A.5 实验五 Tomasulo算法实验报告

**A.5.1 实验目的**

1. 加深对指令级并行性及其开发的理解。
2. 加深对Tomasulo算法的理解。
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。
4. 掌握采用了Tomasulo算法的浮点处理部件的结构。
5. 掌握保留站的结构。
6. 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。

**A.5.2 实验平台**

实验平台采用Tomasulo算法模拟器。

环境的建立：见A.0。

**A.5.3 实验内容及步骤**

首先要掌握Tomasulo模拟器的使用方法。（见使用说明：点击模拟器中标题右侧的小圈）

1. 假设浮点功能部件的延迟时间为加减法2个时钟周期，乘法10个时钟周期，除法40个时钟周期，Load部件2个时钟周期。

（1）对于下面的代码段，给出当指令MUL.D即将确认时，保留站、load缓冲器以及寄存器状态表中的内容。

L.D F6, 24（R2）

L.D F2, 12（R3）

MUL.D F0, F2, F4

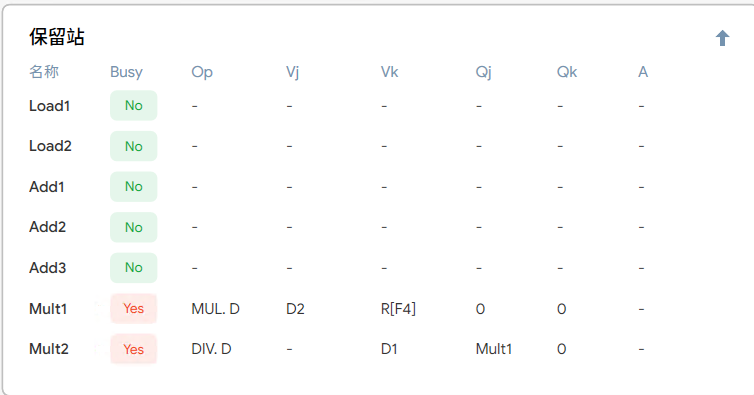
SUB.D F8, F6, F2

DIV.D F10, F0, F6

ADD.D F6, F8, F2

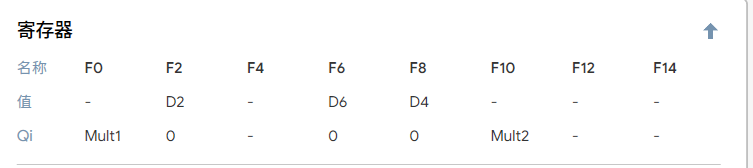
实验结果：

1. 保留站的内容、分析



当MUL.D即将确认时,表示MUL.D的数值都已经读取完毕其中，Vj值为D2,Vk值读取的是寄存器F4中的值。

1. 寄存器状态表的内容，分析



看F0，Qi为Mult1,表明MUL.D写出寄存器为F0,F4没有值，表示为取值到寄存器中，所以Vk中读取不到F4的值。

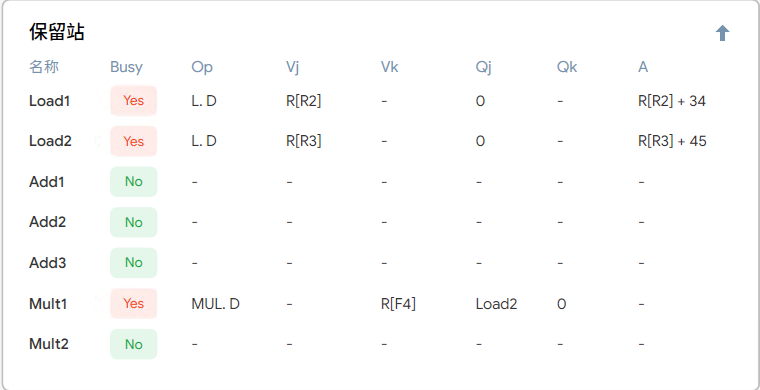
（2）按步进方式执行上述代码，利用模拟器的“向上箭头”（鼠标摸上去）的对比显示功能，观察每一个时钟周期前后各信息表中内容的变化情况。

2. 对于与上面相同的延迟时间和代码段。

1. 给出在第3个时钟周期时，保留站的内容.

结果：

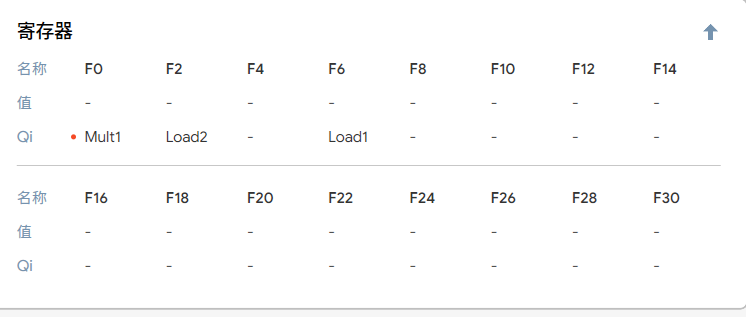
1、保留站的内容、分析



分析：

此时在取出第三条指令MUL.D F0,F2,F4，而且Load2中在执行L.D F2,45(R3)取值中，而且F2为MUL指令中需要使用的值，因此使用了Load2寄存器换名技术，当Load2完成写结果后直接把Load2中的数值写入到Vj中。

1. 寄存器状态表的内容，分析



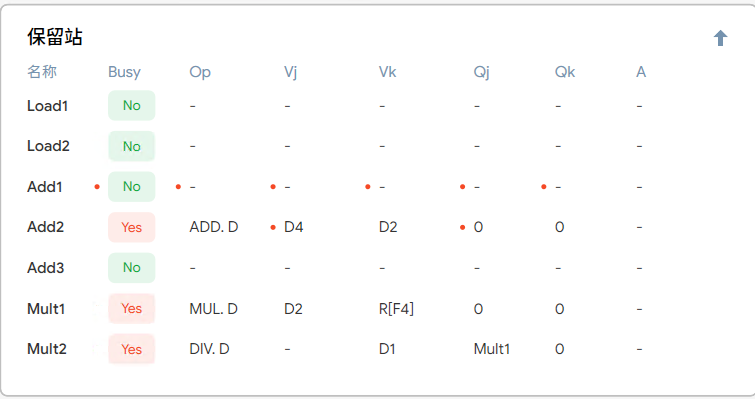
分析：看到了F0中的Mult1,F2中的Load2和F6中的Load1，分别都是了寄存器换名技术，

用来表示写入结果。

2、步进5个时钟周期，给出这时保留站、load缓冲器以及寄存器状态表中的内容。

实验结果：

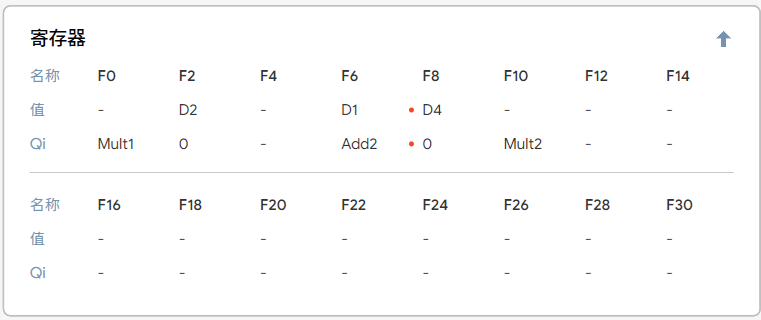
1、保留站的内容、分析



分析：

此时Add1 中SUB.D F8,F2,F6指令中完成写入，而Add2中Qj中使用Add1的换名也可以换成寄存器结果D4。

1. 寄存器状态表的内容，分析



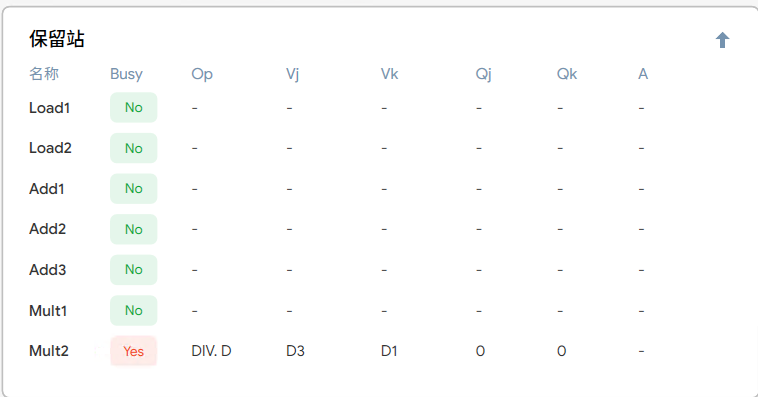
分析：

F8中刚计算完值，把D4值写入。因而将F8的Qi设置为0，表示当前没有正在执行的指令要写该寄存器。

再步进10个时钟周期，给出这时保留站、load缓冲器以及寄存器状态表中的内容。

实验结果：

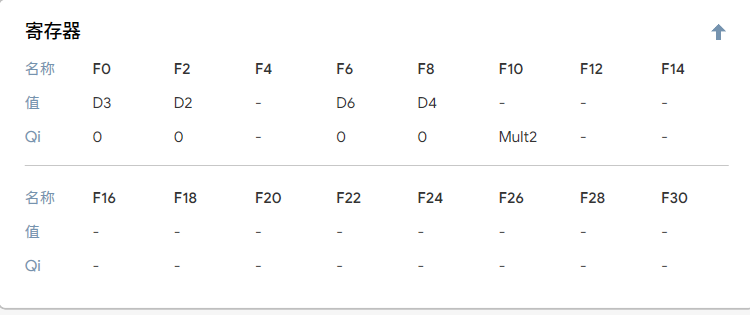
1. 保留站的内容、分析



分析:

在CP16时MUL.D完成写入操作，在17到56中只有DIV.D这个40个执行周期的指令在执行。

1. 寄存器状态表的内容，分析



分析：

寄存器F10中Qi写入保留字中待运行命令的名称-Mult2。

问题 1、执行完6条指令最终用多少个周期？（给出计算步骤）

加减法2个时钟周期，乘法10个时钟周期，除法40个时钟周期，Load部件2个时钟周期。

①L.D F6, 24（R2）执行：第1个周期开始，第4个周期结束。

②L.D F2, 12（R3）执行： 第2个周期开始，第5个周期结束。

④SUB.D F8, F6, F2需要等待第②条执行完毕：第6个周期开始，第8个周期结束。

//同时开始因为功能组件上没有冲突，而使用的数据只存在RAR冲突，可以同时进行。

③MUL.D F0, F2, F4需要等待第②条执行完毕：第6个周期开始，第16个周期结束。

⑥ADD.D F6, F8, F2需要等待第④条执行完毕：第9个周期开始，第11个周期结束。

//ADD中有F8需要等待sub执行完写入结果后开始执行。

⑤DIV.D F10, F0, F6需要F0,F6而F0需要等待第③条执行完毕：第17开始，第57周期结束。

因此最终用了57个周期

2、6条指令流出的顺序是什么？

①②③④⑤⑥

1. 上述指令中DIV.D F10,F0,F6和ADD.D F6,F8,F2的WAR冲突是如何消除的?

因为DIV.D指令中的源操作数F0在MUL.D指令中还没有写入，DIV.D需要等待MUL.D写入完毕才可以执行指令。而ADD.D指令在SUB执行完毕后没有WAR冲突,可以执行。所以ADD.D执行操作，而当MUL.D写入完成后，DIV.D指令的源操作数F6又因为ADD.D在执行中，没有写入到寄存器中。所以DIV.D指令会被停止，等待ADD.D指令写入完毕才可以指令，从而消除了WAR冲突。

1. 假设浮点功能部件的延迟时间为加减法3个时钟周期，乘法8个时钟周期，除法40个时钟周期。自己编写一段程序（要在实验报告中给出），重复上述步骤2的工作。

修改为自己的。

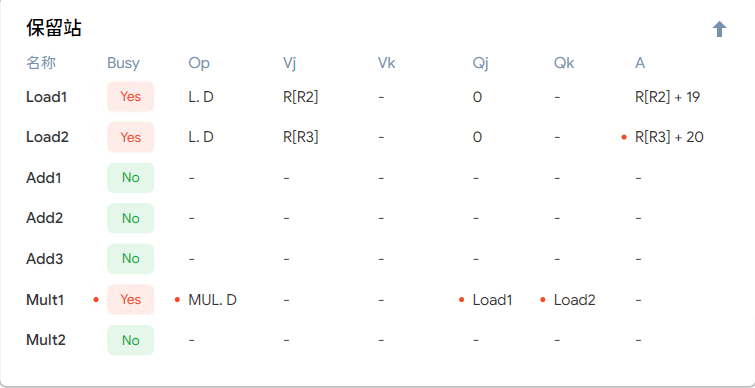


2. 对于与上面相同的延迟时间和代码段。

1. 给出在第3个时钟周期时，保留站的内容.

结果：

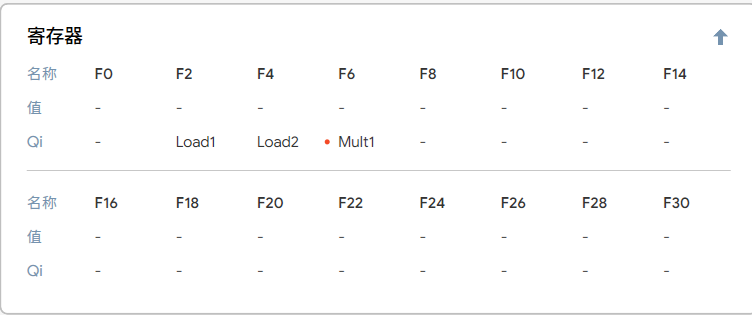
1、保留站的内容、分析



分析：

此时在流出第三条指令MUL.D F6,F2,F4，而且Load2中在执行L.D F4,20(R3)取值中，而且F2和F4为MUL指令中需要使用的值，因此使用了Load1，Load2寄存器换名技术，当Load1,Load2完成写结果后直接把寄存器中对应Load1，Load2中的数值到Vj,Vk中。

1. 寄存器状态表的内容，分析



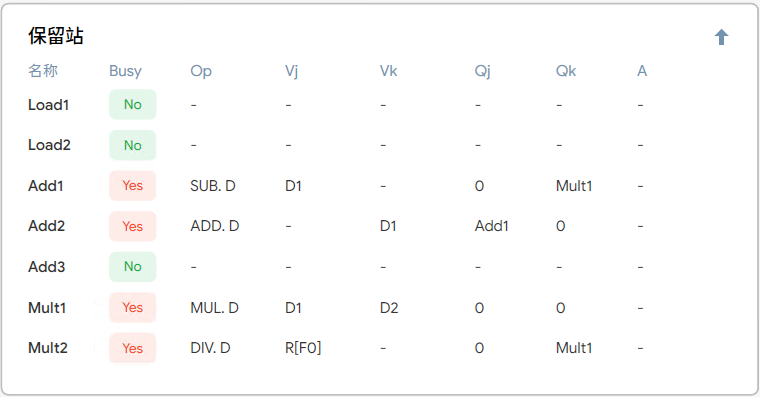
分析：看到了F0中的Mult1,F2中的Load2和F6中的Load1，分别都是使用寄存器换名技术，

用来表示写入结果。

2、步进5个时钟周期，给出这时保留站、load缓冲器以及寄存器状态表中的内容。

实验结果：

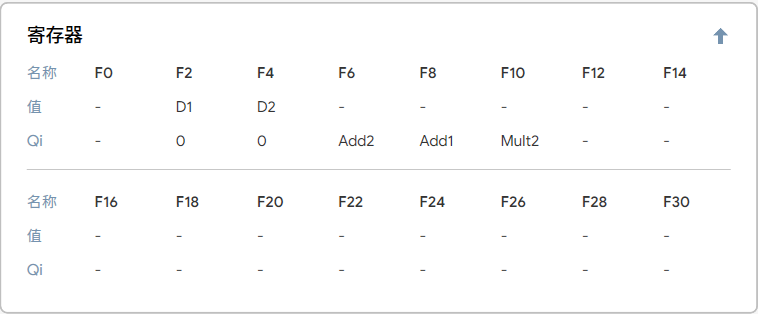
1、保留站的内容、分析



分析：

此时MUlt1 中MUL.D F6,F2,F4指令中正在执行，而add1,mult2模块不能执行是因为他们使用的值都F6都要等mult1执行完毕写入后才能获取到，而add2也是类似的情况，要等add1执行完毕写入完毕才能执行。

1. 寄存器状态表的内容，分析



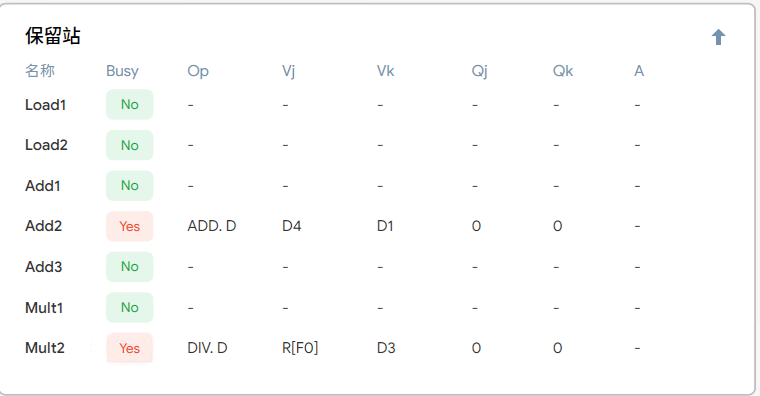
分析：

看到F6中MUL.D和ADD.D都有写入，而寄存器中的F6Qi对应的为Add2说明Qi值被覆盖了，应该使用了寄存器换名技术，所以其他可以获得mult1执行之后F6的值。

再步进10个时钟周期，给出这时保留站、load缓冲器以及寄存器状态表中的内容。

实验结果：

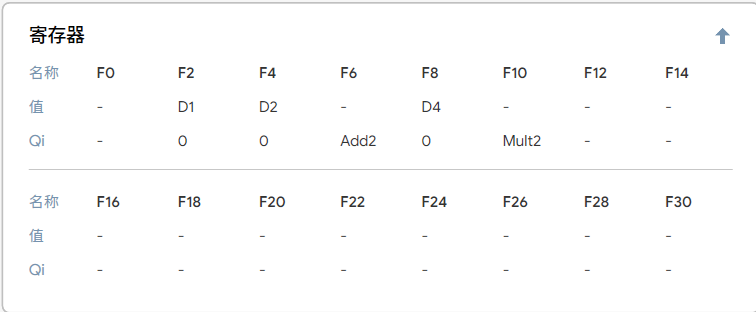
1. 保留站的内容、分析



分析:

之前情况下在CP16时MUL.D完成写入操作，在17到56中只有DIV.D这个40个执行周期的指令在执行。  
而现在18时SUB.D刚完成写入操作，而SUB.D写入的F8为ADD.D中需要的源操作数。

1. 寄存器状态表的内容，分析



分析：

寄存器F10中Qi写入保留字中待运行命令的名称-Mult2。

寄存器F6中Qi写入保留字中待运行命令的名称-Add2。

问题 1、执行完6条指令最终用多少个周期？（给出计算步骤）

加减法3个时钟周期，乘法8个时钟周期，除法40个时钟周期，Load部件2个时钟周期。

①L.D F2, 19（R2）执行：第1个周期开始，第4个周期结束。

②L.D F4, 20（R3）执行： 第2个周期开始，第5个周期结束。

③MUL.D F6, F2, F4需要等待第①②条执行完毕：第6个周期开始，第14个周期结束。

//这里SUB需要等待MUL.D执行完。

④SUB.D F8, F2, F6需要等待第③条执行完毕：第15个周期开始，第18个周期结束。

//ADD中有F8需要等待sub执行完写入结果后开始执行。

⑥ADD.D F6, F8, F2需要等待第④条执行完毕：第19个周期开始，第22个周期结束。

⑤DIV.D F10, F0, F6需要F0,F6而F0需要等待第③条执行完毕：第15个周期开始，第55周期结束。

因此最终用了55个周期。

2、6条指令流出的顺序是什么？

①②③④⑤⑥