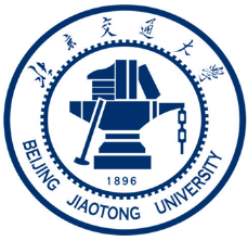


北京交通大學

BEIJING JIAOTONG UNIVERSITY

《计算机体系结构》

实验报告

实验名称:	实验 4_ Tomasulo 算法
学　号:	
姓　名:	
学　院:	计算机与信息技术学院
日　期:	2022 年 11 月 08 日

目录

1. 了解 Tomasulo 算法的模拟软件

运行 Tomasulo 算法的模拟软件，了解其使用说明。

通过简单的学习，了解了 Tomasulo 算法模拟软件的一般用法。

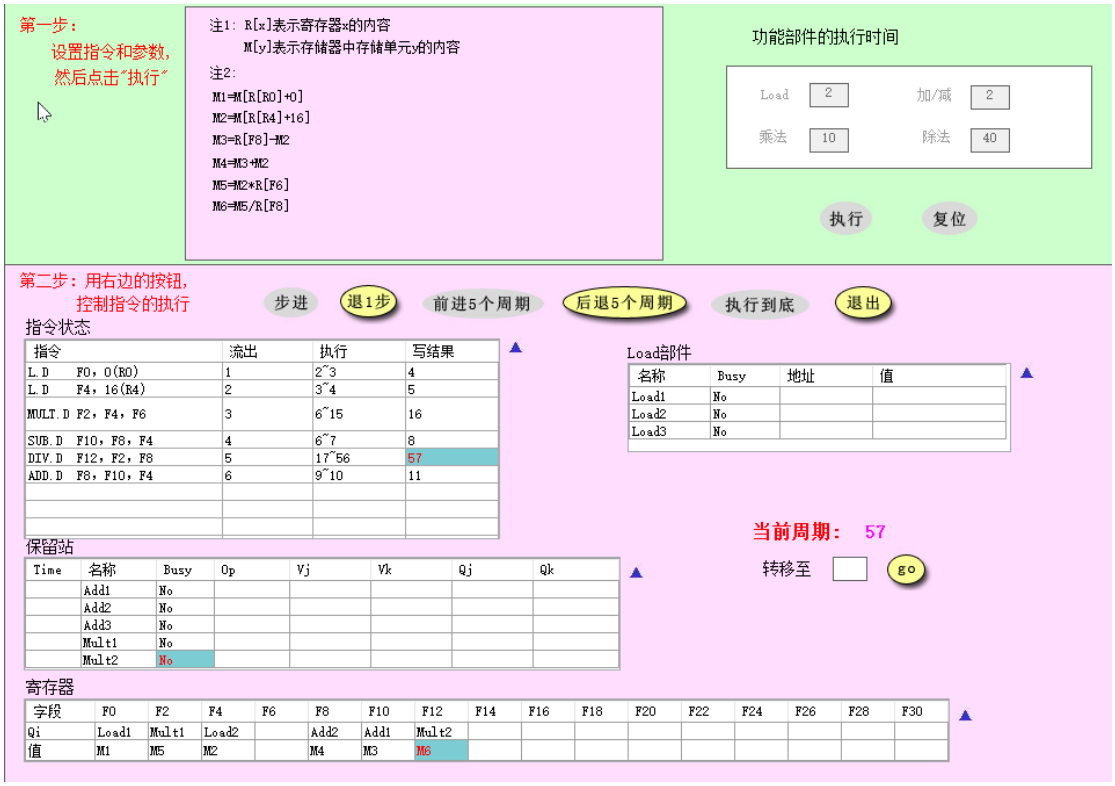


图 1-1 模拟软件学习剪影

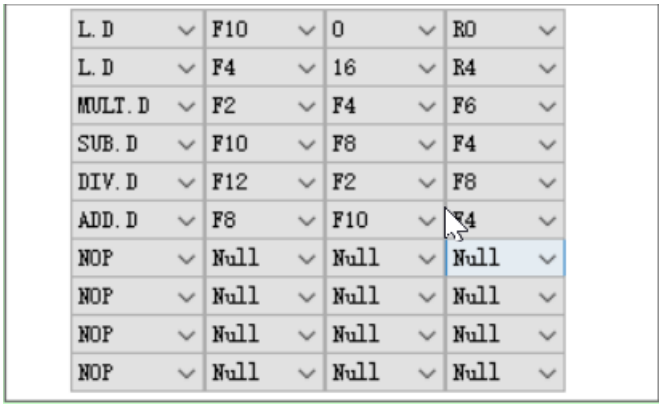
2. 掌握 Tomasulo 算法的运行过程及其原理

可任意设置一系列指令（当然，按软件自动给出的 6 条指令也可以，但提倡自己设置一些比较复杂的指令），然后，按步进方式运行。每运行一步，均对保留站的状态变化进行解释（如果状态无变化，即只有时钟改变时，可不予解释），直至运行结束。解释的内容包括：

* 指明哪条指令从一种状态变到另一种状态。状态包括：流出(我们称为发射)、执行、写结果。

* 对于指令的状态变化，保留站（当然也可以包括寄存器、load 部件）发生了哪些变化。

简单修改指令，进行实验。



L.D	F10	0	R0
L.D	F4	16	R4
MULT.D	F2	F4	F6
SUB.D	F10	F8	F4
DIV.D	F12	F2	F8
ADD.D	F8	F10	F4
NOP	Null	Null	Null
NOP	Null	Null	Null
NOP	Null	Null	Null
NOP	Null	Null	Null

图 2-1 指令的修改

Cycle0:
初始状态。开始执行。

Cycle1:

图 2-2 Cycle1

Load1 发射，进入 Load1 缓冲器，寄存器中 F10 变为 Load1，等待数值到来。

Cycle2:

图 2-3 cycle2

Load2 发射，占用 load2 缓冲器，地址为 16+R4，寄存器记录 F4 为 Load2.

Cycle3：

图 2-4 cycle3

Load1 执行完成。乘法指令进入保留栈 Mult1，需要读 F4 和 F6，其中 F4 状态表为 Load2，则没有就绪，那么，我们需要等待，所以 Qj 为 Load2；F6 为空，就是立即数，Vk 为 R（F6），结果写入 F2，状态表改写。

Cycle4：

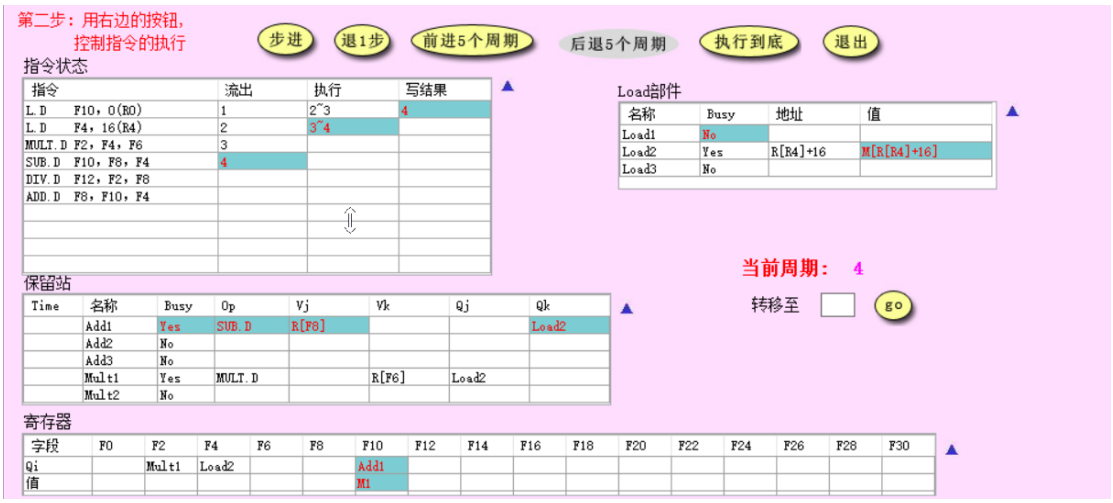


图 2-5 Cycle4

首先 load1 开始写结果，退出 Load1 缓冲器， F10 的值变成 M1=M[R[R0]+0]。Load2 执行结束。Sub 指令发射，占用加法保留栈 Add1， 寄存器状态表中 F8 为立即数，F4 为 load2；所以 Add1 中 Vj 为 R[F8]，而 Qk 为 Load2。Sub 指令写入 F10，所以 F10 记录为 Add1。

Cycle5：

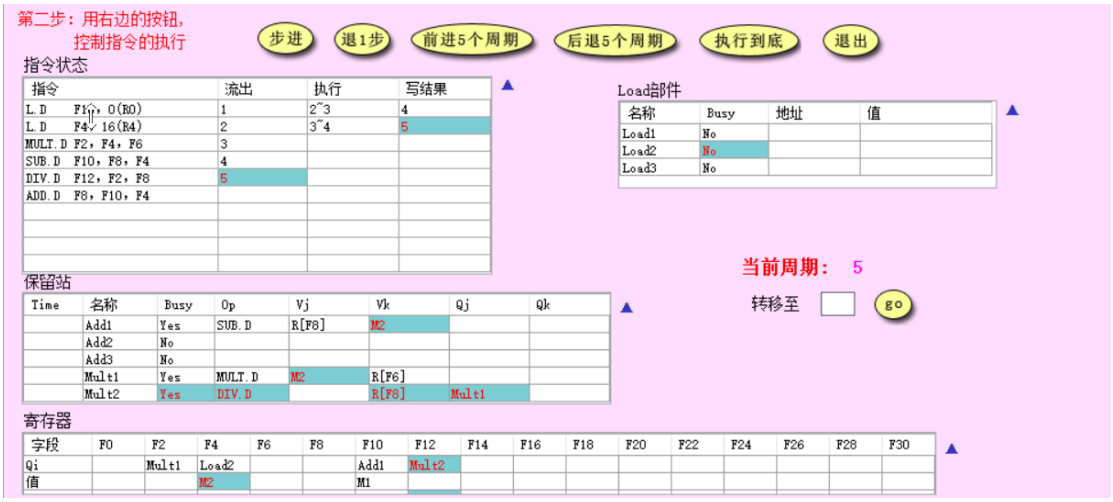


图 2-6 cycle5

Load2 写回，所以 F4 的值变成 M2=[R[R4] + 16]，退出 Load2 缓冲器。Load2 结束，所以 Add1 的 Qk 得到了数，转移到 Vk 中的 M2，同理还有 Mult1 也转移到 Vj。

Div 发射，占用 Mult2，F8 为立即数，写入 Vk，F2 等待 Mult1 的值，F12 状态表改写为 Mult2.

Cycle6：

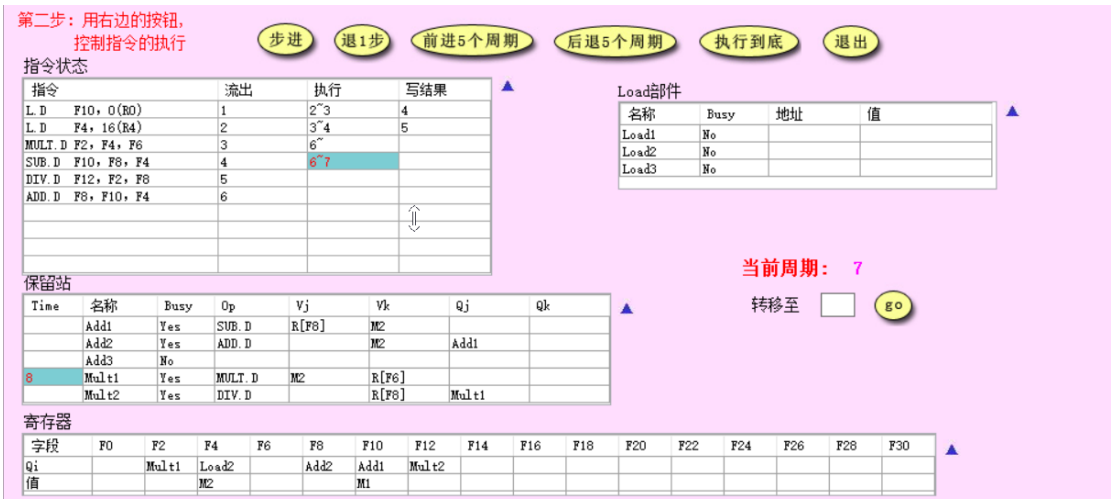


图 2-7 cycle6

Sub 和 Mult 指令在上个周期拿到了所有操作数，开始执行。Add 指令发射，占用 Add2 保留站，F10 等待 Add1 结果，F4 为立即数 M2，F8 状态表改写为 Add2。

Cycle7：

Sub 指令执行完成。

Cycle8：

Sub7 指令写回，让出 Add1 保留站。F10 得到 M3。Add2 得到操作数，M3。

Cycle9：

Add 指令在上个周期得到了所有操作数，这周期开始执行。

Cycle10：

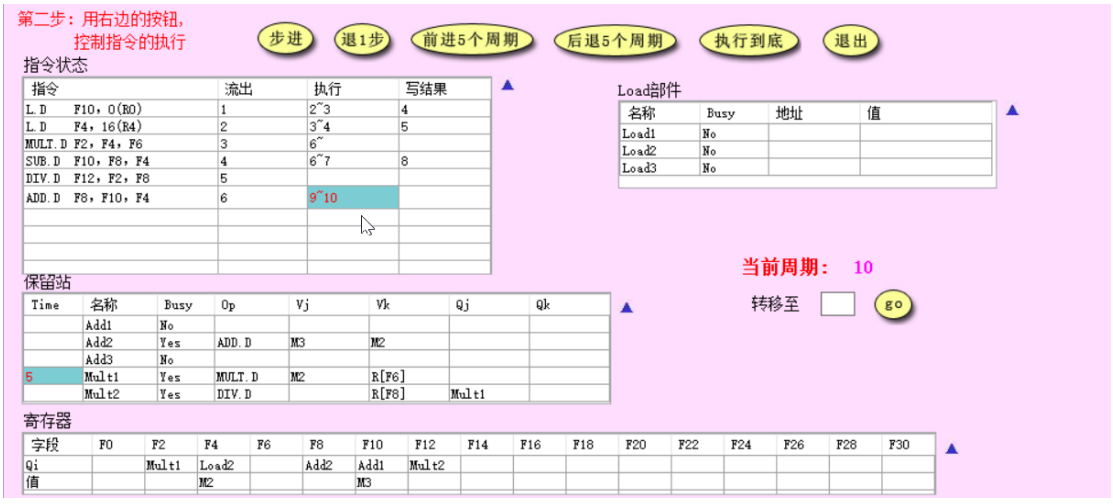


图 2-8 cycle10

加法指令执行完毕。乘法指令继续执行。

Cycle11：

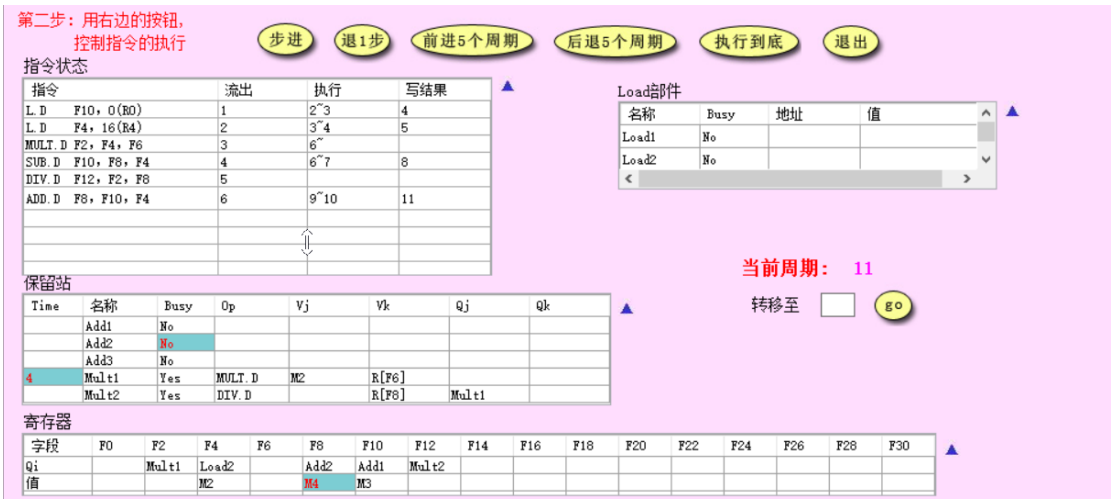


图 2-9 cycle11

Add 指令写回，F8 状态表变为 M4。乘法指令继续执行。

Cycle12~15：
乘法继续执行。Cycle15，乘法执行结束。

Cycle16：

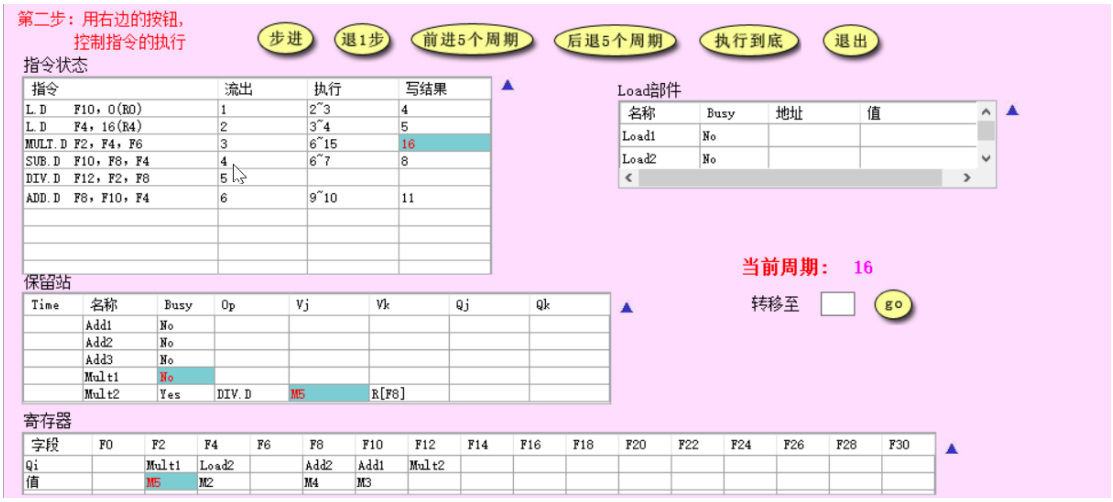


图 2-10 cycle16

乘法指令写回，退出 Mult1 保留站占用，F2 状态表变为 M5 即乘法结果。Mult2Vj 变为 M5。

Cycle17：
除法上一周期拿到所有操作数，开始执行。

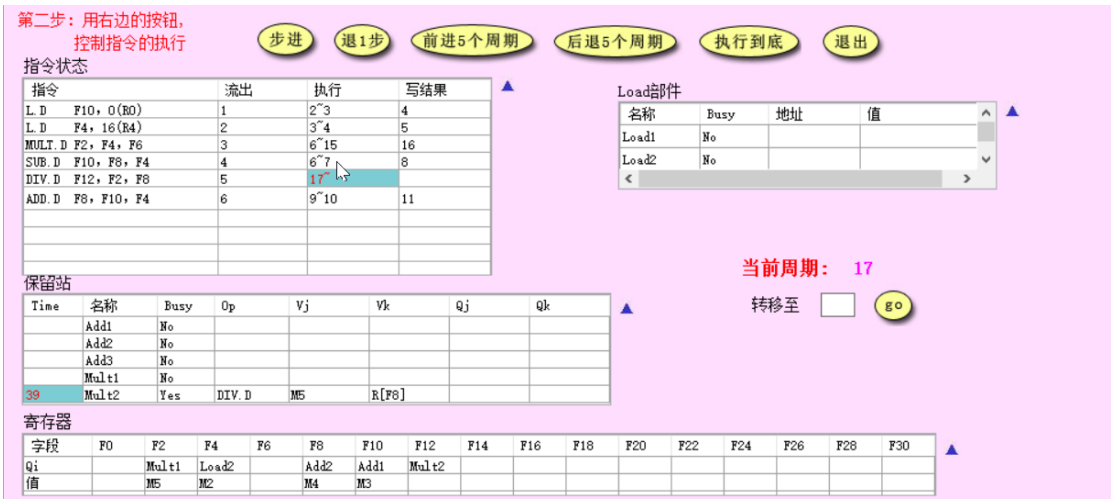


图 2-11 cycle17

Cycle18~56:
除法继续执行。

Cycle57:

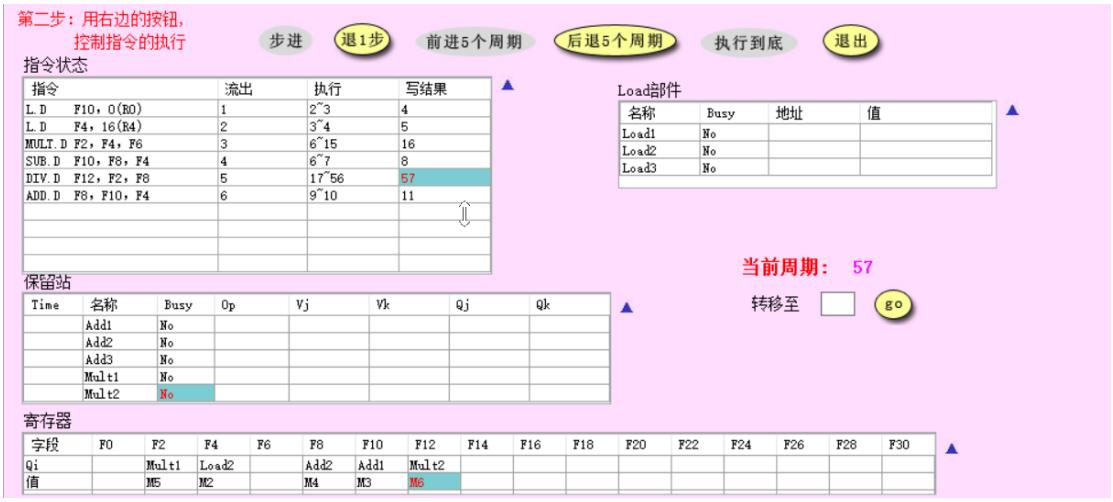


图 2-12 cycle57

除法结果写回。F12 得到 M6 即除法运算结果。退出 Mult2 保留站。

3. 分别举例解释一下为什么 Tomasulo 算法消除了这些指令中的 WAR 和 WAW 冒险

回答：

WAR 的消除。

如 Div F12, F2， F8

Add F8, F10， F4

除法指令要读 F8 但是，Add 写 F8，这时候会出现 WAR。但是 Tomasulo 算法中，Div 直接冲寄存器表中读 F8，这个时候 Add 还没有写入，Div 也不能执行，但是 Div 已经把 F8 读入保留站了。这样就用硬件避免了冲突。

WAW 的消除:

本次程序运行的例子中，不采用 tomasulo 算法也不会有 WAW 风险。

我们可以举一个这样的例子

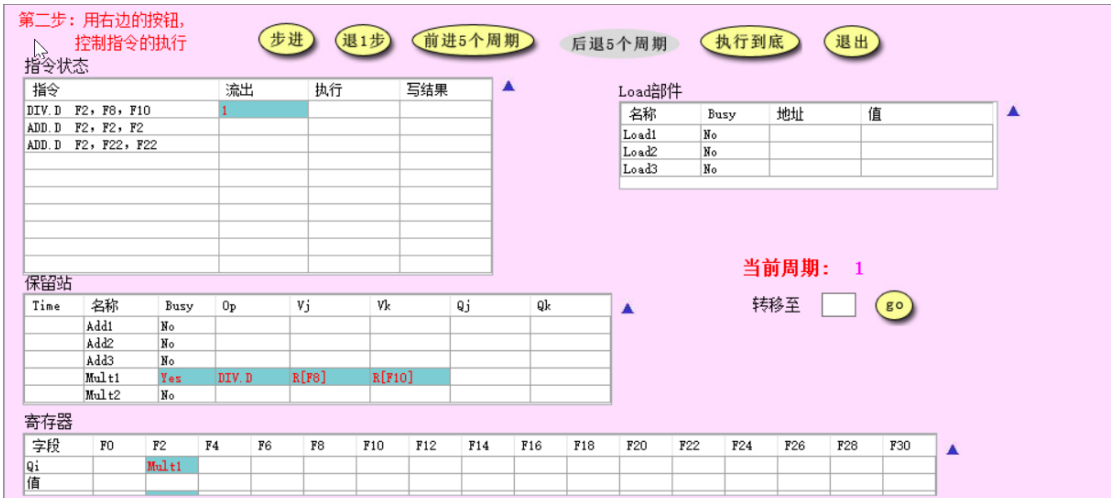
这里主要是 Div 和后边的 add 都会写入 F2，那么第二条指令的 F2 会产生 WAW。

程序的原本意思，当然是第二个 ADD 读 Div 的结果，但是 Div 周期更长，第三个 ADD 的 F2 更早写入，所以会造成第二个 ADD 指令读数可能出问题。

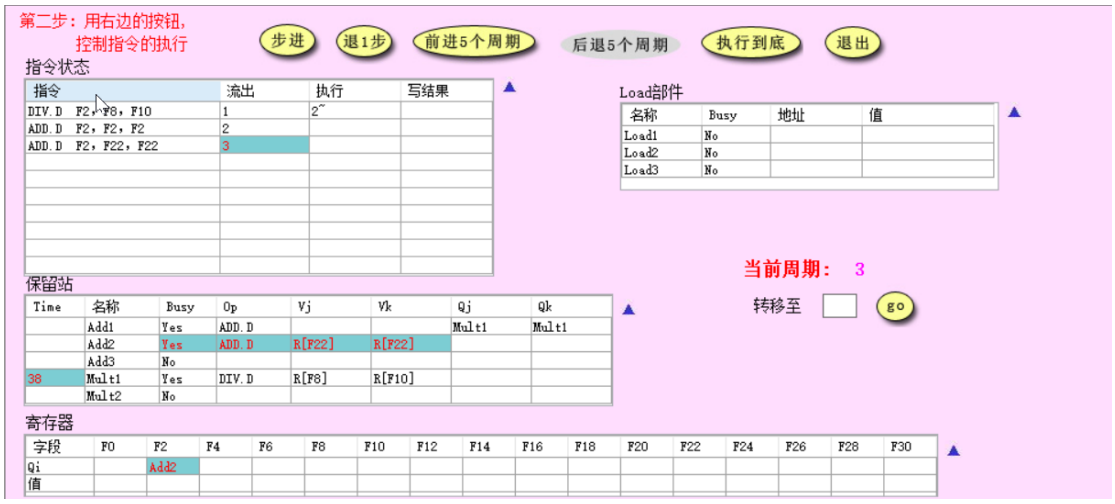
Tomasulo 算法中，对保留站重命名。指令顺序发射，这样第二个 ADD 的操作数只会等待 Div 算出的 F2 的值，即使第三个 ADD 的结果先广播过来，它也不会接受。

第 42 周期，第二个 ADD 才得到正确的操作数。

具体运行图见下:




第二步：用右边的按钮，
控制指令的执行
步进
退1步
前进5个周期
后退5个周期
执行到底
退出
指令状态
指令 流出 执行 写结果
DIV.D F2, F8, F10 1 2~
ADD.D F2, F2, F2 2
ADD.D F2, F22, F22
Load部件
名称 Busy 地址 值
Load1 No
Load2 No
Load3 No
当前周期： 2
转移至 go
保留站
Time 名称 Busy Op Vj Vk Qj Qk
Add1 Yes ADD.D Mult1 Mult1
Add2 No
Add3 No
39 Mult1 Yes DIV.D R[F8] R[F10]
Mult2 No
寄存器
字段 F0 F2 F4 F6 F8 F10 F12 F14 F16 F18 F20 F22 F24 F26 F28 F30
Qi Add1
值


第二步：用右边的按钮，
控制指令的执行
步进
退1步
前进5个周期
后退5个周期
执行到底
退出
指令状态
指令 流出 执行 写结果
DIV.D F2, F8, F10 1 2~
ADD.D F2, F2, F2 2
ADD.D F2, F22, F22 3
Load部件
名称 Busy 地址 值
Load1 No
Load2 No
Load3 No
当前周期： 3
转移至 go
保留站
Time 名称 Busy Op Vj Vk Qj Qk
Add1 Yes ADD.D Mult1 Mult1
Add2 Yes ADD.D R[F22] R[F22]
Add3 No
38 Mult1 Yes DIV.D R[F8] R[F10]
Mult2 No
寄存器
字段 F0 F2 F4 F6 F8 F10 F12 F14 F16 F18 F20 F22 F24 F26 F28 F30
Qi Add2
值


第二步：用右边的按钮，
控制指令的执行
步进
退1步
前进5个周期
后退5个周期
执行到底
退出
指令状态
指令 流出 执行 写结果
DIV.D F2, F8, F10 1 2~
ADD.D F2, F2, F2 2
ADD.D F2, F22, F22 3 4~
Load部件
名称 Busy 地址 值
Load1 No
Load2 No
Load3 No
当前周期： 4
转移至 go
保留站
Time 名称 Busy Op Vj Vk Qj Qk
Add1 Yes ADD.D Mult1 Mult1
1 Add2 Yes ADD.D R[F22] R[F22]
Add3 No
37 Mult1 Yes DIV.D R[F8] R[F10]
Mult2 No
寄存器
字段 F0 F2 F4 F6 F8 F10 F12 F14 F16 F18 F20 F22 F24 F26 F28 F30
Qi Add2
值

第二步：用右边的按钮，
控制指令的执行

步进 | 退1步 | 前进5个周期 | 后退5个周期 | 执行到底 | 退出

指令状态

指令	流出	执行	写结果
DIV.D F2, F8, F10	1	2~	
ADD.D F2, F2, F2	2		
ADD.D F2, F22, F22	3	4~5	

Load部件

名称	Busy	地址	值
Load1	No		
Load2	No		
Load3	No		

当前周期： 5

转移至 go

保留站

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
	Add1	Yes	ADD.D			Mult1	Mult1
	Add2	Yes	ADD.D	R[F22]	R[F22]		
	Add3	No					
36	Mult1	Yes	DIV.D	R[F8]	R[F10]		
	Mult2	No					

寄存器

字段	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qi		Add2														
值																

第二步：用右边的按钮，
控制指令的执行

步进 | 退1步 | 前进5个周期 | 后退5个周期 | 执行到底 | 退出

指令状态

指令	流出	执行	写结果
DIV.D F2, F8, F10	1	2~	
ADD.D F2, F2, F2	2		
ADD.D F2, F22, F22	3	4~5	6

Load部件

名称	Busy	地址	值
Load1	No		
Load2	No		
Load3	No		

当前周期： 6

转移至 go

保留站

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
	Add1	Yes	ADD.D			Mult1	Mult1
	Add2	No					
	Add3	No					
35	Mult1	Yes	DIV.D	R[F8]	R[F10]		
	Mult2	No					

寄存器

字段	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qi		Add2														
值		M1														

第二步：用右边的按钮，
控制指令的执行

步进 | 退1步 | 前进5个周期 | 后退5个周期 | 执行到底 | 退出

指令状态

指令	流出	执行	写结果
DIV.D F2, F8, F10	1	2~	
ADD.D F2, F2, F2	2		
ADD.D F2, F22, F22	3	4~5	6

Load部件

名称	Busy	地址	值
Load1	No		
Load2	No		
Load3	No		

当前周期： 7

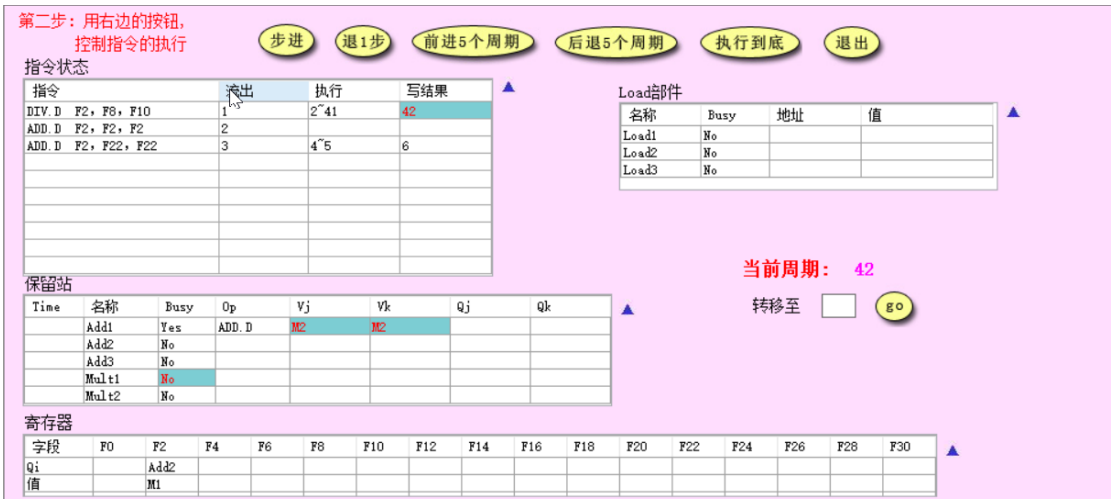
转移至 go

保留站

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
	Add1	Yes	ADD.D			Mult1	Mult1
	Add2	No					
	Add3	No					
34	Mult1	Yes	DIV.D	R[F8]	R[F10]		
	Mult2	No					

寄存器

字段	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qi		Add2														
值		M1														


第二步：用右边的按钮，
控制指令的执行
步进
退1步
前进5个周期
后退5个周期
执行到底
退出
指令状态
指令 流出 执行 写结果
DIV.D F2，F8，F10 1 2~41 42
ADD.D F2，F2，F2 2
ADD.D F2，F22，F22 3 4~5 6
Load部件
名称 Busy 地址 值
Load1 No
Load2 No
Load3 No
当前周期： 42
转移至
go
保留站
Time 名称 Busy Op Vj Vk Qj Qk
Add1 Yes ADD.D M2 M2
Add2 No
Add3 No
Mult1 No
Mult2 No
寄存器
字段 F0 F2 F4 F6 F8 F10 F12 F14 F16 F18 F20 F22 F24 F26 F28 F30
Qi Add2
值 M1