Lab5 Tomasulo和cache一致性

廖洲洲 PB17081504

实验目的

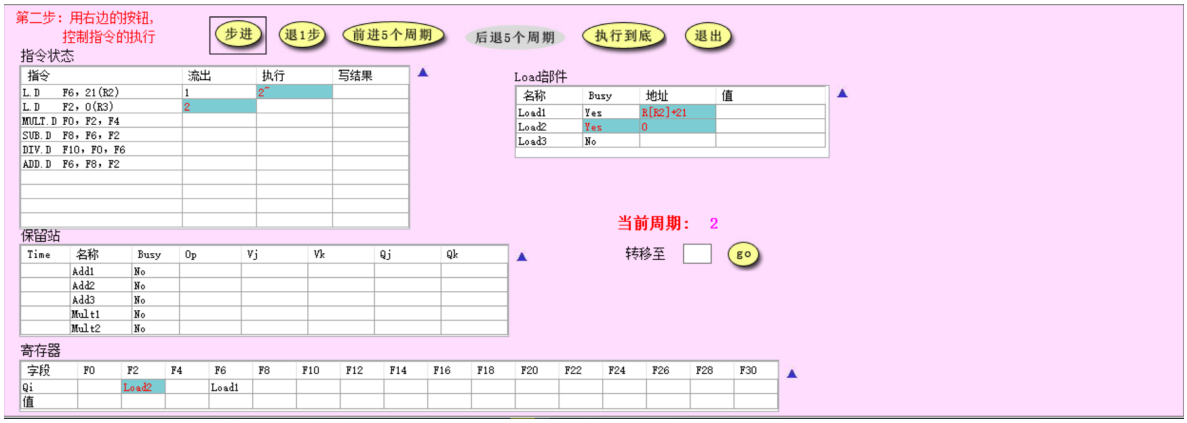
1. 熟悉Tomasulo模拟器和cache一致性模拟器（监听法和目录法）的使用
2. 加深对Tomasulo算法的理解，从而理解指令级并行的一种方式-动态指令调度
3. 掌握Tomasulo算法在指令流出、执行、写结果各阶段对浮点操作指令以及load和store指令进行什么处理；给定被执行代码片段，对于具体某个时钟周期，能够写出保留站、指令状态表以及浮点寄存器状态表内容的变化情况
4. 理解监听法和目录法的基本思想，加深对多cache一致性的理解
5. 做到给出指定的读写序列，可以模拟出读写过程中发生的替换、换出等操作，同时模拟出cache块的无效、共享和独占态的相互切换

实验内容

一、Tomasulo算法模拟器

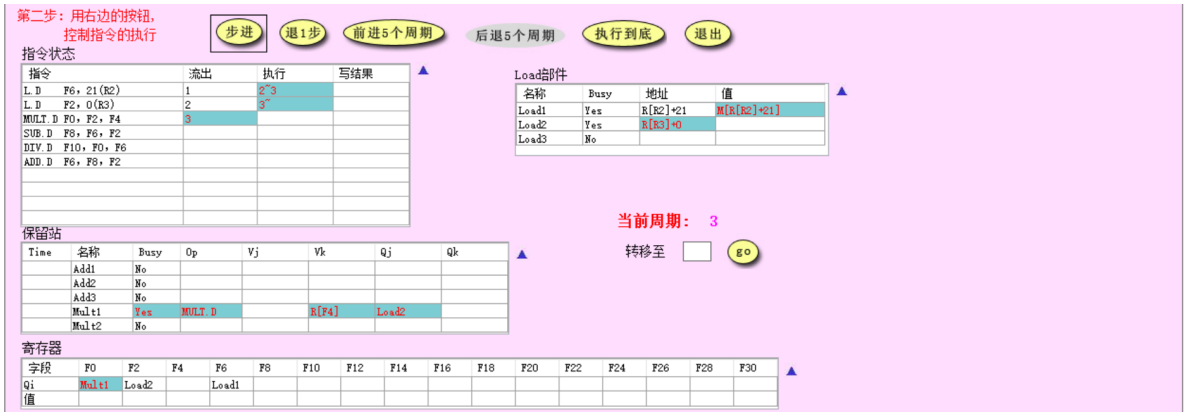
1. 分别截图（当前周期2和当前周期3），请简要说明load部件做了什么改动

答：当前周期2：



Load部件的改动：Load1部件将计算出的有效地址R[R2]+21放入地址字段；Load2部件的Busy字段标记为"YES",同时将偏移量0放入地址字段

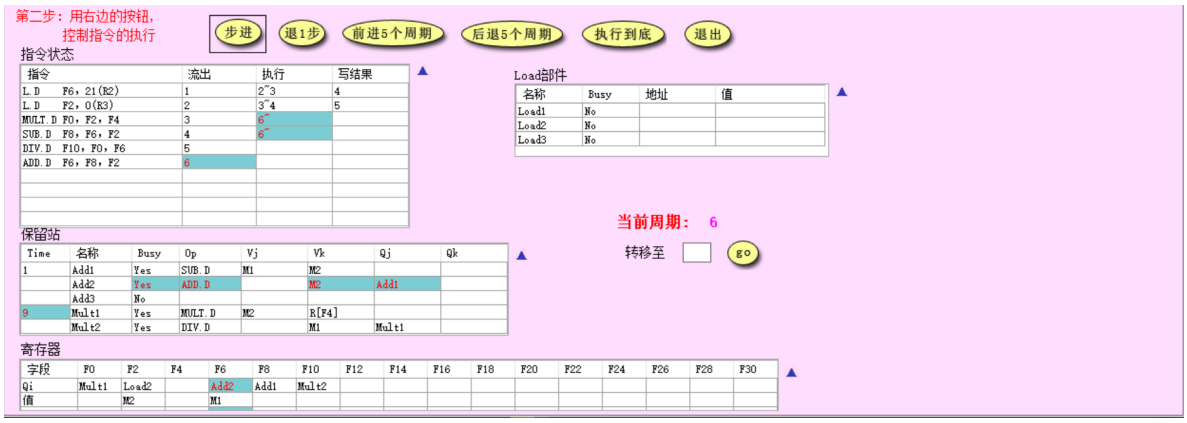
当前周期3：



Load部件的改动：Load1部件将存储器中地址为R[R2]+21存储单元的值写入值字段；Load2部件将计算出的有效地址R[R3]+0放入地址字段

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）

答：MUL.D刚开始执行时系统状态：



发生的改动

指令状态：ADD.D指令在第6个周期流出，MULT.D和SUB.D在第6个周期开始执行

保留站：Add2部件被占用，Busy字段修改为"YES",Op字段为"ADD.D",V_k为"M2",Q_j为"Add1";Add1剩余执行周期变为1，Mult1剩余执行周期变为9

寄存器：F6的Q_i由Load1变为Add2

Load部件：无变化

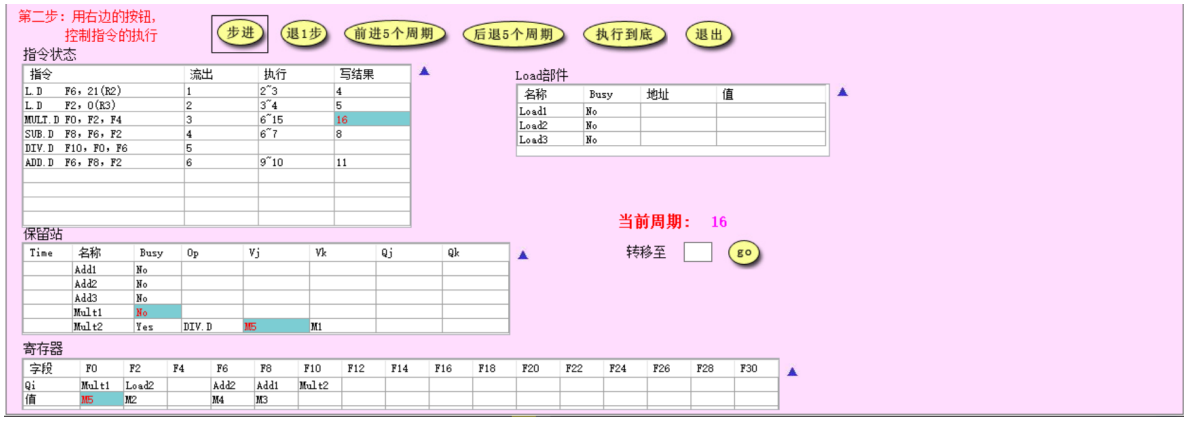
3. 简要说明是什么相关导致MUL.D流出后没有立即执行

答：RAW相关，L.D F2,0(R3)需要写F2,而MULT.D F0,F2,F4需要读F2，因此其源操作数F2需要等待来自Load2的值

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

答：15周期：

16周期：



15到16周期的变化：

Load部件：不变

指令状态：MULT.D指令执行完成，在第16周期写结果，因此写结果字段修改为16

保留站：MULT.D指令执行完毕，释放Mult1，修改Mult1的Busy字段为"NO"；Mult2监听到来自Mult1的数据，修改V_j字段为M5

寄存器：F0字段的值写为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CDB时认为指令流执行结束）

答：

第57周期

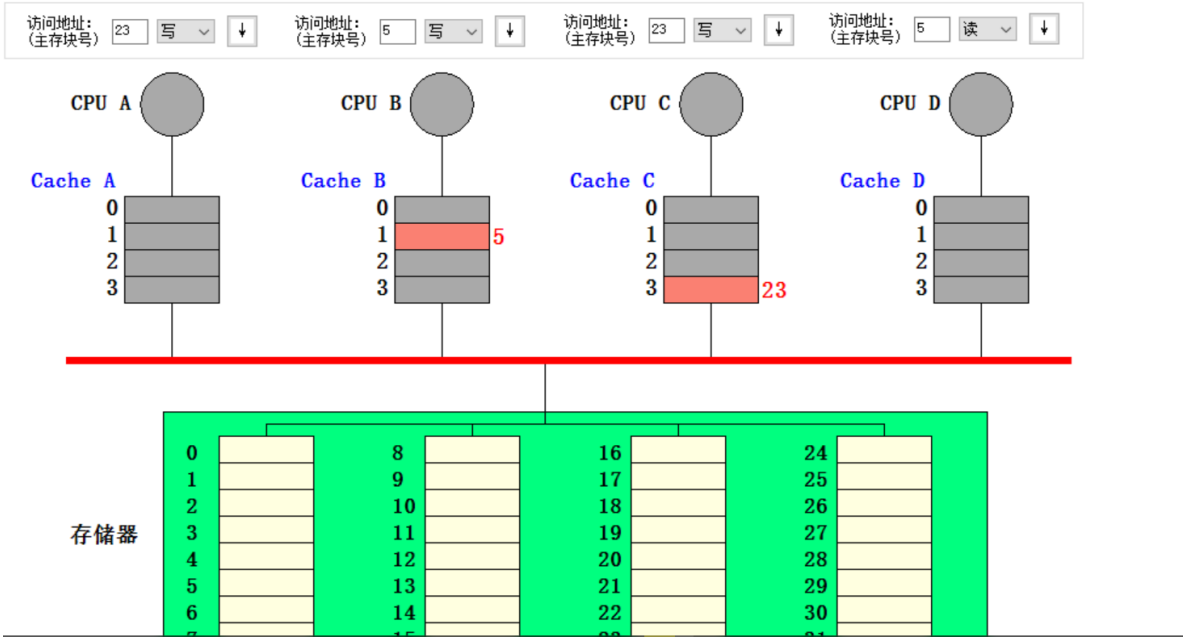
二、多cache一致性算法-监听法

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	否	否	Cache A读不命中，将读不命中放在总线上，从存储器中读取数据到Cache A，Cache A块1状态修改为共享。
CPU B 读第5块	否	否	Cache B读不命中，将读不命中放在总线上，从存储器中读取数据到Cache B，Cache B块1状态修改为共享。
CPU C 读第5块	否	否	Cache C读不命中，将读不命中放在总线上，从存储器中读取数据到Cache C，Cache C块1状态修改为共享。
CPU B 写第5块	否	否	Cache B写命中，将作废信号放在总线上，Cache A和Cache C块1状态修改为失效，Cache B状态修改为独占。
CPU D 读第5块	否	是	Cache D读不命中，将读不命中信号放在总线上，Cache B写回块1，修改块1状态为共享，从存储器读取块5到Cache D，Cache D块1状态修改为共享。
CPU B 写第21块	是	否	Cache B写不命中，将写不命中信号放在总线上，替换Cache B的块1，修改Cache B块1的状态为独占。
CPU A 写第23块	否	否	Cache A写不命中，将写不命中信号放在总线上，修改Cache A块3状态为独占

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 C 写第23块	否	是	Cache C写不命中，将写不命中信号放在总线上，Cache A写回块3，Cache C写块3，Cache A块3状态修改为无效，Cache C块3状态修改为独占。
CPU B 读第29块	是	是	Cache B读不命中，写回块1，修改Cache B块1状态为无效；读取存储器块29的数据到Cache B块1，修改Cache B块1状态为共享。
CPU B 写第5块	是	否	Cache B写不命中，将写不命中信号放在总线上，将存储器块5取到Cache B块1，修改块1数据，修改Cache B块1状态为独占，Cache D块1状态为无效。

2.请截图，展示执行完以上操作后整个cache系统的状态

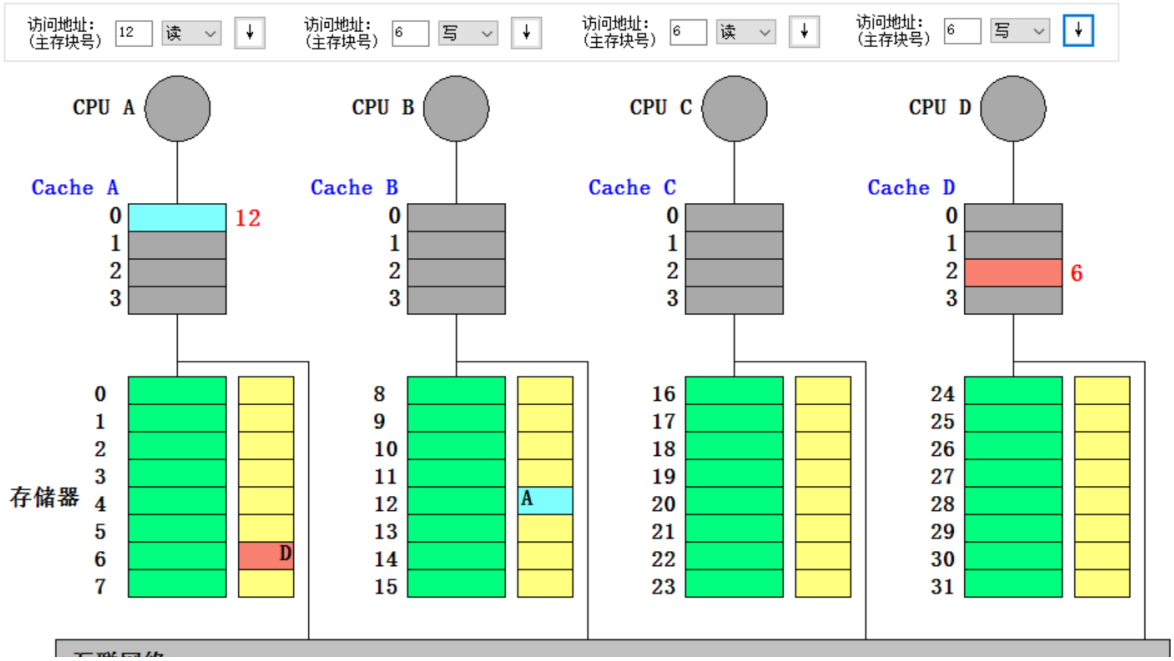


三、多cache一致性算法-目录法

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	CPU A读不命中，本地向宿主结点A发读不命中(A,6)消息，宿主把数据块送给本地结点，存储器块6的状态变为共享，共享集合为{A},Cache A的的块2状态变为共享
CPU B 读第6块	CPU B读不命中，本地向宿主结点A发读不命中(B,6)消息，宿主把数据块送给本地结点B，存储器块6共享集合变为{A,B},Cache B的块2状态变为共享
CPU D 读第6块	CPU D读不命中，本地向宿主结点A发读不命中(D,6)消息，宿主把数据块送给本地结点D，存储器块6共享集合变为{A,B,D},Cache D的块2状态变为共享
CPU B 写第6块	CPU B写命中，本地向宿主结点A发写命中(B,6)消息，宿主结点A向远程结点A、D发作废(6)消息，存储器块6共享集合变为{B},状态变为独占，Cache A、D块2状态变为无效，Cache B块2状态变为独占
CPU C 读第6块	CPU C读不命中，本地向宿主结点A发读不命中(C,6)消息，宿主A给远程结点B发取数据块(6)的消息，远程结点B把数据块送给宿主结点A，Cache B的块2状态变为共享，宿主把数据块送给本地结点，Cache C块2状态变为共享，存储器块6状态变为共享，共享集合为{B,C}
CPU D 写第20块	CPU D写不命中，本地向宿主结点C发写不命中(D,20)消息，宿主C把数据块送给本地结点，Cache D块0状态变为独占，存储器块20状态变为独占，共享集合为{D}
CPU A 写第20块	CPU A写不命中，本地向宿主结点C发写不命中(A,20)消息，宿主C给远程结点D发送取并作废(20)的消息，远程结点D把数据块送给宿主结点，Cache D块0状态变为无效，宿主C把数据块送给本地结点A，存储器共享集合变为{A},Cache A块0状态变为独占
CPU D 写第6块	CPU D写不命中，本地向宿主结点A发写不命中(D,6)消息，宿主A向远程结点B、C发作废(6)消息，Cache B、C块2状态变为无效，宿主A把数据块送给本地结点，Cache D块2状态变为独占，存储器块6状态变为独占，共享集合变为{D}
CPU A 读第12块	CPU A读不命中，本地向被替换块20的宿主结点C发写回并修改共享集(A,20)的消息，存储器块20状态变为未缓冲，本地向宿主结点B发读不命中(A,12)消息，宿主结点B把数据块送给本地结点A，Cache A块0状态变为共享，存储器块12状态变为共享，共享集合变为{A}

2. 请截图，展示执行完以上操作后整个cache系统的状态。



四、综合回答

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）

答：对于监听法：其在每次发生缓存缺失时都需要与所有缓存进行通信，包括对共享数据的写入操作。其优点是没有任何用于跟踪缓存状态的集中式数据结构，可以降低成本。但是监听法是基于总线的，总线的可扩放性受到一定限制，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的问题；在由大量处理器构成的多处理器系统中，监听带宽是瓶颈。

对于目录法：其优点是发生缺失时不需要向所有其它缓存进行广播，缺点是成本更高，存在目录开销，对于大型多处理器，需要一些方法来高效扩展目录结构。

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）

答：相同之处：都能检测结构相关，但是无法消除，即有结构冲突时不发射；消除RAW的思想相同，通过推迟指令执行，直到操作数可用为止，避免RAW相关；

不同之处：Score Board只能检测WAR和WAW相关，但是无法消除，需要插入停顿来解决；而Tomasulo算法可以通过寄存器重命名来消除WAR和WAW相关；Score Board控制和缓存集中在记分牌，因此是集中式；Tomasulo控制和缓存分布在各部件中，因此是分布式。

3.Tomasulo算法是如何解决结构、RAW、WAR和WAW相关的?

答：结构相关：如果没有空保留站，则存在结构性冒险，该指令会停顿，直到有保留站或缓存区被释放为止。

RAW：如果还有一个或多个操作数不可用，则在等待计算的同时监听公共数据总线。当一个操作数可用时，就将它放到任何一个正在等待它的保留站中。当所有操作数可用时，则可以执行运算。即通过延迟指令执行，直到操作数可用为止来解决RAW相关。

WAR和WAW：利用保留站来重命名寄存器，并在操作数可用时立即将其存储在保留站中。