计算机体系结构 实验五 Tomasulo 和 cache 一致性模拟器使用

PB18151866 龚小航

【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

【实验环境】

实验提供的模拟器

【实验要求】

根据每阶段的任务，完成实验并回答问题。

【实验过程及具体实现】

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
```

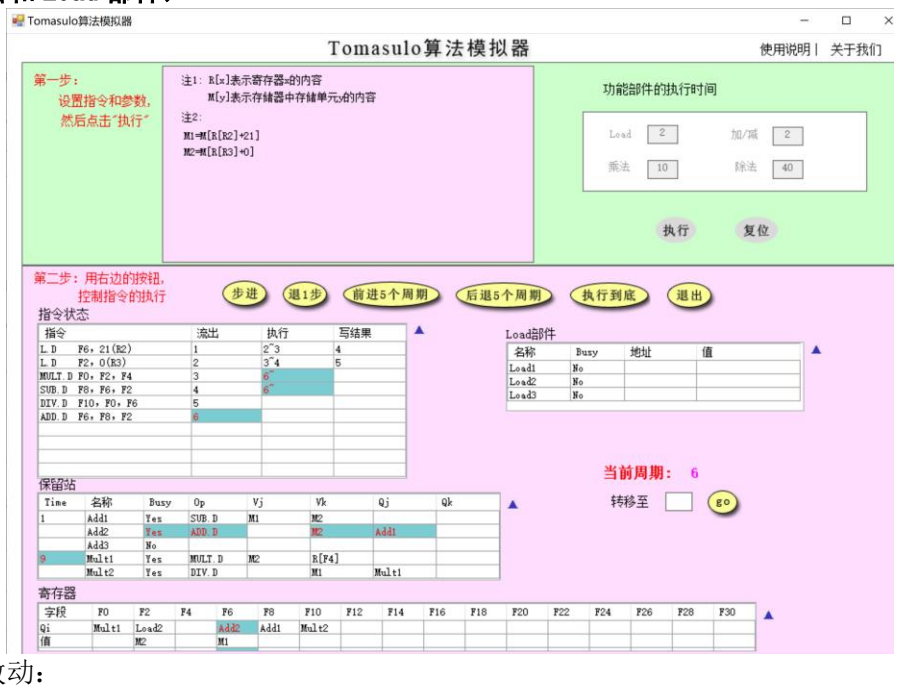
假设浮点功能部件的延迟时间：加减法 2 个周期，乘法 10 个周期，load/store2 个周期，除法 40 个周期。

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

周期 2：Load1 部件设为 busy，地址为第一条指令寻址地址。同时第二条 LD 指令发射，Load2 部件设为 Busy，但第二条指令还未执行，Load2 地址暂设为默认值 0.

周期 3：第二条 LD 指令执行，Load2 的地址设为它的寻址地址；同时第一条 LD 继续执行，Load1 把对应地址的存储器值载入，更新 Load 部件的值字段。

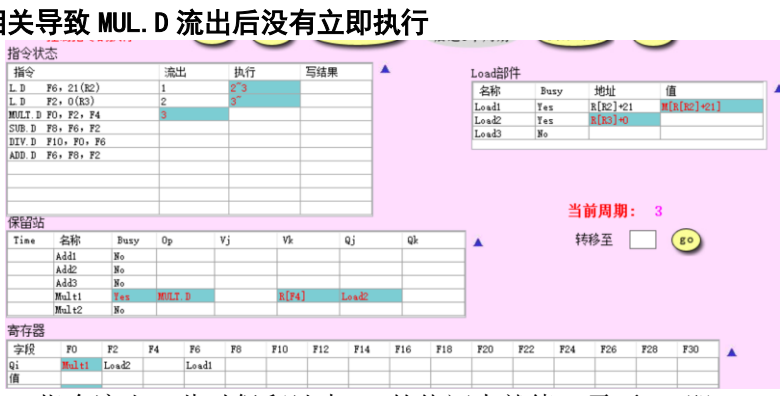
2. 请截图（MUL.D 刚开始执行时系统状态），并说明该周期相比上一周期整个系统发生了哪些改动（指令状态、保留站、寄存器和 Load 部件）



发生的改动：

- 指令状态：ADD.D F6,F8,F2 指令流出，MULT.D F0,F2,F4 和 SUB.D F8,F6,F2 指令开始执行；
- 保留站：随着最后一条加法指令发射，Add2 设为 Busy，Op 设为 ADD.D，Vk 设为 M1，Qj 设为 Add1（Vj 需要等待 Add1 的结果），Mult1 的 Time 设为 9，即执行完这条指令还需 9 个周期。
- 寄存器：F6 字段上的 Qi 由 Load1 改为 Add2，表示 Add2 的计算结果需要写入 F6。
- Load 部件：与 LD 指令无关，无改动。

3. 简要说明是什么相关导致 MUL.D 流出后没有立即执行

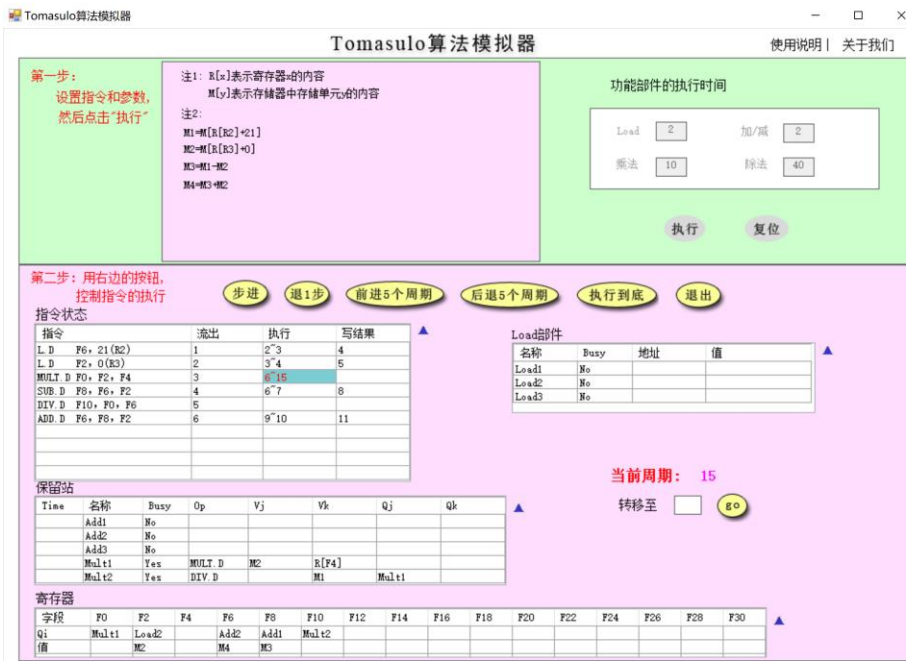


周期 3 时 MUL.D 指令流出，此时保留站中 Vj 的值还未就绪，需要 Qj 即 Load2 部件提供的值。而 Load 部件部分此时 Load2 还未读出存储器内对应的值，因此这条乘法指令不能立即执行。

即第二条指令与第三条指令之间存在关于寄存器 F2 的写后读（RAW）相关。

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

• 周期 15：按指令状态可知，12，13，14 周期仅 MULT.D 指令还未执行完毕，整个系统仅保留站中的 Time 每周期-1。15 周期时，MULT.D 指令刚好执行完毕，在指令状态栏中这条指令的执行周期填补完整，为 6~15，同时保留站中 MUL1 对应的 Time 字段清空。



• 周期 16：MULT.D 指令写结果，指令状态栏中这条指令的写结果列设为本周期 16，同时保留站中所有 MUL1 相关内容全部清空，因为这条指令以及执行完成。由于结果已经可用，先将寄存器中 F0 对应的值改为算出的结果 M5，同时将保留站中 MUL2 的等待参数 Vj 设为 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 CBD 时认为指令流执行结束）

由图可知，最后一条指令执行完毕实在 57 周期时，DIV.D 指令写 CBD。

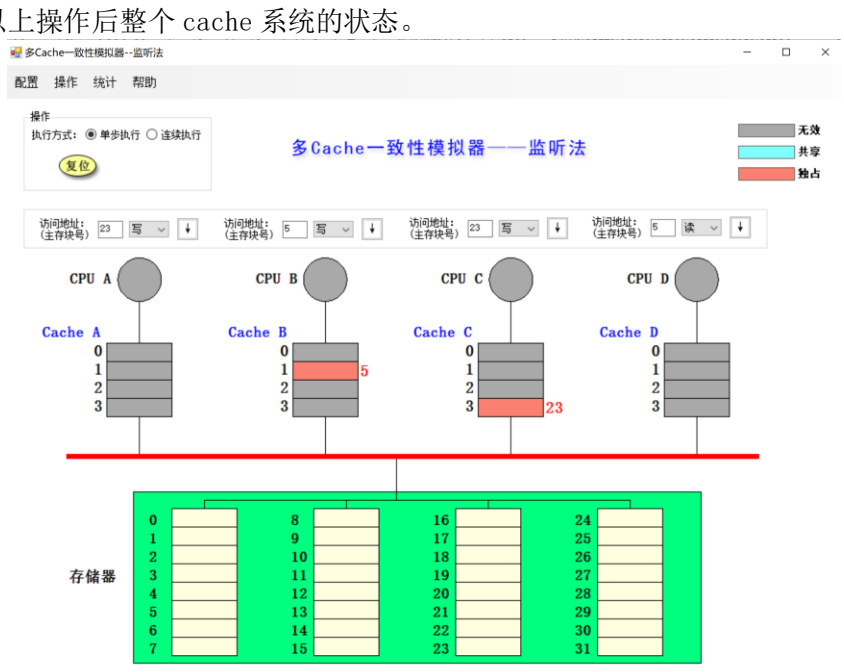
二、多 cache 一致性算法-监听法

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	5 替换 cacheA 块 1	/	CacheA 读不命中，从存储器中取出块 5 放入 CacheA 块 1，完成后 CacheA 将信息传递给 CPUA
CPU B 读第 5 块	5 替换 cacheB 块 1	/	CacheB 读不命中，从存储器中取出块 5 放入 CacheB 块 1，完成后 CacheB 将信息传递给 CPUB
CPU C 读第 5 块	5 替换 cacheC 块 1	/	CacheC 读不命中，从存储器中取出块 5 放入 CacheC 块 1，完成后 CacheC 将信息传递给 CPUC
CPU B 写第 5 块	/	/	CacheB 写命中，总线发出写作废信号，CacheA 和 CacheC 中块 1 的内容作废。最后 CPUB 向 CacheB 写块 1(独占)

CPU D 读第 5 块	5 替换 CacheD 块 1	CacheB 块 1 写回 5	CacheD 读不命中,总线散播消息, B 将块 1 写回存储器并将 CacheB 内块 1 设为共享。最后存储器块 5 换入 CacheD 块 1（共享）
CPU B 写第 21 块	21 替换 CacheB 块 1	/	CacheB 写不命中，总线散播消息，并将存储器 21 替换 CacheB 块 1。最后 CPUB 向 CacheB 写块 1（独占）
CPU A 写第 23 块	23 替换 CacheA 块 3	/	CacheA 写不命中，总线散播消息，并将存储器 23 替换 CacheA 块 3。最后 CPUA 向 CacheA 写块 3（独占）
CPU C 写第 23 块	23 替换 CacheA 块 3	CacheA 块 3 写回 23	CacheC 写不命中，总线散播消息。A 将块 3 写回存储器 23，同时将块 3 设为（无效）。而后存储器块 23 块经总线替换 CacheC 块 3。最后 CPUC 向 CacheC 写块 3（独占）
CPU B 读第 29 块	29 替换 CacheB 块 1	CacheB 块 1 写回 21	CacheB 读不命中，总线散播消息，CacheB 将块 1 写回存储器 21，同时读不命中信号通过总线传播，此时存储器中取出块 29 放入 CacheB 块 1(共享)，完成后 CacheB 将信息传递给 CPUB
CPU B 写第 5 块	5 替换 CacheB 块 1	/	CacheB 写不命中，总线散播消息，存储器中块 5 替换 CacheB 中块 1（共享），同时 CacheD 收到总线消息将块 1 位置设为无效。最后 CPUB 向 CacheB 写块 1（独占）

2. 请截图，展示执行完以上操作后整个 cache 系统的状态。

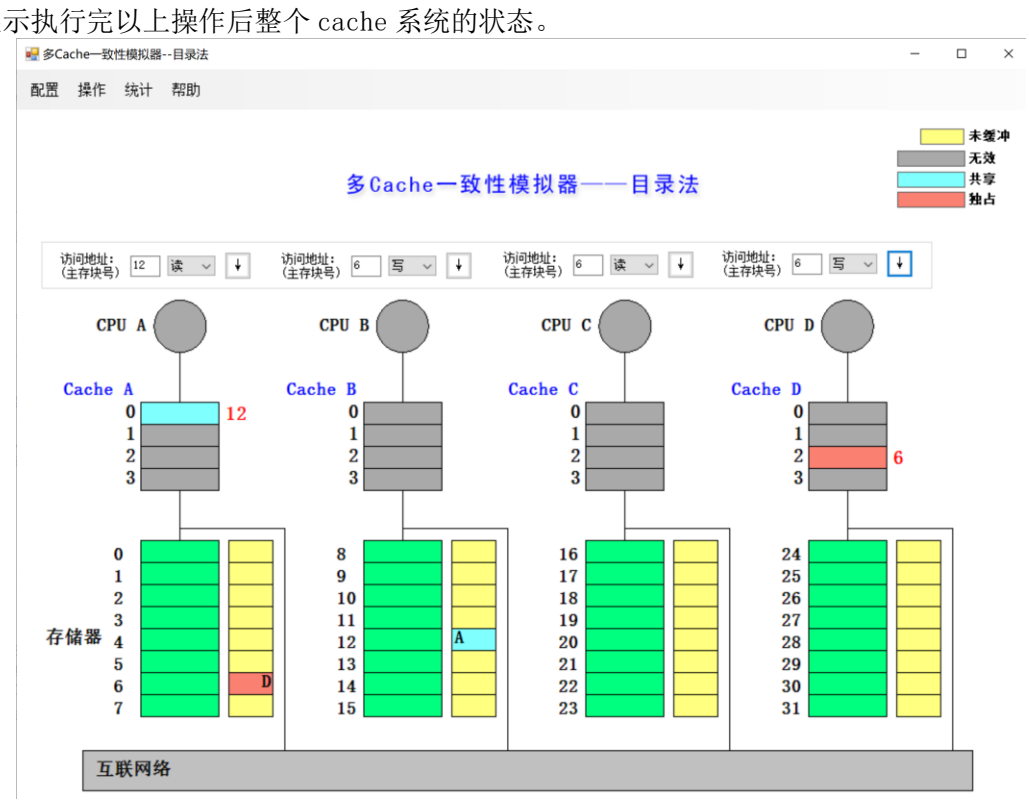


三、多 cache 一致性算法-目录法

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	CPUA 读 CacheA 块 2 不命中，向宿主结点(A)存储器 6 发送读不命中(A,6)消息，宿主把数据块传送给本地结点 CacheA 块 2，CacheA 块 2 状态由无效改为共享，存储器 6 的共享集合设为{A}，最后 CacheA 将数据传给 CPUA。
CPU B 读第 6 块	CPUB 读 CacheB 块 2 不命中，向远程宿主结点(A)存储器 6 发送读不命中(B,6)消息，远程宿主(A)的存储器将块 6 通过互联网络发送到本地节点 B 并将该块数据放入 CacheB 块 2 位置，CacheB 块 2 状态由无效改为共享，远程宿主(A)修改存储器 6 的共享集合为{A,B}。最后 CacheB 将数据传给 CPUB
CPU D 读第 6 块	CPUD 读 CacheD 块 2 不命中，向远程宿主结点(A)存储器 6 发送读不命中(D,6)消息，远程宿主(A)的存储器将块 6 通过互联网络发送到本地节点 D 并将该块数据放入 CacheD 块 2 位置，CacheD 块 2 状态由无效改为共享，远程宿主(A)修改存储器 6 的共享集合为{A,B,D}。最后 CacheD 将数据传给 CPUD
CPU B 写第 6 块	CPUB 写 CacheB 块 2 命中，向远程宿主结点(A)存储器 6 发送写命中(B,6)消息，远程宿主(A)的存储器块 6 向共享集合中所有其他共享节点发送写作废消息，即 A,D 中块 2 全部作废。远程宿主(A)将数据块发送给本地节点(B),存储器块 6 的共享集合为{B}（独占）。最后 CPUB 向 CacheB 写块 2(独占)
CPU C 读第 6 块	CPUC 读 CacheC 块 2 不命中，向远程宿主结点(A)存储器 6 发送读不命中(C,6)消息，远程宿主(A)向远程节点(B)发送取数据块(6)消息。同时远程节点(B)把数据块发送给宿主节点(A)，并将 CacheC 块 2 设为共享。随后宿主节点(A)将数据块发送给本地节点(C)，放入 CacheC 块 2 中（共享），此时 A 中存储器 6 的共享集合设为{B,C}（共享）。最后 CacheC 将块 2 数据发送给 CPUC
CPU D 写第 20 块	CPUD 写 CacheD 块 0 不命中，向远程宿主结点(C)存储器 20 发送写不命中(D,20)消息，远程宿主(C)将数据块发送给本地节点(D)，同时存储器块 20 的共享集合为{D}(独占)。最后 CPUD 向 CacheD 写块 0(独占)
CPUA 写第 20 块	CPUA 写 CacheA 块 0 不命中，向远程宿主结点(C)存储器 20 发送写不命中(A,20)消息，远程宿主(C)通过互联网络向远程节点(D)发送取并作废(20)消息，远程节点(D)将 CacheD 块 0 发送给宿主节点(C)同时将 CacheD 中块 0 作废。随后宿主节点(C)将数据块发送给本地节点(A)，存于 CacheA 块 0。远程宿主(C)的存储器 20 的共享集合设为{A}(独占)。最后 CPUA 向 CacheA 写块 0(独占)
CPUD 写第 6 块	CPUD 写 CacheD 块 2 不命中，向远程宿主结点(A)存储器 6 发送写不命中(D,6)消息，远程宿主(A)的存储器块 6 向共享集合中所有其他共享节点发送写作废消息，即 B,C 中块 2 全部作废。远程宿主(A)将数据块发送给本地节点(D),同时存储器块 6 的共享集合为{D}（独占）。最后 CPUD 向 CacheD 写块 2(独占)
CPUA 读第 12 块	CPUA 读 CacheA 块 0 不命中，块 0 原本存储 20 块的修改过后独占数据，先向被替换块(20)的宿主节点发送写回并修改共享集(A,20)消息，块 0 空出。此时再向宿主节点(B)发送读不命中(A,12)消息，宿主节点(B)将数据块 12 传送给本地节点(A)，存于 CacheA 块 0 中（共享）。同时存储器 12 的共享集合为{A}（共享），最后 CacheA 将数据传给 CPUA。

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）

监听法优势：

1. 存储成本低，所有信息通过总线传递，且存储器全局唯一不需要存储器-存储器通讯
2. 可扩展性强，增加新处理器时只需要将其连接到总线上即可，简单快速。

监听法劣势：

1.总线事务较多，通信开销大

2.处理器数目多时，总线竞争激烈

目录法优势：

1. 对总线带宽的占用较低，总线通信压力小
2. 将存储器分为多个部分，通过互联网络连接，增加了并发性。

目录法劣势：

1.当处理器数目较多时存储开销会较大

2.存储器通信频繁，存储器访存速度可能会成为瓶颈

3.拓展性不如监听法，新增若干个新处理器需要重新分配存储器宿主

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）

记分牌方法能检测 WAR 和 WAW 相关，一旦检测到存在 WAR 和 WAW 相关，通过插入停顿周期来解决这一相关。所以，记分牌方法不能消除 WAR 和 WAW 相关。

Tomasulo 算法通过寄存器重命名可以消除 WAR 和 WAW 相关。

Tomasulo 算法是分布式的，ScoreBoard 算法是集中式的。

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指令

RAW 相关：检测到没有冲突，即存储器就绪后再读取操作数，才能进入执行阶段

WAR 相关和 WAW 相关：使用寄存器值或者指向寄存器的指针代替指令中的寄存器来避免，即寄存器重命名