏，Add1和Mult1开始计时，因为MULT.D和SUB.D指令开始执行了
  - Add2所在行状态变更，填入ADD.D指令的信息。Busy列变为busy,Op为加法指令，两个操作数分别填入Vk和Qj。填入Qj内容为Add1，表示第一个读操作数F8在等待Add1完成。
- 寄存器中，
  - 由于ADD.D指令的流出，F6字段的Qi为Add2，表示F6寄存器等待写入Add2的计算结果。
- Load部件无变化

3. 导致MUL.D流出后没有执行的原因

   a. MUL.D流出后，它的第一个读操作数F2正在等待Load2写入。即发生了RAW相关。

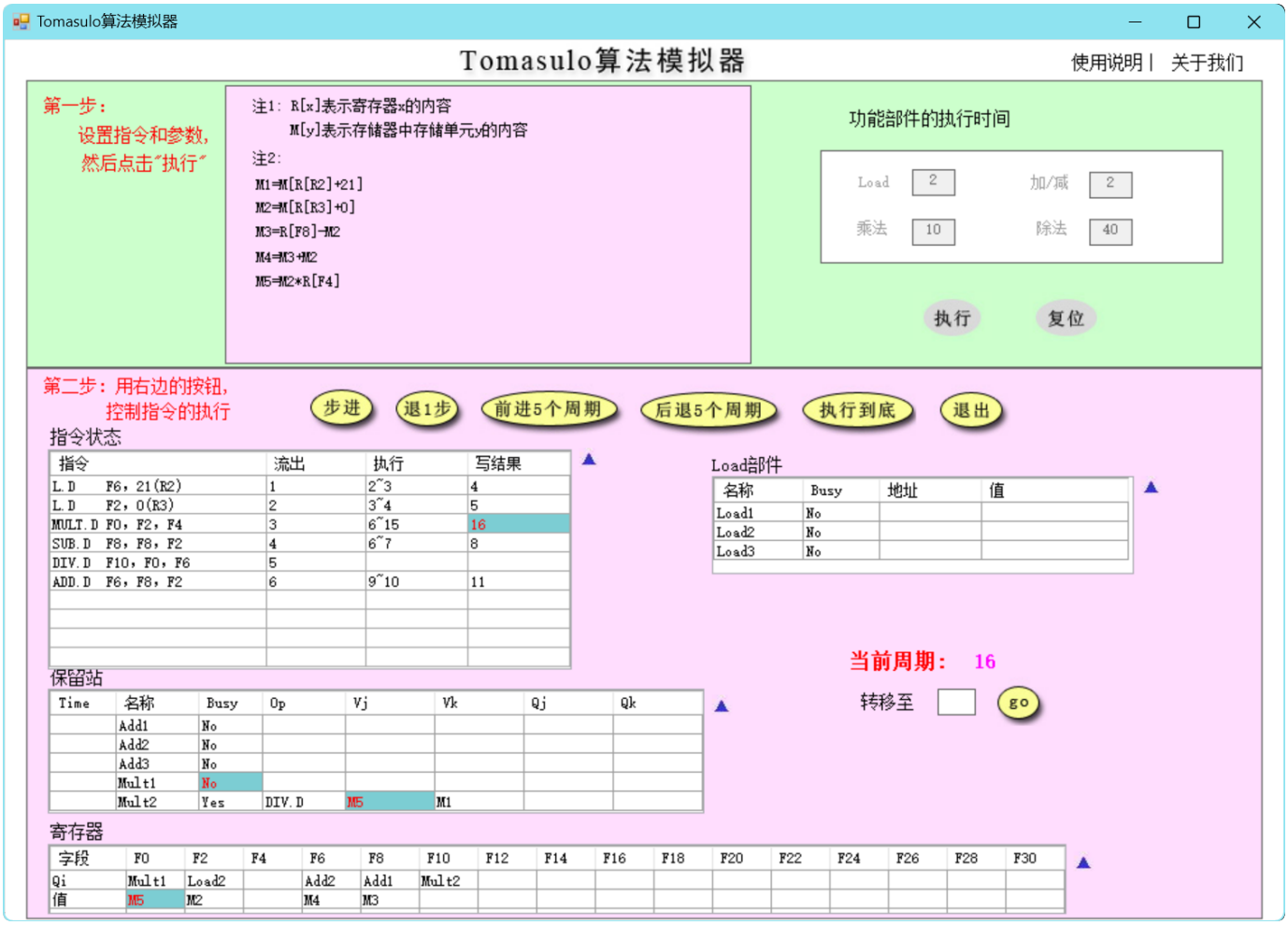
4. 第15周期



从第14周期到15周期：

- 14周期时，Mult1部件正在执行（MULT.D指令正在执行），且只差一个周期执行结束，故15周期时，MULT.D指令执行完成，指令状态表中，填入MULT.D指令执行时间为6~15周期。
- 在14周期时 ADD.D指令执行完成，故15周期时该指令写入结果。

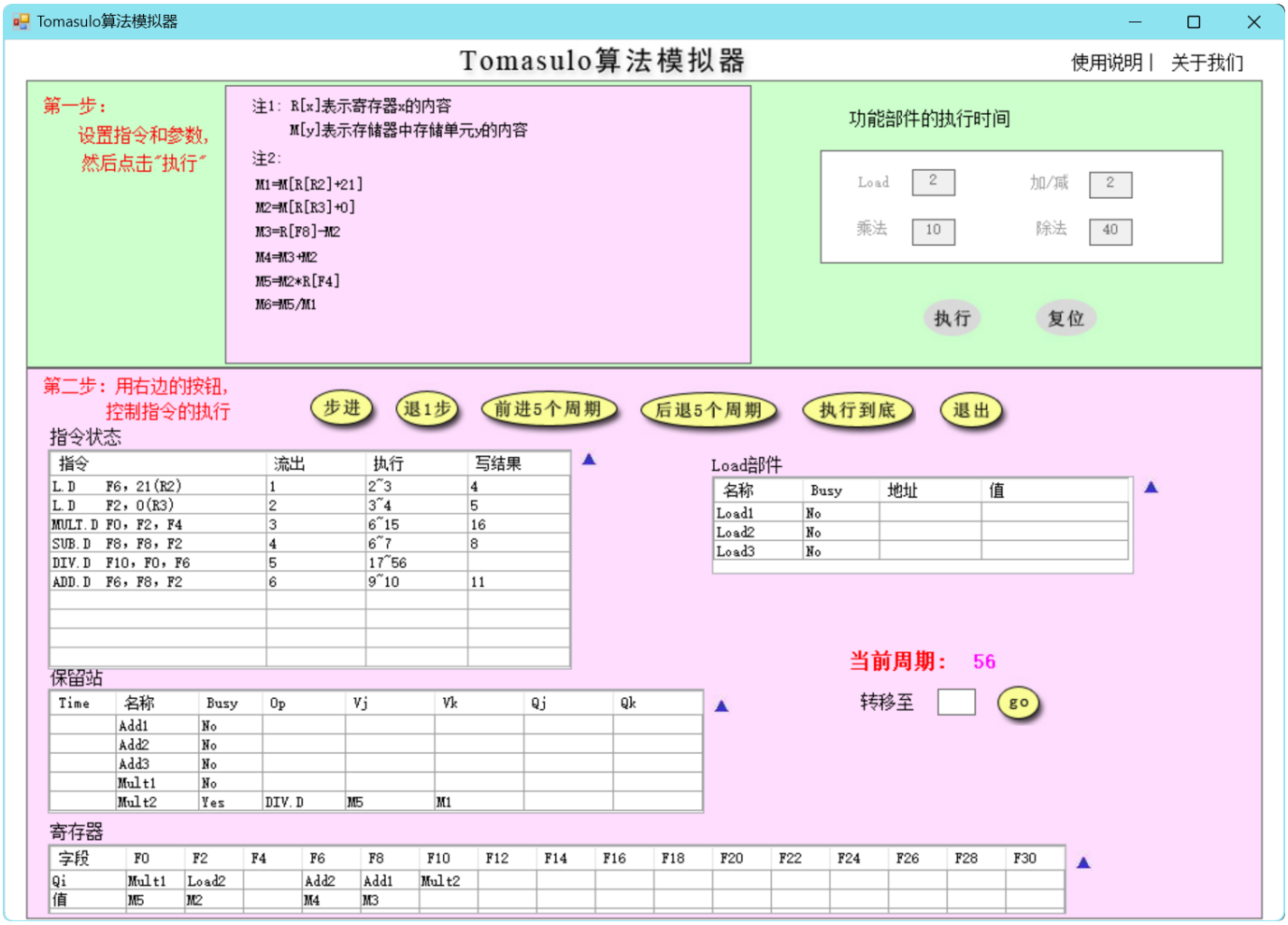
第16周期：



从第15周期到第16周期：

- MULT.D指令执行完成，写入结果
- Mult1部件执行完成，清空相应保留站
- Mult2等待的F0寄存器由于Mult1的完成而可用，故Mult2的Qj栏变到Vj栏。至此，Mult2部件的数据都准备好了，即将开始执行。

5. 指令刚刚执行完毕时是第56个周期。（实验文档描述有点歧义，“写CBD时”是理解成CBD还没写，即将写，还是理解成CBD刚刚写入。两种理解方式对应指令在第56、57周期执行完毕）



多cache一致性算法–监听法

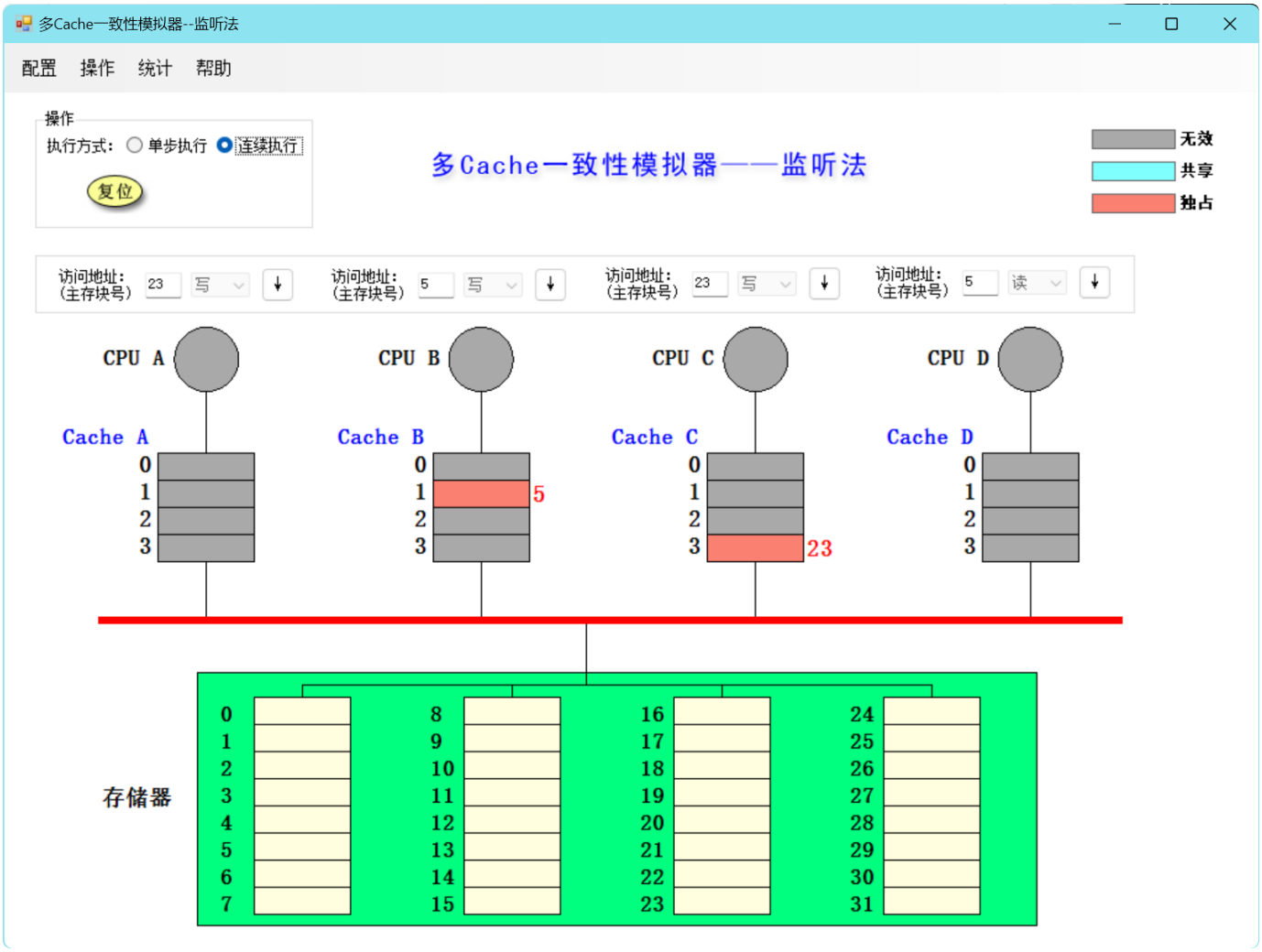
1. 模拟

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	是，存储器5替换A_1	否	cache A 读不命中，从存储器中读第5块替换cache A中的块1，块1进入S状态

CPU B 读第5块	是，存储器5替换B_1	否	cache B 读不命中，从存储器中读第5块替换cache B中的块1，块1进入S状态
CPU C 读第5块	是，存储器5替换C_1	否	cache A 读不命中，从存储器中读第5块替换cache A中的块1，块1进入S状态
CPU B 写第5块	否	否	cache A写命中，向总线广播作废信息，cache A和cache C中的块1变为I状态；cache B中的块1更新，并进入M状态
CPU D 读第5块	是，存储器5替换D_1	是，B_1写回存储器5	cache D读不命中，cache B中的块1写回存储器，且状态变为S状态，然后cache D从存储器中取出第五块，替换cache D中的块1，并进入S状态。
CPU B 写第21块	是，存储器21替换B_1	否	cache B写不命中，从存储器中取出第21块，放入块1，块1进入M状态

CPU A 写第23块	是，存储器23替换A_3	否	cache A写不命中，从存储器中取出第23块，放入块3，块3进入M状态
CPU C 写第23块	是，存储器23替换cache C_3	是，cache A_3写回存储器23	cache C写不命中，cache A中块3写回存储器，cache A的块3 进入I状态；存储器的第23块，替换cache C的块3，块3进入M状态，CPU C写入cache C的块3.
CPU B 读第29块	是，存储器29替换B_1	是，cache B_1写回存储器21	cache B读不命中，cache B_1为M状态，故先将其写回存储器21；从存储器中读出块29替换cache B_1，并进入S状态
CPU B 写第5块	是，存储器5替换cache B_1	否	cache B写不命中，从存储器中取出第5块，替换cache B_1，进入M状态，且cache D中的块1进入I状态，CPU B写入cache B_1

结束时，状态如下：

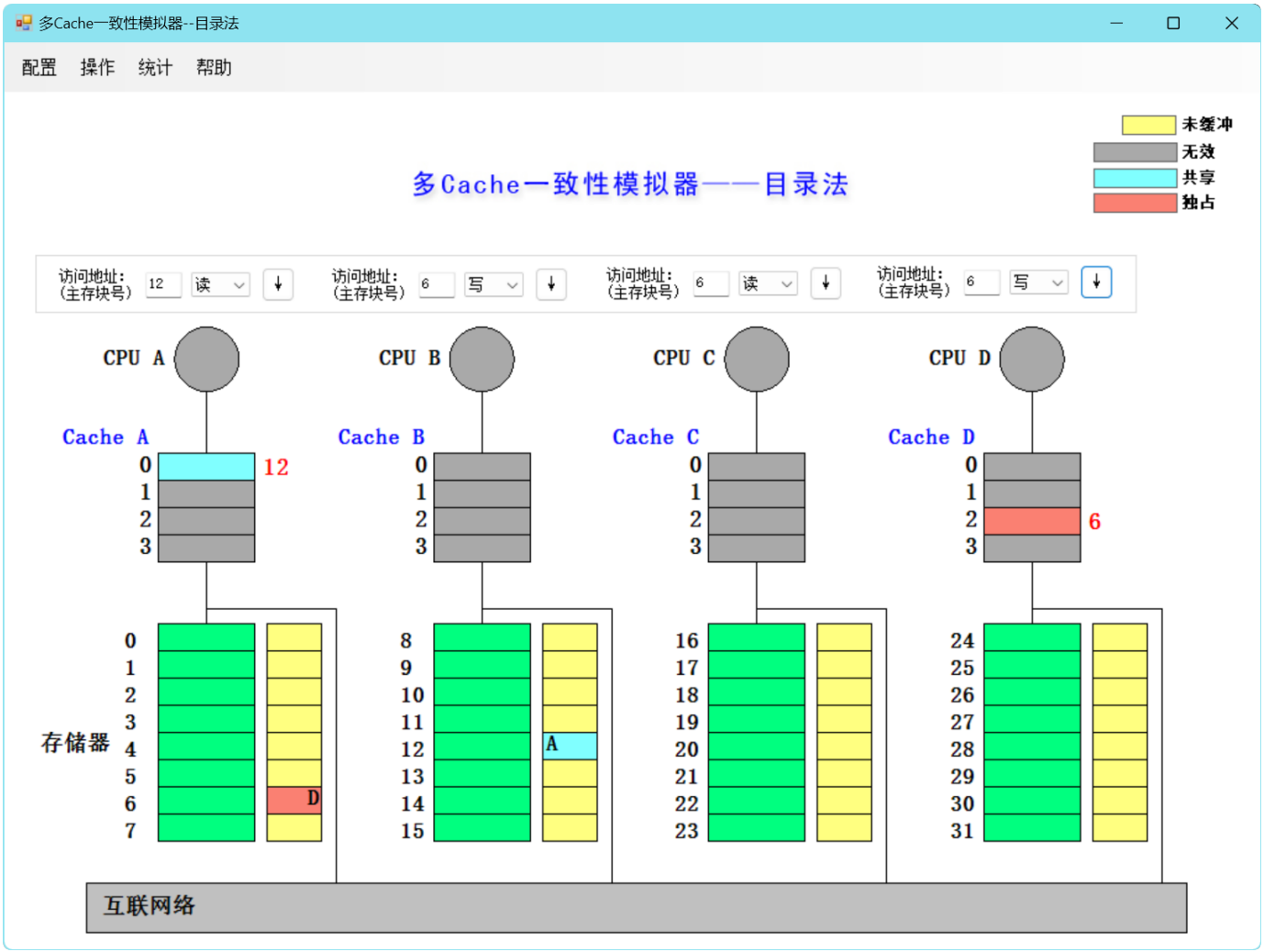


多cache一致性模拟--目录法

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	cache读不命中，本地向宿主发送“读不命中(A,6)”； 宿主把数据块送给本地节点cache A_2，cache A_2进入S状态； 存储器块6的共享集合为{A}；cache向CPU发送数据
CPU B读第6块	cache读不命中，本地向宿主发送”读不命中(B,6)"； 宿主把数据块发送给本地结点cache B_2，cache B_2进入S状态； 存储器块6的共享集合为{AB}；cache向CPU发送数据

CPU D 读第6块	cache读不命中，本地向宿主发送“读不命中(D,6)”； 宿主把存储器中的块6送给本地结点cache D_6,D_6进入S状态； 存储器块6的共享集合为{ABD};cache向CPU发送数据
CPU B 写第6块	cache B写命中；本地向宿主发送“写命中(B,6)”； 宿主收到信息，向cache A和D发送作废6信息，A_2和D_2进入I状态； 存储器块6的共享集合变为{B}，集合进入M状态;CPU向cache写入内容，B_2变为M状态
CPU C 读第6块	cache C读不命中；本地向宿主发送“读不命中(C,6)信息； 存储器块6对应共享集合为M状态，向cache B发送取数据块信息，cache B向存储器块发送数据； 存储器向cache C发送数据块；cache C_2进入S状态； 存储器6的共享集合变为{BC};cache向CPU发送数据
CPU D写第20块	cache写不命中，本地向宿主发送“写不命中(D,20)”； 宿主把数据块送给本地节点cache D_0，cache D_0进入M状态； 存储器块20的共享集合为{D}；CPU向cache写入内容
CPU A写第20块	cache A写不命中；发送”写不命中(A,20)“信息； 存储器（宿主）给cache D发送”取并作废20“； cache D_0发送数据给存储器，且D_0变为I状态； 存储器向cache A发送数据块，存储器20的共享集合变为{A}； CPU向cache写数据；

CPU D写第6块	cache D写不命中；向宿主发送”写不命中（D,6)“信息； 存储器收到信息后，向cache B和cache C发送作废6信息； cache B_2和C_2状态变为I；存储器向cache D发送数据块； 存储器6的共享集合变为{D}；CPU向cache D_2写入数据；
CPU A 读第12块	cache A读不命中；cache A向存储器写回并修改共享集（A,20）的信息；存储器20共享集合清空； cache A向宿主发送”读不命中(A,12)“信息；宿主向cache A_0发送数据。A_0变为S状态； 存储器12的共享集合变为{A};cache A向CPU发送数据



综合问答

1. 目录法和监听法分别基于集中式和基于总线，两者优劣是什么?

目录法实现简单，总线带宽占用低，可以通过扩展互联网络支持更多处理器。缺点是需要额外空间存储共享集合，当存储器块数多时，存储开销大；且存储器接口速度会限制数据传输速度。

监听法实现了写互斥和写串行，能有有效避免多个处理器同时修改同一个缓存块。缺点是存在瓶颈，性能和可扩展性会随着处理器数量的增加而受到限制。

2. Tomasulo算法相比Score Board算法有什么异同?

- Tomasulo 算法为分布式，通过在寄存器重命名来消除 WAR和 WAW ，仅在操作数都可用时才执行指令，从而避免了RAW。
- Score Board 算法为集中式，其不能直接消除 WAR 和 WAW 相关，只能检测到 WAR 和 WAW 相关，然后通过stall来解决相关问题。

3. Tomasulo 算法是如何解决结构、 RAW、 WAR 和 WAW 相关的?

- 结构相关：有结构冲突不发射
- RAW 相关：仅在操作数都可用时才执行指令。即保留站中Vj和Vk都准备好才允许执行。
- WAW 相关：寄存器重命名
- WAR 相关：寄存器重命名