**指令调度和Tomasulo算法**

实验目的

1. 加深对指令调度技术的理解。

2. 加深对Tomasulo算法的理解。

3. 熟练掌握用指令调度技术来解决流水线中的数据冲突的方法，进一步理解指令调度技术对CPU性能的改进。

4. 掌握采用了Tomasulo算法的浮点处理部件的结构、保留站的结构，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。

实验内容和步骤

首先要掌握MIPSsim模拟器的使用方法。见文档《MIPSsim使用手册》。

1. 启动MIPSsim。

2. 根据流水线相关知识，进一步理解流水线窗口中各段的功能，掌握各流水寄存器的含义（双击各段，就可以看到各流水寄存器的内容）。

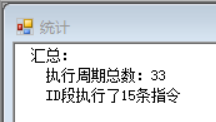
3. 选择“配置”→“流水方式”选项，使模拟器工作于流水方式下。

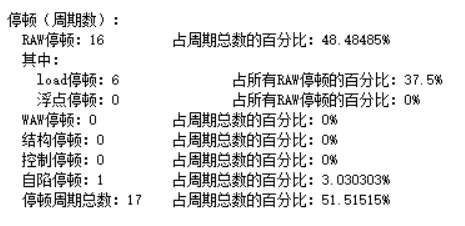
**4. 用指令调度技术解决流水线中的结构冲突与数据冲突。**

（1）启动MIPSsim。

（2）用MIPSsim的“文件”→“载入程序”选项来加载schedule.asm（在模拟器所在文件夹下的“样例程序”文件夹中）。

（3）关闭定向功能。这是通过在“配置”菜单中关闭“定向”（使该项前面没有“√”号）来实现的。

（4）执行所载入的程序。通过查看统计数据和时钟周期图，找出并记录程序执行过程中各种冲突发生的次数、发生冲突的指令组合，以及程序执行的总时钟周期数33。

各种冲突发生次数：

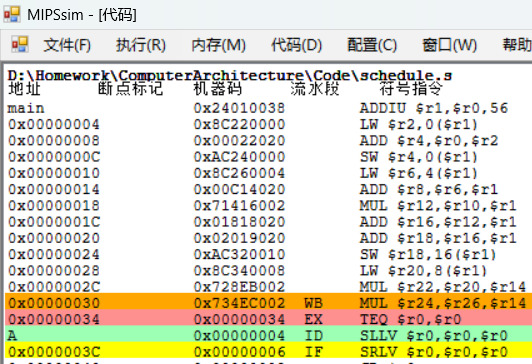
RAW停顿：16，占比：48.48485%

load停顿：6，占RAW比37.5%

自陷停顿：1，占比3.030303%

停顿周期总数：17，占比51.51515%

发生冲突指令：

LW $r2 , 0($r1) 和 ADD $r4, $r0, $r2

ADD $r4, $r0, $r2和 SW $r4 , 0 ($r1)

SW $r4 , 0 ($r1) 和 LW $r6 , 4 ($r1)

ADD $r8, $r6, $r1 和 MUL $r12, $r10, $r1

ADD $r16 , $r12, $r1 和 ADD $r18, $r16, $r1

ADD $r18, $r16, $r1和 SW $r18, 16($r1)

SW $r18, 16($r1) 和 LW $r20 , 8 ($r1)

MUL $r22, $r20, $r14 和 MUL $r24, $r26, $r14

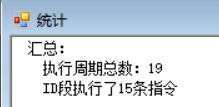
（5）采用指令调度技术对程序进行指令调度，消除冲突。将调度后的程序保持到 after-schedule.asm中：

after-schedule:

文本

描述已自动生成

（6）载入after-schedule.asm。

（7）执行该程序，观察程序在流水线中的执行情况，记录程序执行的总时钟周期数19；

（8）根据记录结果，比较调度前和调度后的性能。论述指令调度对于提高CPU性能的作用：

指令调度后时钟总周期数从33降低到19，指令调度使指令重新组合，可以消除部分的数据冲突，从而通过指令调度技术显著提高了CPU的使用率，大大减少了指令冲突的次数，提高了CPU的性能。

**5. 使用Tomasulo实现指令动态调度。**

首先要掌握Tomasulo模拟器的使用方法。

（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。执行下面的代码段：

L.DF6,24（R2）

L.DF2,12（R3）

MUL.DF0,F2,F4

SUB.DF8,F6,F2

DIV.DF10,F0,F6

ADD.DF6,F8,F2

（2）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：（MUL.D指令执行完成写结果M5到寄存器F0, 保留站中原来存放第三条指令的位置清空）

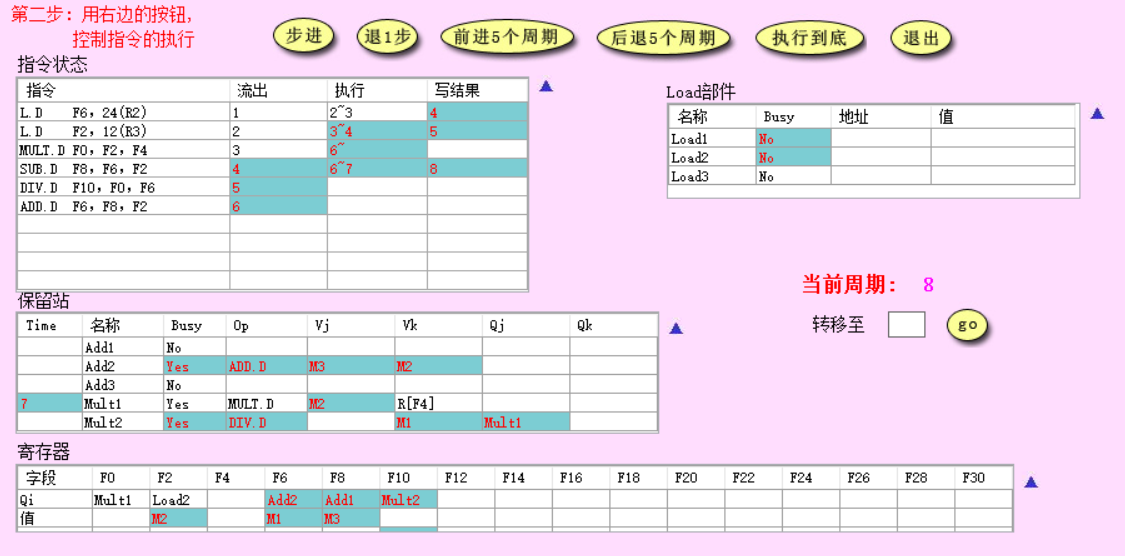
日历

描述已自动生成

（3）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。

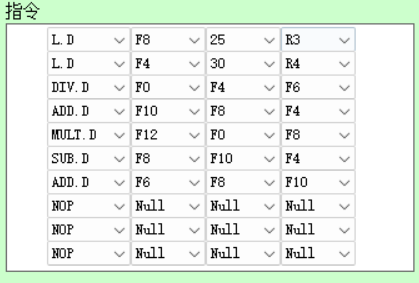
（4）对于与上面相同的延迟时间和代码段。给出在第3个时钟周期时，保留站的内容。

（5）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：

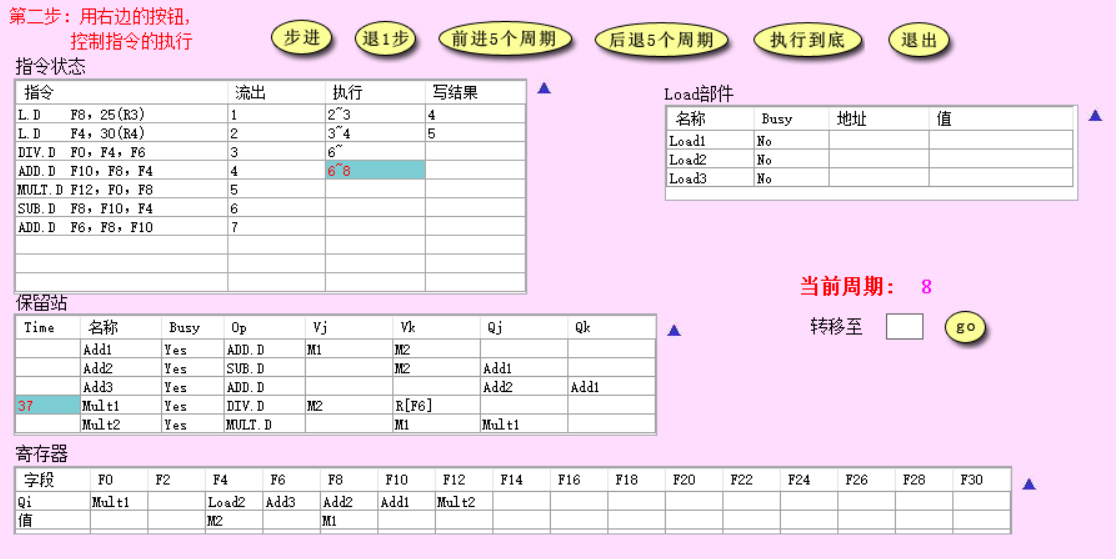


（6）假设浮点功能部件的延迟时间为加减法3个时钟周期，乘法8个时钟周期，除法40个时钟周期。自己编写一段程序（要在实验报告中给出），重复上述步骤（5）的工作。

编写指令：



第八个周期结果：



执行至此时，该阶段前两个指令执行完成，第3、4指令处于执行状态。第5、6、7指令进入待执行阶段。该阶段还有Add1、Add3、Mult1、Mult2这些指令流出但是未完成。Load和Add2指令已经执行完成、并把结果送到对应的浮点寄存器中。该阶段所有load缓冲器都是处于闲置状态。该阶段Add1中的加法指令由于在第5个时钟周期已经取到了两个操作数，所以加法浮点器执行了3个周期，处于执行阶段的最后一个时钟周期；Add2中的减法指令需要Add1中的结果作为第一操作数，所以为了避免写读冲突，需要等待Add1中的操作执行结束并写出结果；Add3中的加法指令需要Add1和Add2中的结果作为两个操作数，所以为了避免写读冲突，需要等待Add1和Add2中的操作执行结束并写出结果；Mult1中的除法指令处于功能部件的执行阶段，除法指令已经执行了3个时钟周期；Mult2中的乘法指令需要Mult1中的结果作为第一操作数，所以为了避免写读冲突，需要等待Mult1中的操作执行结束并写出结果。