

《 计算机体系结构 》

实验报告二

|  |  |
| --- | --- |
| 班 级： | **计203** |
| 学 号： | **20002462** |
| 姓 名： | **刘子言** |
| 指导教师： | **梁建宁** |

信息科学与工程学院

2022年 11月

**实验名称** Tomasulo算法

**实验地点** 信息楼418 **实验日期** 2022.11.23

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、写回各阶段对浮点指令和Load/ Store指令进行的操作；

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。

1. **实验设备**

个人PC机，实验平台采用Tomasulo算法模拟器。

1. **实验原理**

1、指令集并行性及其开发

（1）由于指令可以并行执行，所以指令之间可能实现的重叠执行称为指令级并行（ILP）。

（2）基础块是一段顺序执行的代码，除了入口外没有其他的转入分支，除了出口外没有其他的转出分支。一个基础块一般包含3-6条指令，基础块类的指令一般互相依赖，挖掘并行性很受限制，所以一般需要在不同的基础块之间寻找指令并行性。

（3）指令级并行大体有两种不同的开发方法：一是依靠硬件来帮助动态发现和开发并行，二是依靠软件技术在编译时静态发现并行。

（4）挖掘指令级并行性的目的：最大化IPC，或者最小化CPI。

2、Tomasulo算法基本步骤

（1）发射

从FIFO指令缓冲栈中获取下一条指令，如果保留栈有空闲，将指令发射到保留栈中；如果操作数尚未获得，暂时让指令停顿；重命名寄存器，能够消除WAR、WRW冒险。

（2）执行

当操作数可用时，将其保存到正等待该数的保留栈中；当所有操作数都准备好了，执行指令（要避免RAW冒险）；为了防止存储器冒险，Load和Store指令维持原程序中的顺序；为了维持异常行为跟原来一样，指令不允许开始执行，一直到前面所有的分支指令都完成。

（3）写回

通过公共数据通道CDB将结果写到保留栈和寄存器堆；Store 指令保留在Store缓冲栈，需要一直等到存储值和存储地址可用为止，然后在有空闲存储器单元时，立即写入结果。

1. **实验操作及运行结果**

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即将写回时，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)

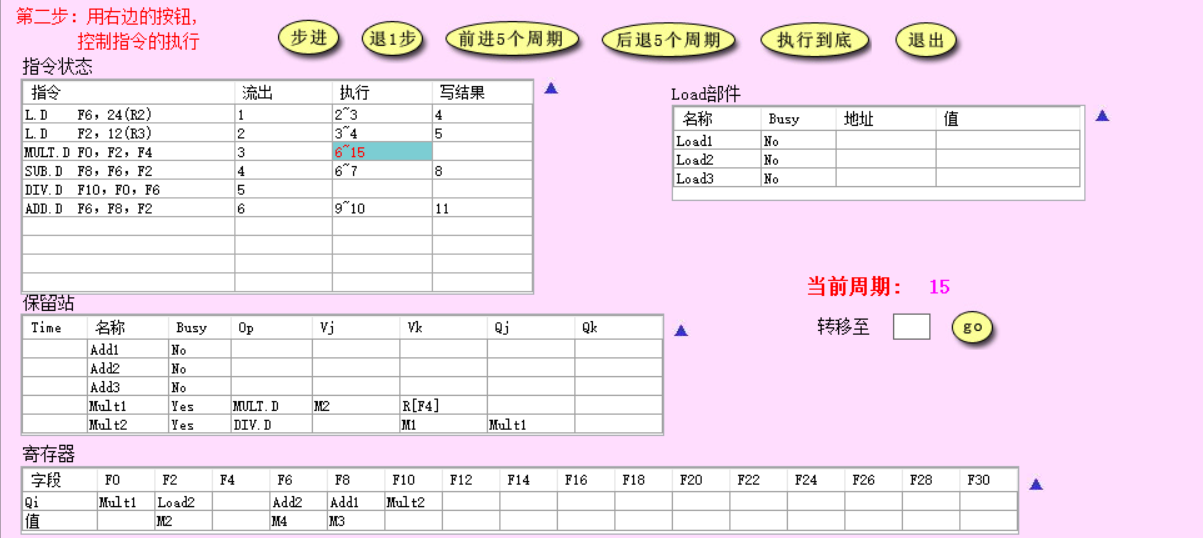
MUL.D F0, F2, F4

SUB.D F8, F6, F2

DIV.D F10, F0, F6

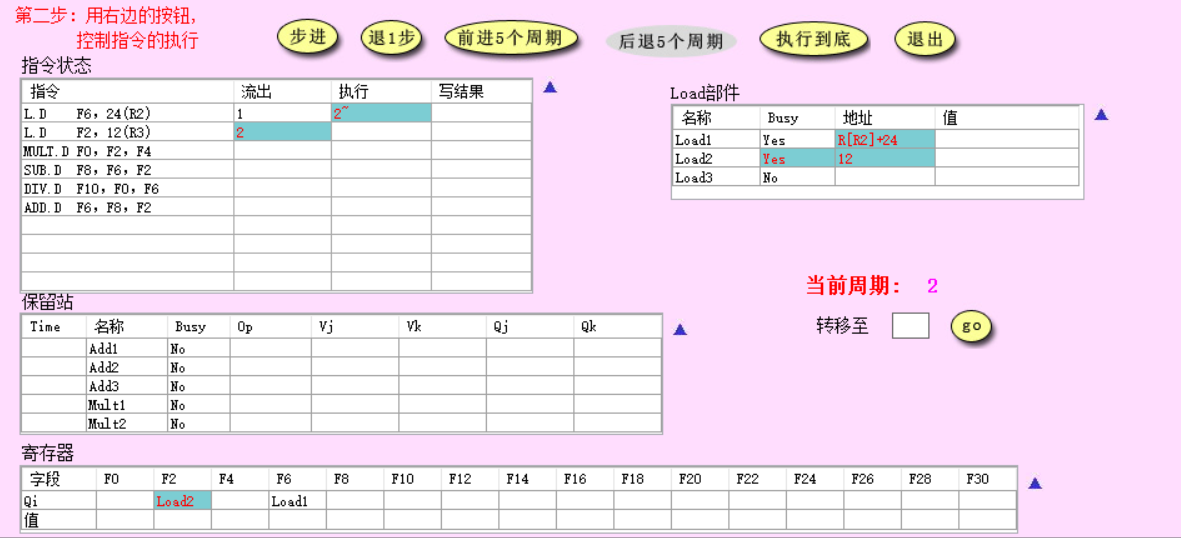
ADD.D F6, F8, F2

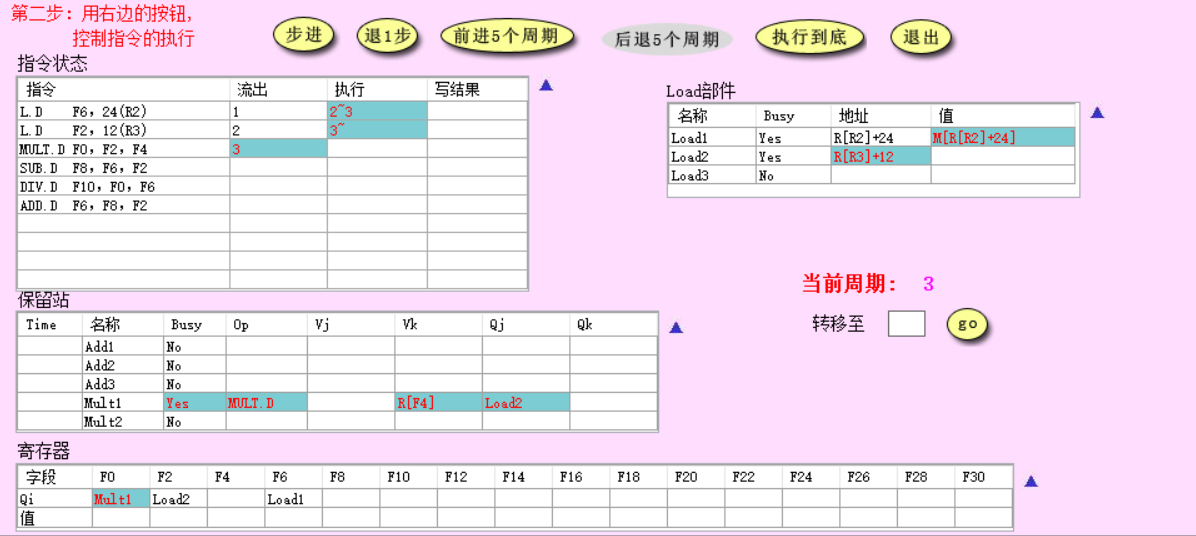
在第15个时钟周期时，指令MUL.D即将写回（该指令在第16个时钟周期写结果），其保留站、Load缓冲器以及寄存器状态表中的内容如下图所示：

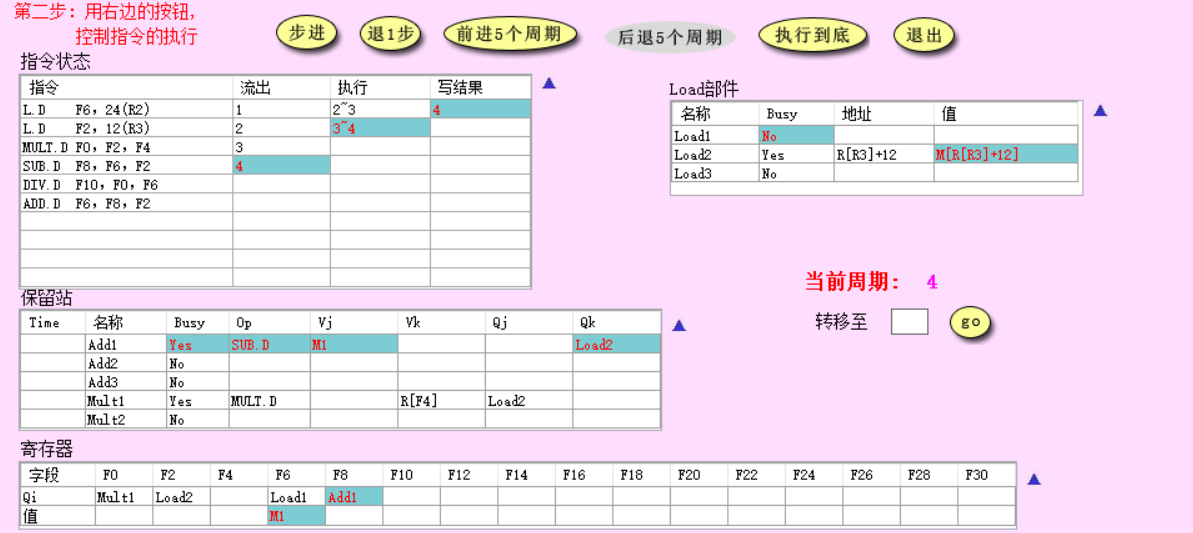


② 按步进方式执行上述代码，利用模拟器的“小三角按钮”的对比显示功能，观察每一个时钟周期前后各信息表中内容的变化情况。**（针对步进过程中的一些比较重要的周期，在截图下方都附有文字解释）**

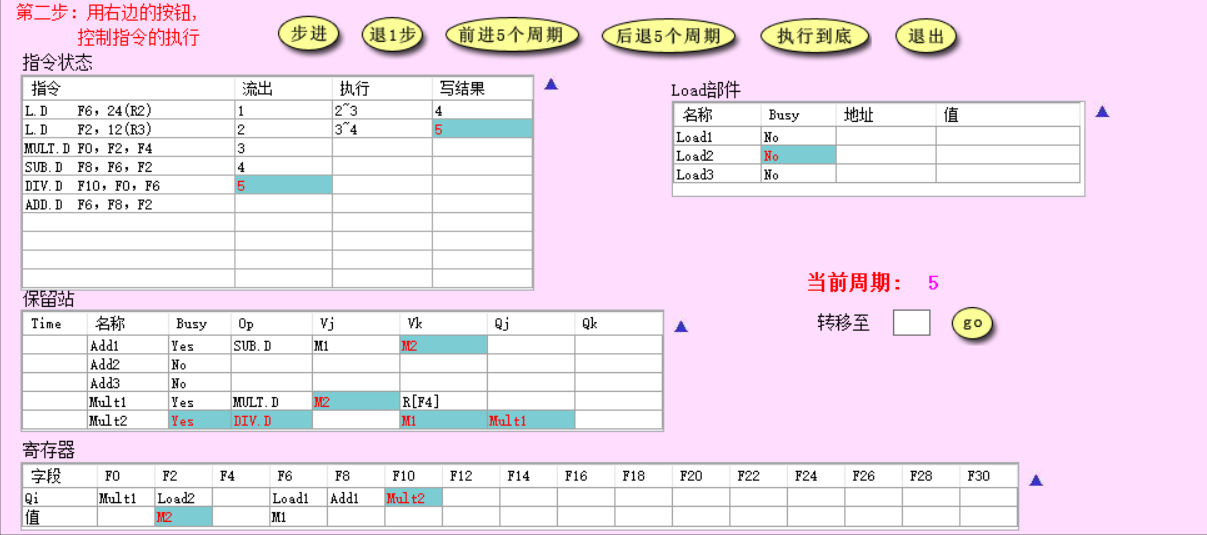




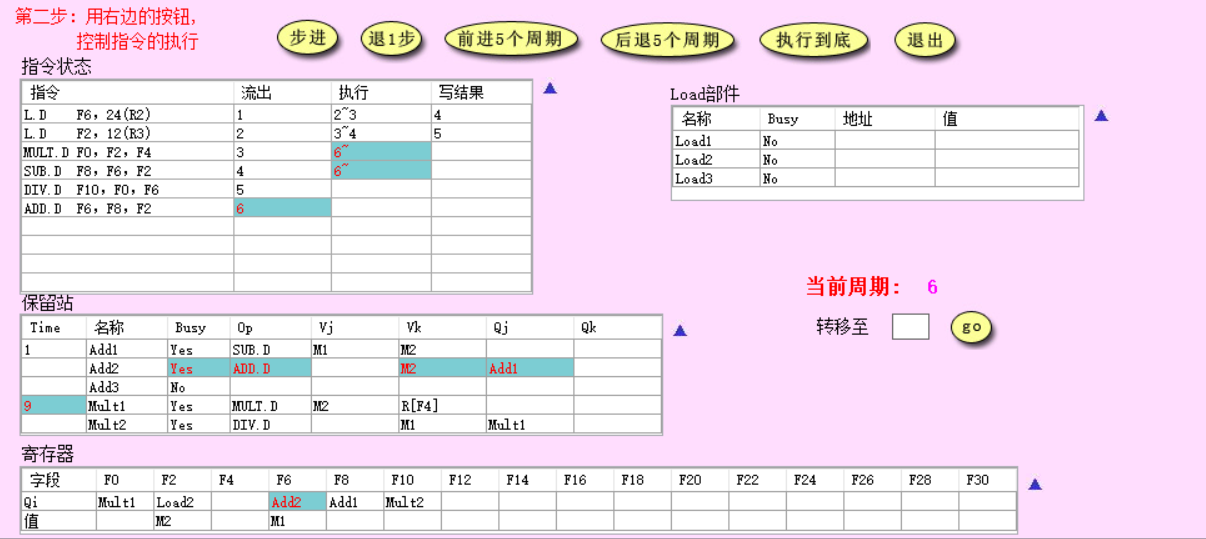




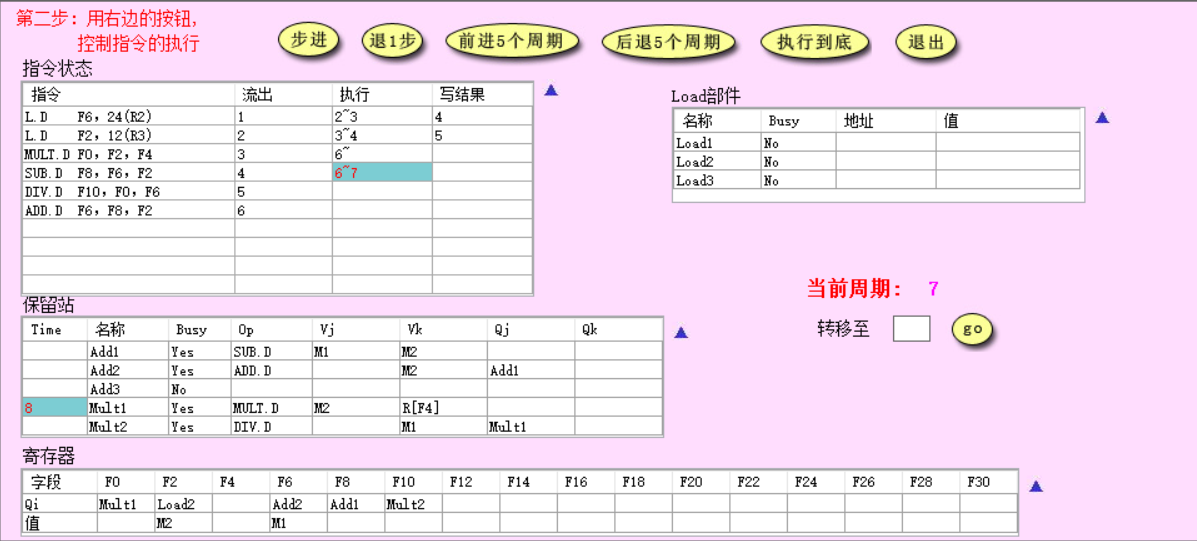
前4个周期中，第1条load指令完成了流出-执行-写结果，第2条load指令完成了流出-执行过程，尚未写回。寄存器F6、F2、F0、F8重命名为Load1、Load2、Mult1、Add1。寄存器F6的值更新，存进寄存器状态表中（M1），并在保留站中暂存MULT和SUB的相关操作数的信息。由于第2条Load指令与MULT指令和SUB指令之间存在数据相关（寄存器F2），所以第4周期开始时MULT指令还没有进入执行阶段。



到第5个周期，第2条Load指令完成写回，DIV指令流出，将寄存器F10重命名为Mult2。寄存器F2的值更新，值存进寄存器状态表中（M2）；MULT指令和SUB指令可以正常往下执行，并将对应操作数的值存进保留站中。



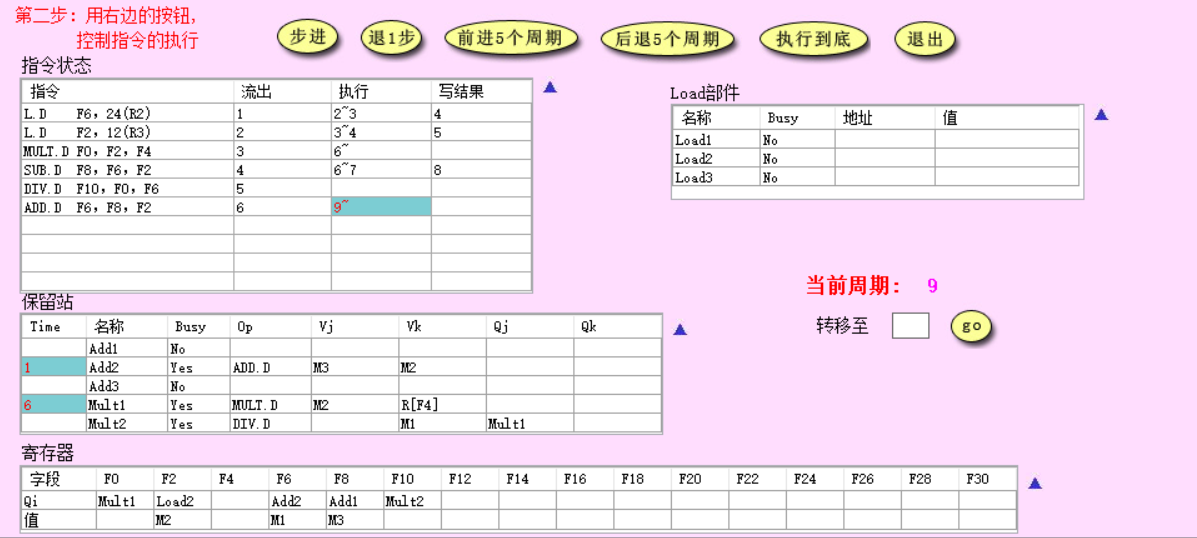
到第6个周期，寄存器F6重命名为Add2。MULT指令与DIV指令之间存在数据相关（寄存器F0），所以第6周期开始时DIV指令还没有进入执行阶段。

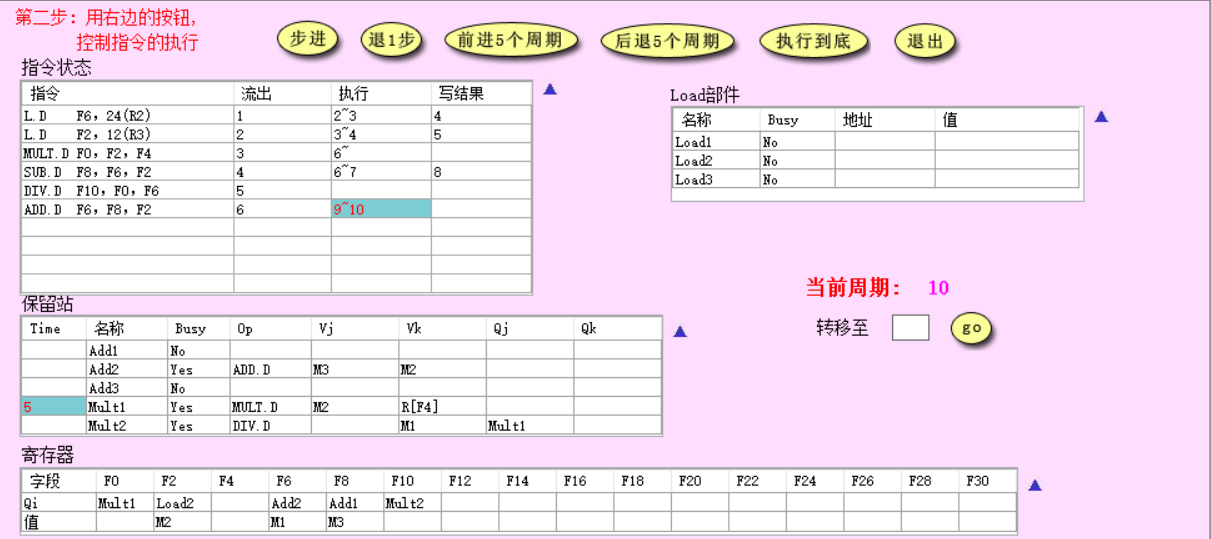


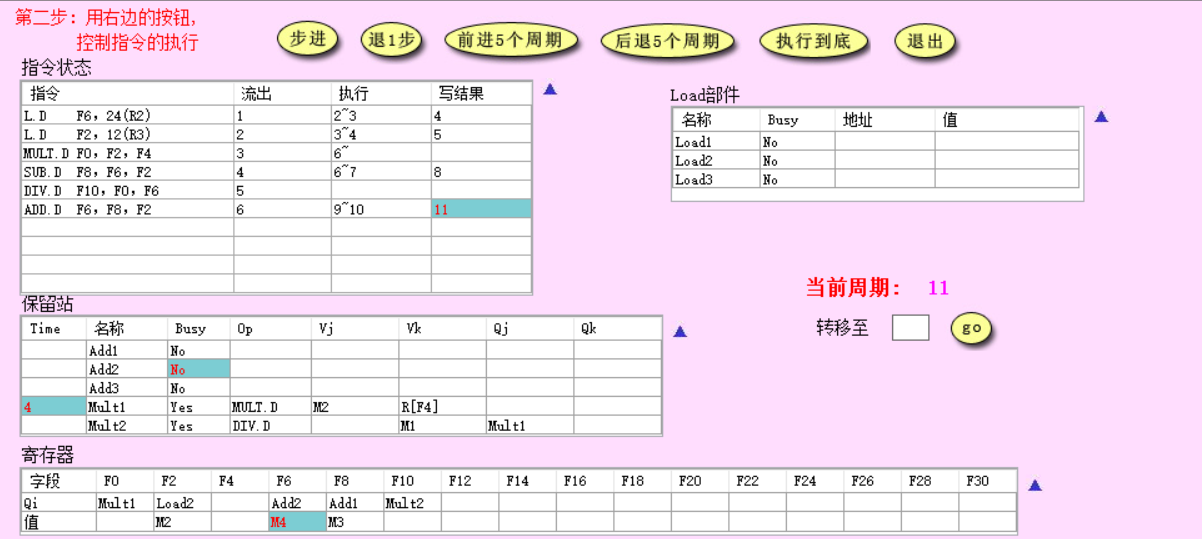
SUB指令与ADD指令之间存在数据相关（寄存器F8），所以第7周期开始时ADD指令还没有进入执行阶段。



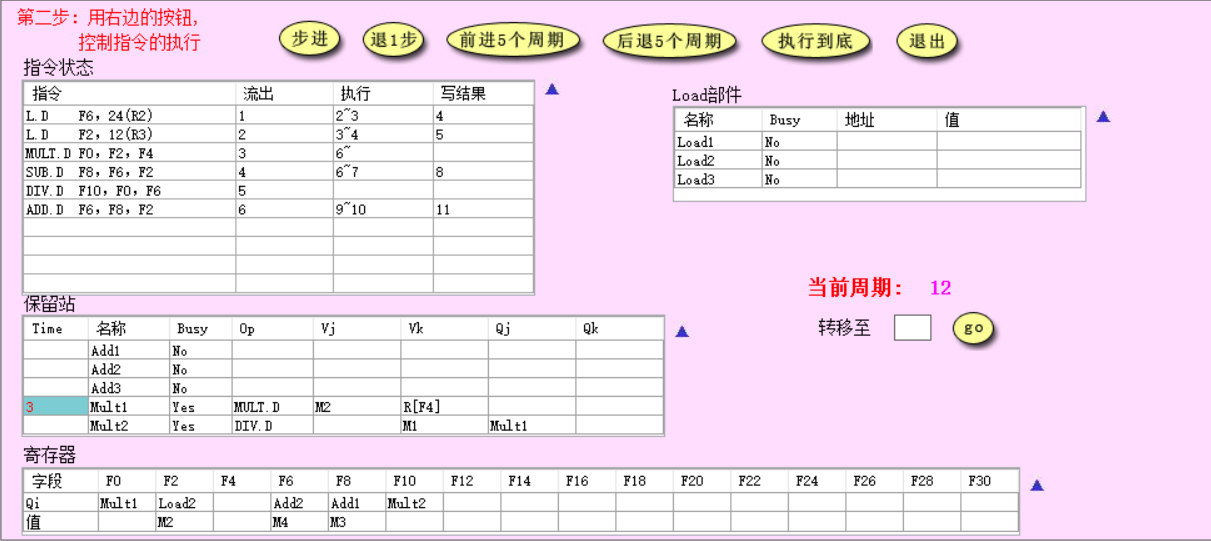
到第8个周期，SUB指令完成写结果。寄存器F8的值更新，存进寄存器状态表中（M3）；ADD指令可以正常往下执行，并将对应操作数的值M3存进保留站中。



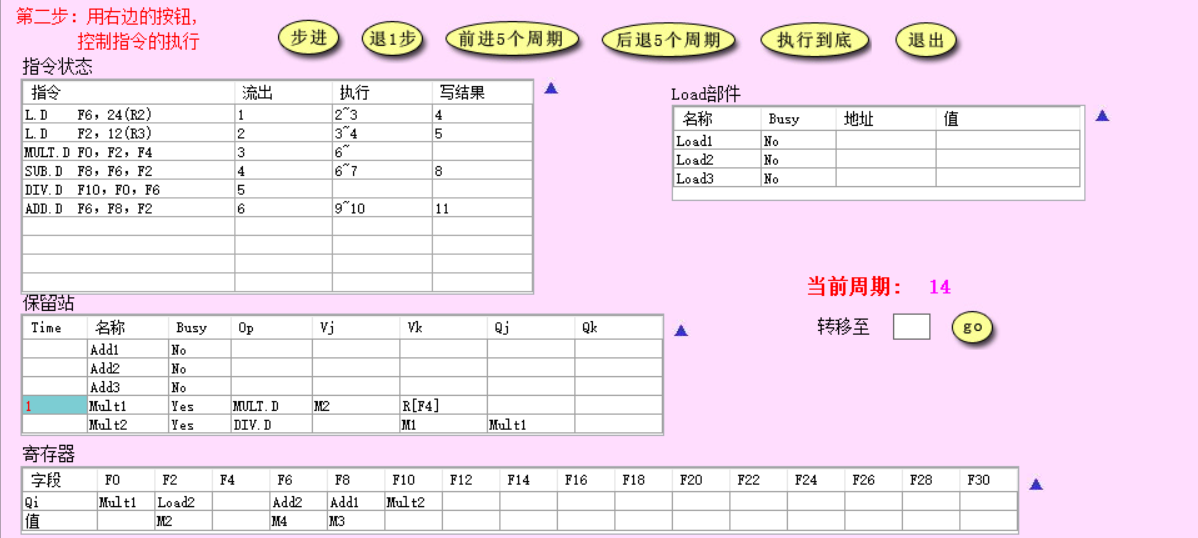


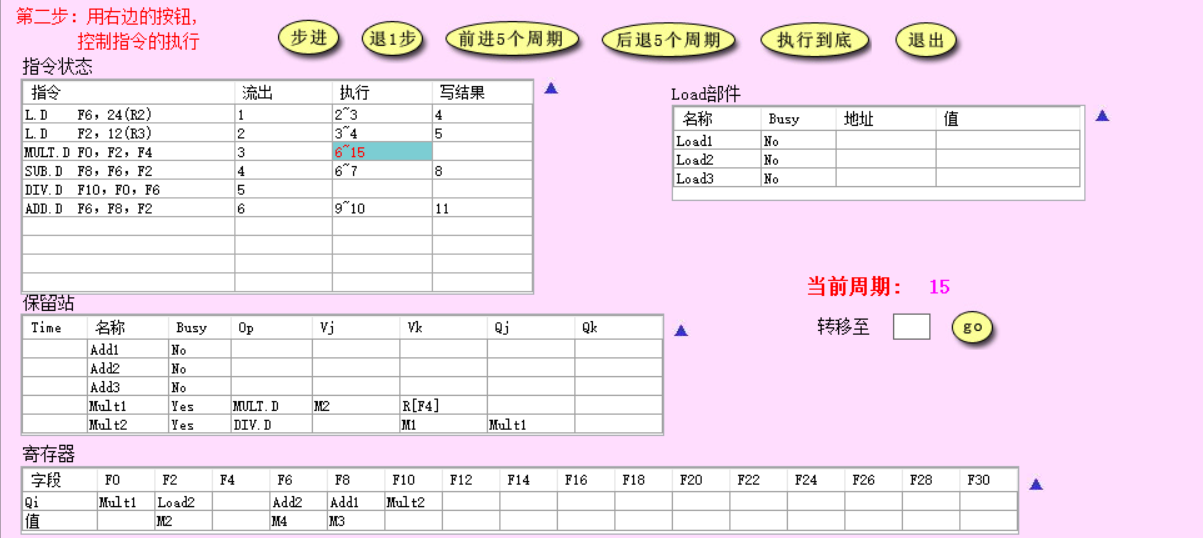


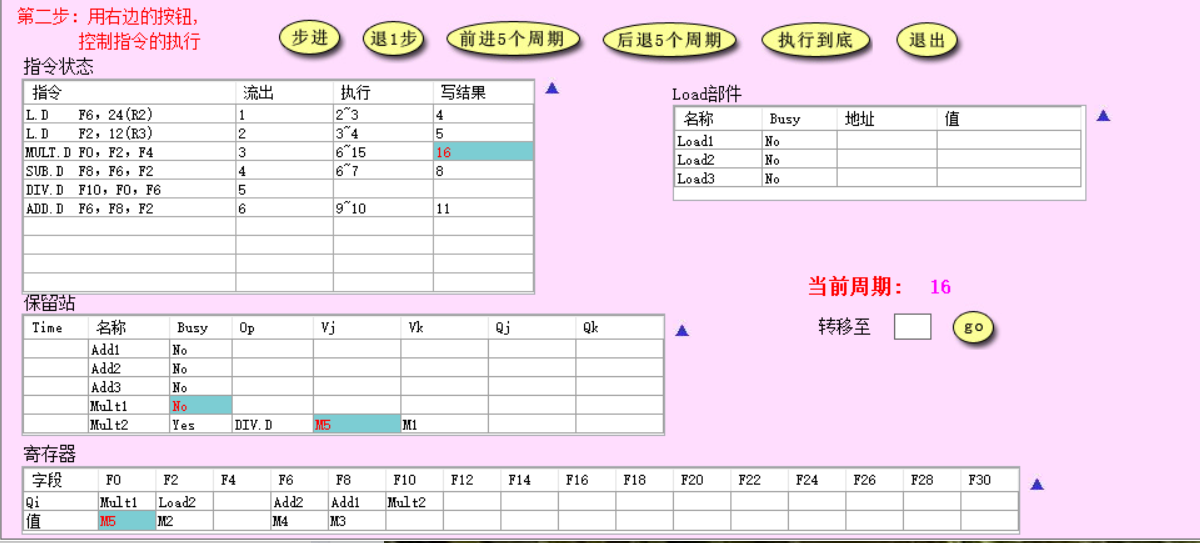
到第11个周期，ADD指令完成写结果。寄存器F6的值更新，存进寄存器状态表中（M4），保留站中清除ADD指令相关记录，Add2的状态Busy还原为No。



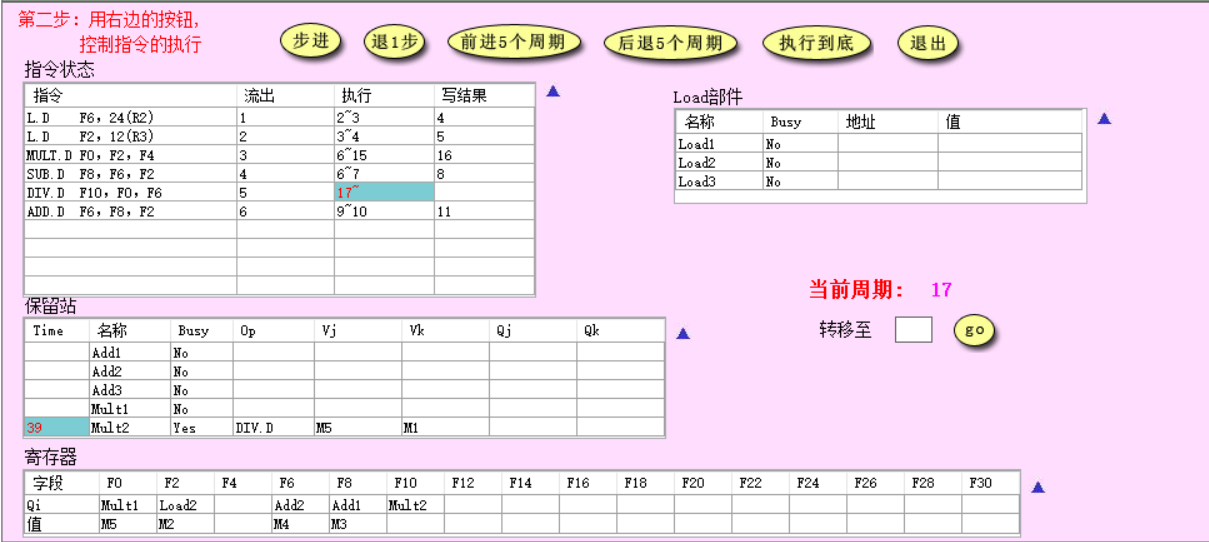




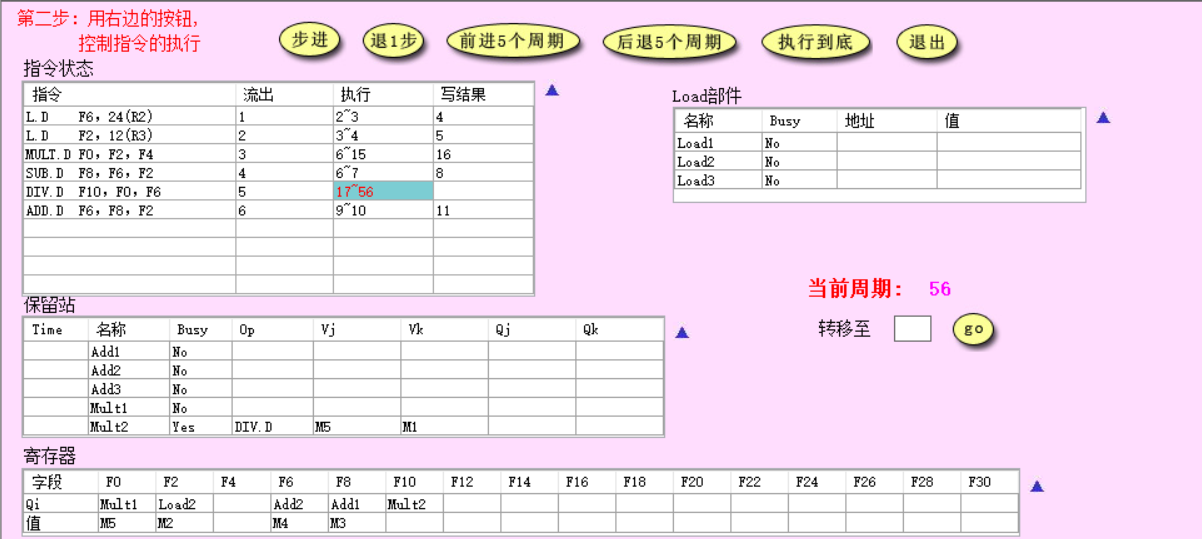




从第6-15个周期，MULT指令始终处于执行阶段（乘法需要10个周期延迟），到第16个周期，MULT指令完成写结果。保留站中清除MULT指令相关记录，Mult1的状态Busy还原为No。寄存器F0的值更新，存进寄存器状态表中（M5）；DIV指令可以正常开始执行，并将对应操作数的值M5存进保留站中,。



此处省略从第18-55个周期的截图，此阶段（第17-56周期）为DIV指令的执行阶段（除法需要延迟40个周期），其他指令、保留站、寄存器状态表、Load部件表等没有变化。

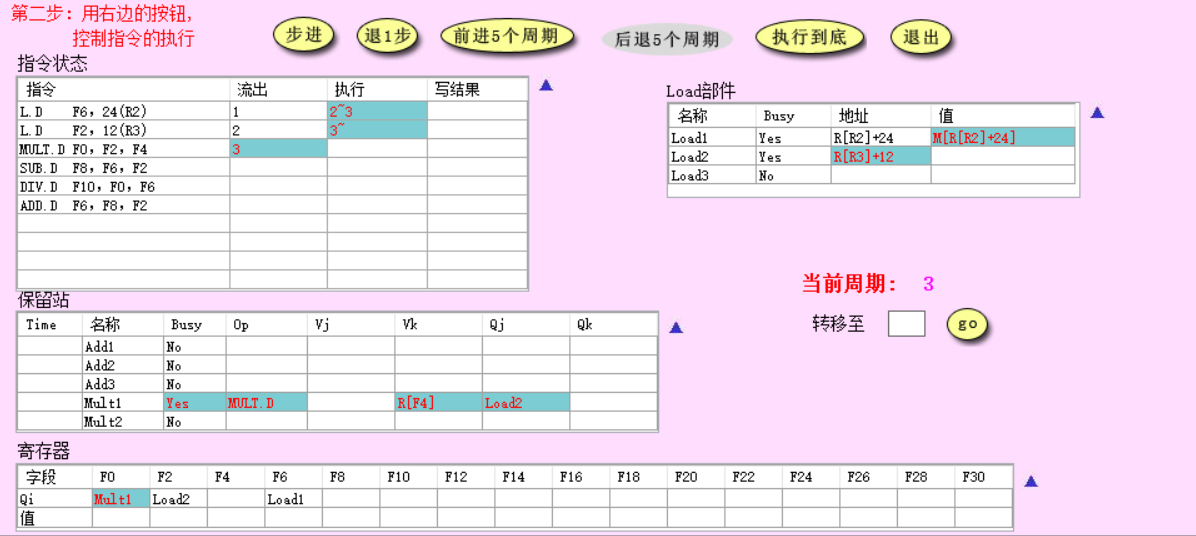




到第57个周期，DIV指令完成写结果。保留站中清除DIV指令相关记录，Mult2的状态Busy还原为No。寄存器F10的值更新，存进寄存器状态表中（M6）。至此，该代码段的所有指令全部执行完毕。

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。



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。

在①的基础上再步进5个周期，此时为第8个时钟周期，保留站、Load缓冲器以及寄存器状态表中的内容如下图：



③ 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。

在②的基础上再步进10个周期，此时为第18个时钟周期，保留站、Load缓冲器以及寄存器状态表中的内容如下图：



3、假设浮点功能部件的延迟为：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。

（1） 自己编写的程序如下：

L.D F6, 24(R2)

L.D F2, 12(R3)

MUL.D F0, F2, F4

ADD.D F8, F2, F6

DIV.D F10, F0, F6

SUB.D F6, F8, F2

（2）重复上述步骤2中的工作：

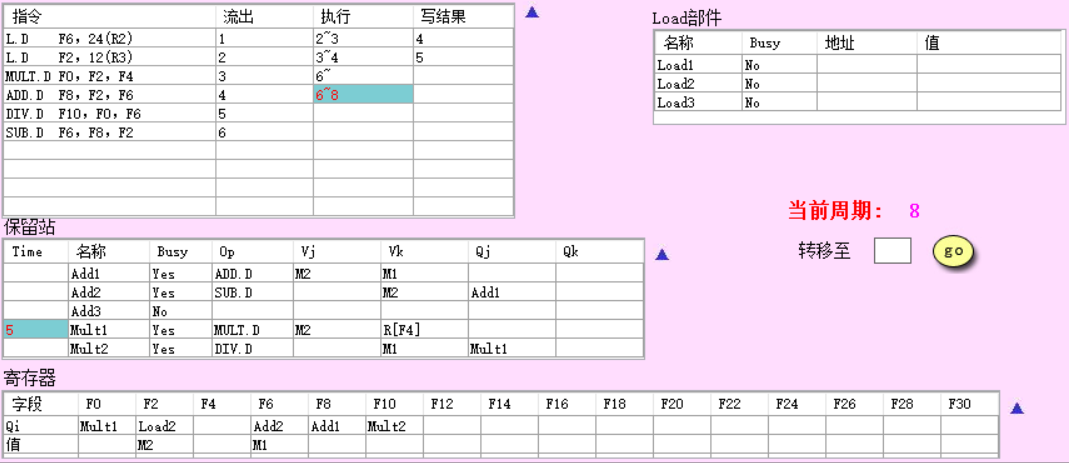
先在Tomasulo算法模拟器中重新输入指令，然后根据要求修改各个浮点功能部件的延迟时间，随后开始逐步执行程序。

① 给出在第3个时钟周期时保留站的内容。



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。

在①的基础上再步进5个周期，此时为第8个时钟周期，保留站、Load缓冲器以及寄存器状态表中的内容如下图：



③ 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。

在②的基础上再步进10个周期，此时为第18个时钟周期，保留站、Load缓冲器以及寄存器状态表中的内容如下图：



1. **实验中出现的问题和解决方法**

·实验中关于Tomasulo算法模拟器的使用

**问题：**该实验中运用Tomasulo算法模拟器，主要是为了更清楚地学习掌握Tomasulo算法的执行流程，体会该算法在动态调度中的优越性；而如果在输入代码段后直接采用“执行到底”按钮，则只能得到执行结果，无法观察到程序执行的细节部分，不利于我们对算法流程的掌握。

**解决方案**：先尽量多使用Tomasulo算法模拟器上的“步进”按钮，进行步进形式执行，仔细观察每一个时钟周期前后各信息表中的内容变化情况，了解Tomasulo算法中各程序执行的详细过程；有需要的情况下再灵活使用“退1步”“前进5个周期”“后退5个周期”等按钮，辅助观察程序执行情况。