实验四：Tomasulo 算法实验

19281171　　王雨潇

实验报告要求：实验报告用 Word 排版，文件内一定要有"学号姓名_学号"。实验报告应提交到课程平台，按时提交。实验报告的内容应尽量全面，避免简单化。简述实验内容，实验过程，实验结果，实验分析，心得体会等。答每道题前应将题目拷贝到实验报告上。

一、 实验目的

熟悉 Tomasulo 算法的实现流程，掌握指令流水化过程中乱序执行和寄存器重命名原理。

二、 实验过程

1. 仔细阅读给出的 Tomasulo 模拟器的使用说明，熟悉该模拟器的使用。

答：Tomasulo 模拟器的使用说明如下所示

1. 设置指令和参数

本模拟器最多可以模拟执行10条指令。你可以在“指令”区选择和设置所要的指令。供选择的指令有以下5种，其中：

（1）L.D指令从主存读取一个双精度浮点数；

（2）ADD.D是双精度浮点加法指令；

（3）SUB.D是双精度浮点减法指令；

（4）MULT.D是双精度浮点乘法指令；

（5）DIV.D是双精度浮点除法指令；

你可以在窗口的右上区域设置各部件的执行时间（时钟周期数）。

“复位”的作用是使所有设置恢复为默认值。

2. 执行

点击“执行”按钮，就进入执行状态。你可以用中间的按钮来控制指令的执行，包括“步进”、“退1步”、“前进5个周期”、“后退5个周期”、“执行到底”、“退出”等。还可以用 “go” 按钮直接转到你所指定的时钟周期。

向前执行后，状态表中抹色的字段表示其内容发生了变化。

3. 对比状态表

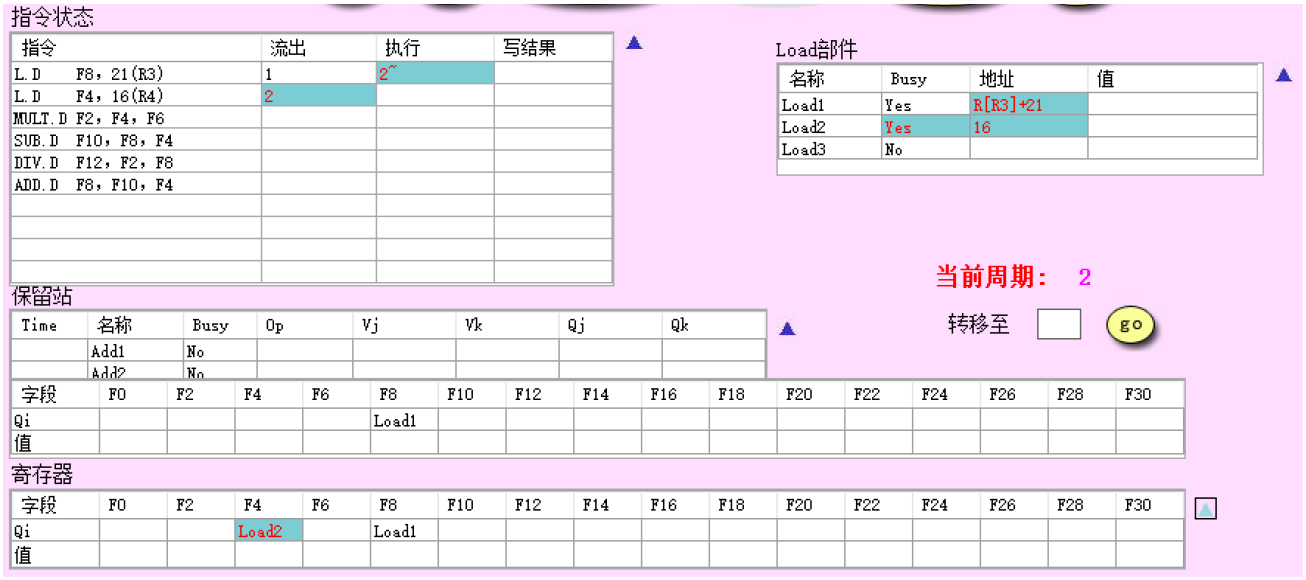
每一个状态表的右上角外侧都有一个小三角，用鼠标左键点击它，会弹出该表在上一个时钟周期的内容。这是为了让你通过对比来了解哪些内容发生了变化。在弹出表以外的区域再次点击鼠标，就可以将弹出的表收回。

2. 分析该模拟器中默认程序的指令乱序执行和寄存器重命名过程。观察并分析在程序执行过程中，指令状态、保留站、load 部件、寄存器的状态变化。（可以选取几个 cycle 来说明）

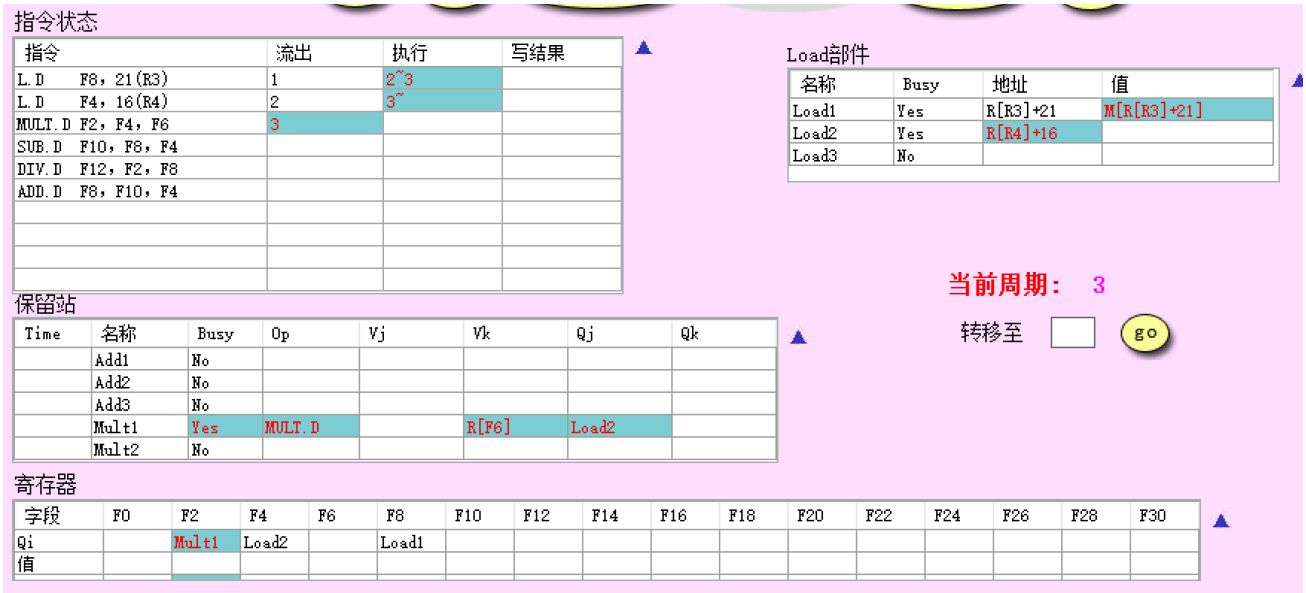
答：模拟器的默认程序和功能部件执行时间如下图所示

第 1 个周期，可以看到第 1 条指令 L.D F8, 21(R3)是一个 load 取数指令，此时 Load 部件有空间，所以 L.D F8, 21(R3)可以发射，F8 重命名为 Load1，地址填入数据地址 21，尚不可用所以 Busy 填入 yes；寄存器 F8 的 Qi 填入 Load1，表示操作的结果在 Load1 中；

第 2 个周期，第 1 条指令 L.D F8, 21(R3)已经进入了执行阶段，Load1 已经计算出了要取数的存储器地址是[R3]+21；由于 Load 部件还有空位，第 2 条指令 L.D F4, 16(R4)发射，F4 重命名为 Load2，地址填入数据地址 16，Busy 填入 yes，寄存器 F4 的 Qi 字段填入 Load2；



第 3 个周期，前 2 条指令都在执行阶段，Load1 已经从要取数的存储器地址取得了内容，Load2 已经计算出了要取数的存储器地址是[R4]+16；第 3 条指令 MULT.D F2, F4, F6 是一个 FP 操作，目前保留站有空位，所以发射；F2 重命名为保留站 Mult1，Busy 字段填入 yes，Op 填入操作 MULT.D，Vk 源操作数填入[F6]，由于[F4]还需要等 Load2 算出来，所以 Vj 空着，Qj 填入 Load2；寄存器中 F2 的 Qi 值填入 Mult1；

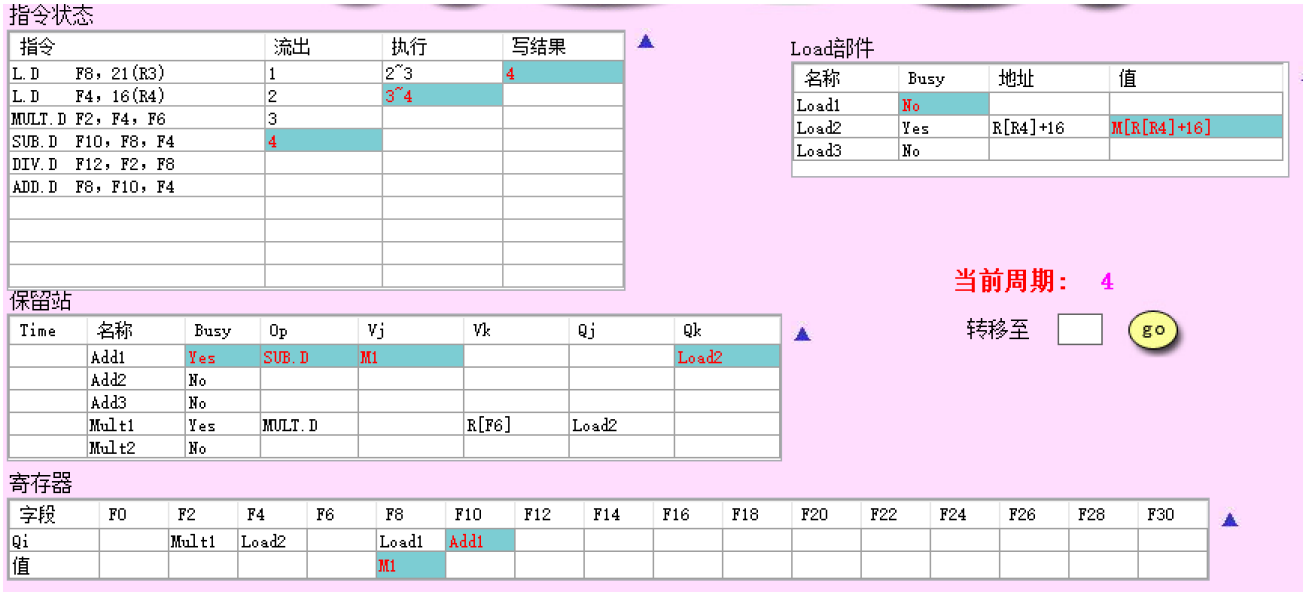


第 4 个周期，第 1 条指令 L.D F8, 21(R3)执行完成，到了写结果状态，Load1 的值记为 M1 送到寄存器 F8，释放占用的 Load 缓冲区空间；

第 2 条指令 L.D F4, 16(R4)还在执行，Load2 已经从要取数地址取得了内容；

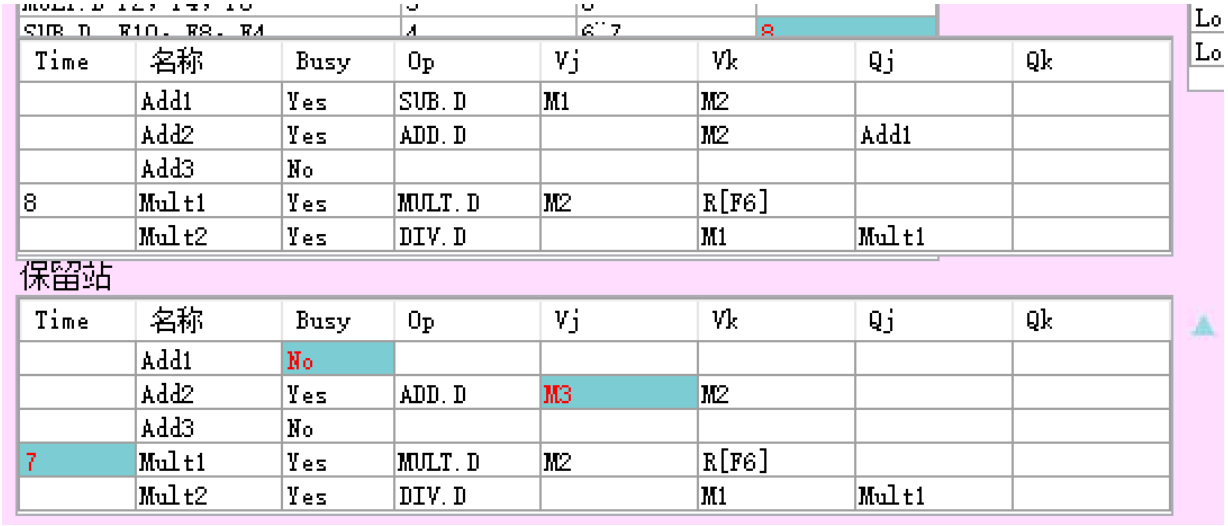
第 3 条指令 MULT.D F2, F4, F6 因为没等到 R4 停在发射状态；

由于保留站还有空间，所以第 4 条指令 SUB.D F10, F8, F4 发射，保留站 Add1 的 Busy 字段改为 yes，Op 填入 SUB.D，Vj 源操作数填入 M1，由于 F4 还没被 Load2 算出来，所以 Vk 为空，Qk 填入 Load2；



3. 结合相关指令举例说明 Tomasulo 算法是如何处理 RAW 相关的?

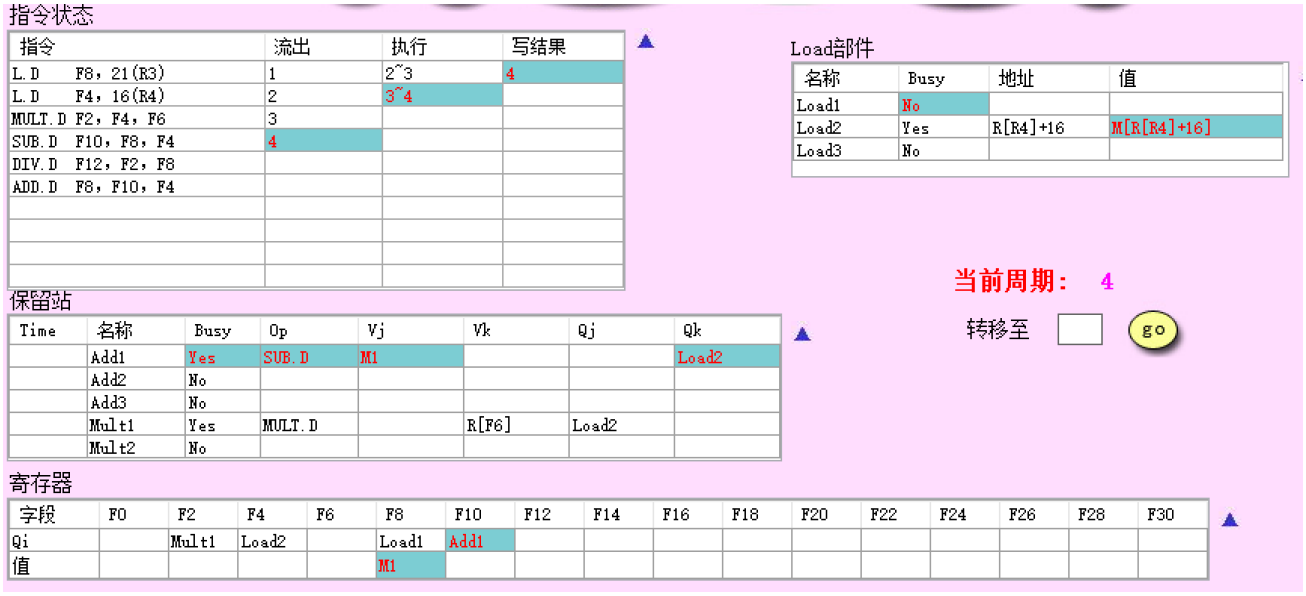
答：Tomasulo 算法避免 RAW 相关的方式是在执行状态检测，待所有需要的操作数都就绪后再执行；例如下图是 SUB.D F10, F8, F4 执行前后的保留站情况对比，可以看到保留站在 Vj 和 Vk 两个操作数齐全后才真正执行了读数再减的过程，消除了 RAW 相关；



4. Tomasulo 算法是如何实现寄存器重命名的？举例说明。

答：Tomasulo 算法用虚拟寄存器（保留站、Load 部件）的名称代替有限的真实寄存器，具体操作是用数值或者保留站的名称代替指令中的真实寄存器，实现了寄存器重命名；

例如，第 4 周期时，可以看到寄存器情况如下：F2 的结果在 Mult1，F4 的结果在 Load2，F8 的结果在 Load1，Load1 的值记为 M1，F10 的结果在 Add1；重命名之前需要依赖 F4 的指令（如 Mult1，被重命名前是 MULT.D　F2,F4,F6）都改为依赖 Load2。

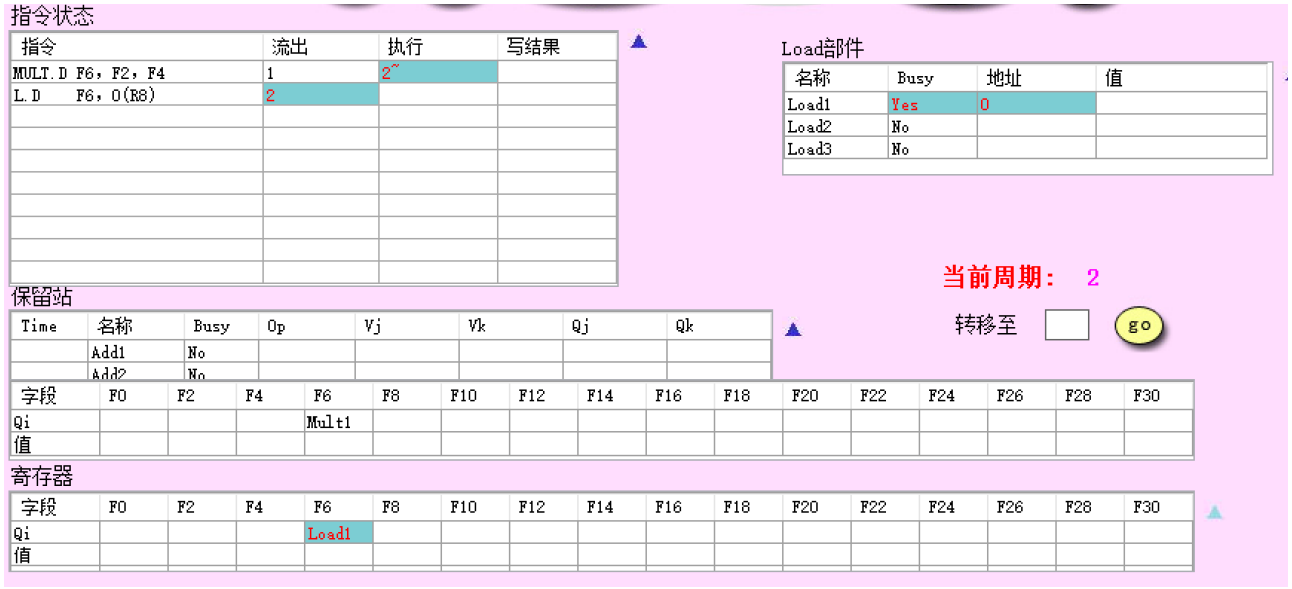


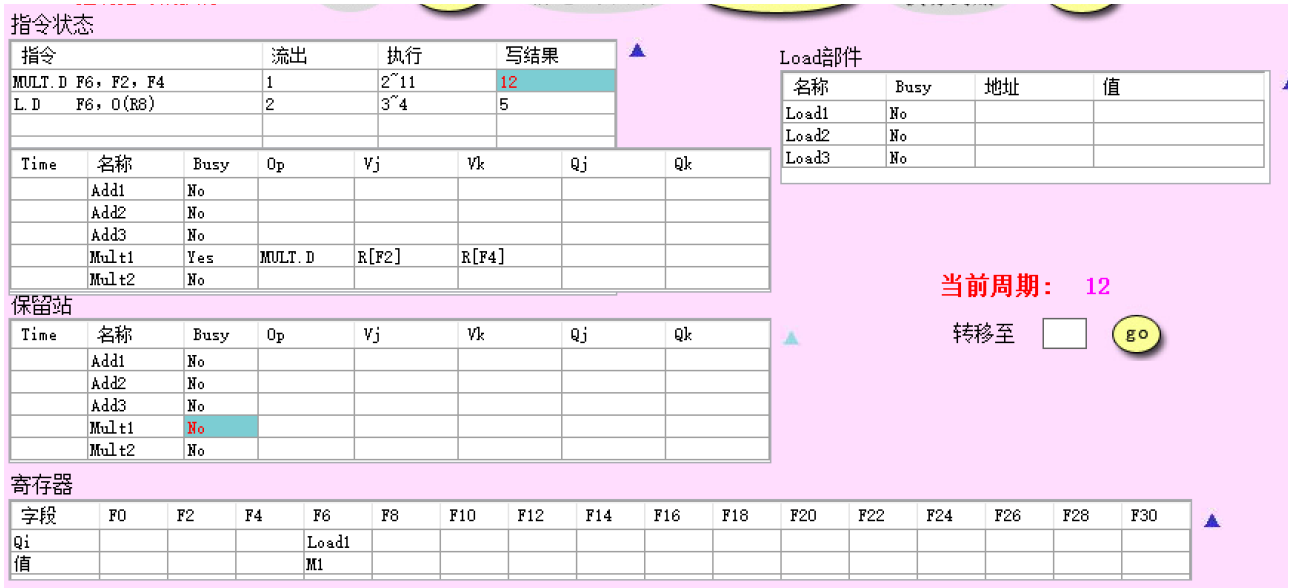
5. Tomasulo 算法能避免哪些数据相关？结合相关指令举例说明 Tomasulo 算法是如何消除这些数据相关的？（请结合指令状态、保留站、load 部件、寄存器的状态变化进行详细分析。可自行对指令进行修改。）

答：Tomasulo 算法能避免 RAW 相关、WAW 相关、WAR 相关。

其中，WAW 相关、或 WAR 相关的两条指令之间，并没有数据的生产-消费关系，也被称为名相关，因此在发射阶段用寄存器重命名，保证指令按原序发射就解决了。

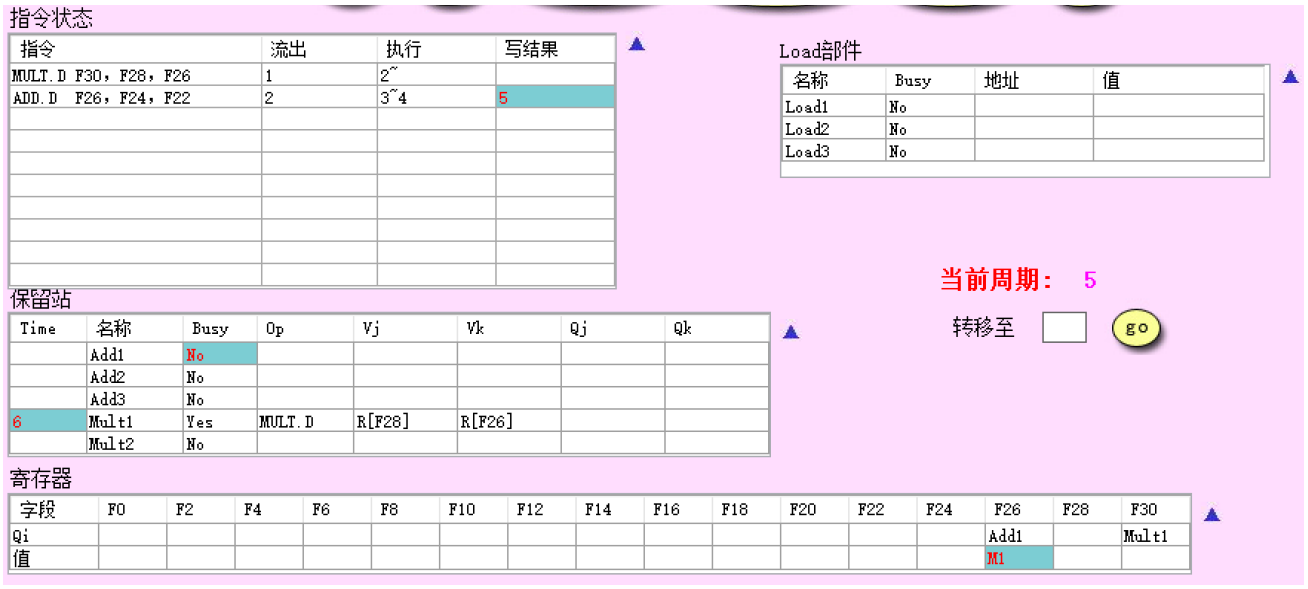
WAW 相关的例子如下图所示：这里 MULT.D　F6,F4,F2 和后面的 L.D　F6,0(R8)都写入 F6，且 L.D 执行时间远小于 MULT.D，假如按原始的流水线执行，后完成的 MULT.D 会覆盖 F6 寄存器的结果，把 MULT.D 的 F6 重命名为 Mult1，就消除了 WAW 相关。

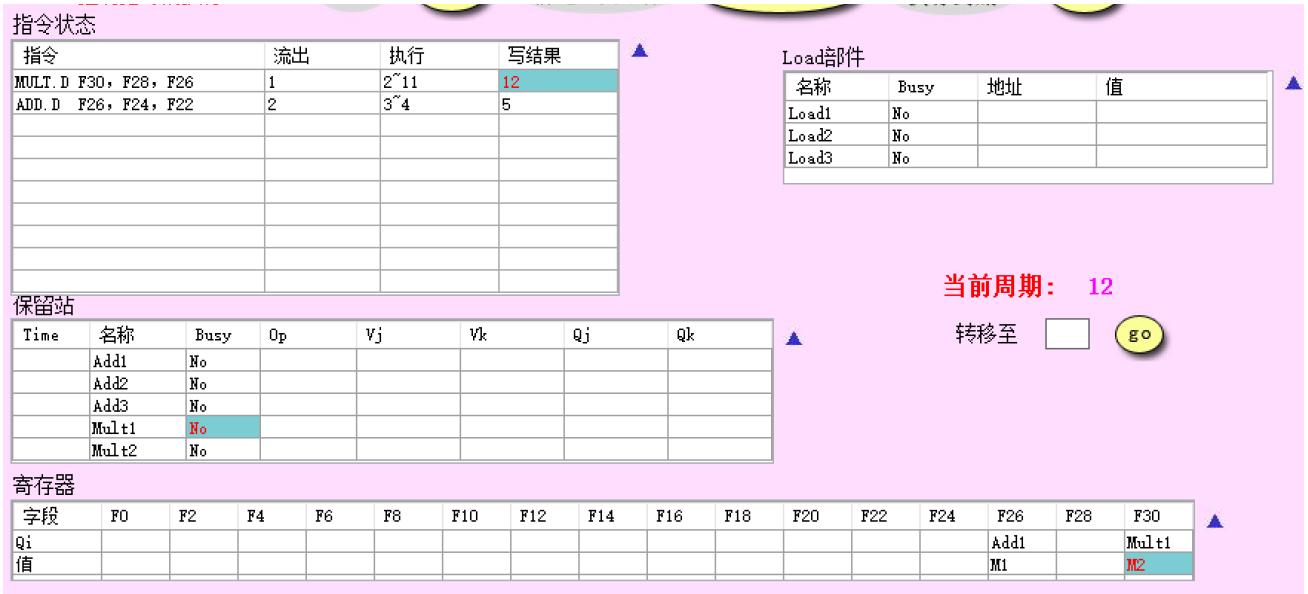




WAR 相关的例子如下图所示：这里 MULT.D　F30,F28,F26 和后面的 ADD.D F26,F24,F22 若按原始的流水线执行，MULT.D 要先读入 F26，而执行更快的 ADD.D 会抢先写入 F26。可以发现，只要让这两条指令中的同一个 F26 寄存器换成两个即可解决相关。

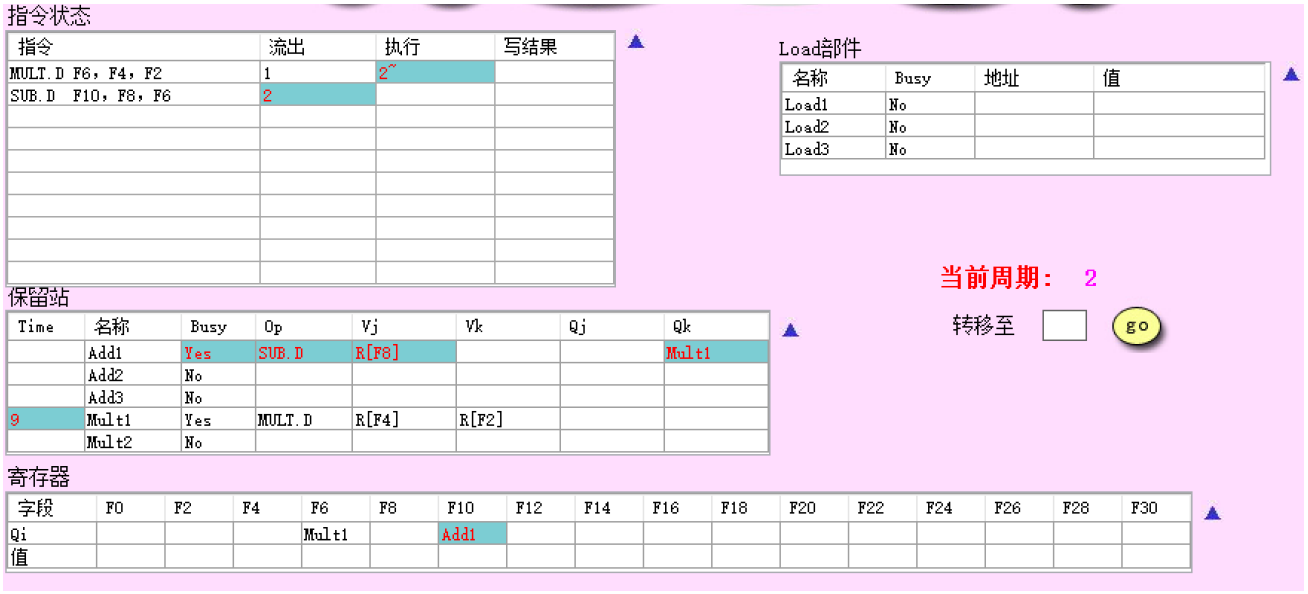
此时，Tomasulo 算法的处理是，在 MULT.D 先发射时，就把 MULT.D 的 F26 重命名为 Mult1，Mult1 用到的源操作数 Vk 保存了 F26 寄存器的值副本，后面 ADD.D（重命名为 Add1）写入 F26 也不会影响 F26 旧值的副本，从而消除了 WAR 相关；

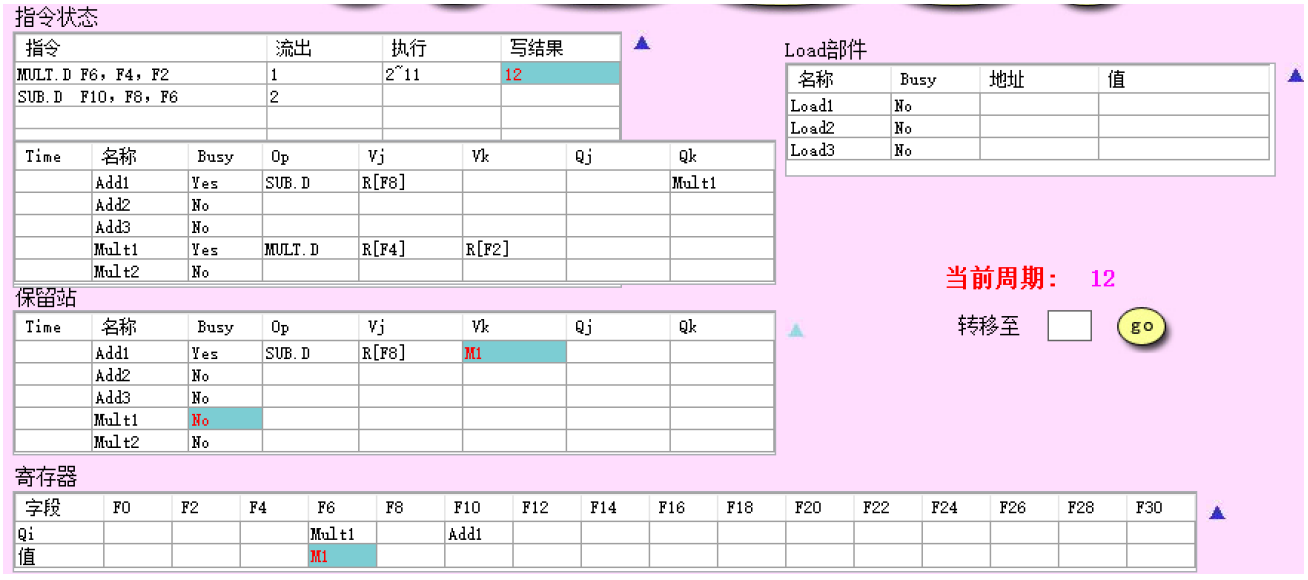




而 RAW 相关是在执行阶段解决的，例子如下图所示：这里 MULT.D　F6,F4,F2 要写入 F6，SUB.D　F10,F8,F6 要读 F6 的值，Tomasulo 算法的处理办法是在 SUB.D（它的 F10 重命名为 Add1）的 Qk 字段填入 Mult1，表示等待 Mult1 保留站提供源操作数；

Mult1 完成后，Add1 的 Qk 字段为空，Vk 字段填入之前 Mult1 保留站广播的值 M1，Vj, Vk 两个源操作均已就绪，下一步 Add1 才进入执行状态。





三、 心得体会

通过本次实验的，我观察 Tomasulo 模拟器的运行，加深了对 Tomasulo 算法硬件结构和执行顺序的理解。Tomasulo 算法在硬件上使用了共享数据总线 CDB 作为数据缓冲，通过保留站、缓冲器的各字段情况实现了检测和消除相关，提高了更多乱序执行的机会，从而提高了流水线效率。

对于 Tomasulo 算法而言，RAW 相关与 WAW 和 WAR 相关有本质区别，RAW 更像是真正的“数据相关”，所以 RAW 相关不能仅用寄存器重命名解决，而是要在执行时检测源操作数是否全部就绪。

注意：

1.熟悉 Tomasulo 算法模拟器的使用，该模拟器已上传至课程平台。

2.希望通过实际操作熟悉我们课程上学习的 Tomasulo 算法详细流程，以及该算法是如何处理或消除各类数据相关。

3.实验报告的内容应该充实，细致，不要过于简单化。要注明实验名称、姓名、班级、学号等必要信息。

4.实验报告可提交 word 或者 pdf 格式文档,文档命名为:学号姓名_第三章第二次实验，并在截止日期前提交到课程平台。