

《 计算机体系结构 》

实验报告

|  |  |
| --- | --- |
| 班 级： | **计211** |
| 学 号： | **21013017** |
| 姓 名： | **蒋文强** |
| 指导教师： | **梁建宁** |

信息科学与工程学院

2023年 11 月

**实验名称** **Tomasulo算法** **实验地点** 信息楼418**实验日期** 11.29

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。

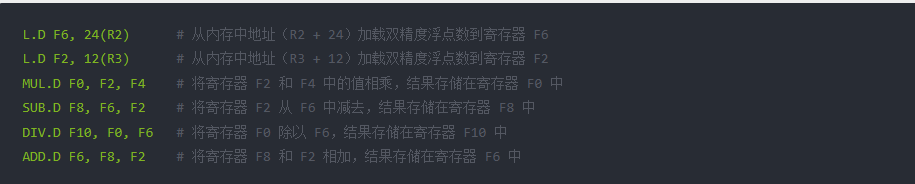
1. **实验设备**

实验平台采用Tomasulo算法模拟器。

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。

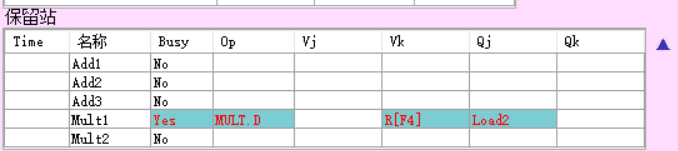
① 对于下面的代码段，给出当指令MUL.D即将写回时，保留站、Load缓冲器以及寄存器状态表中的内容。



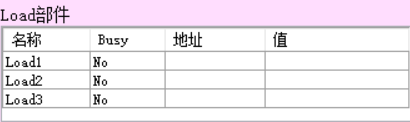
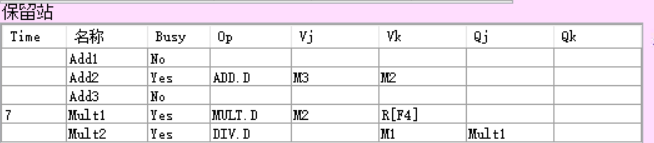
②按步进方式执行上述代码，利用模拟器的“小三角按钮”的对比显示功能，观察每一个时钟周期前后各信息表中内容的变化情况。

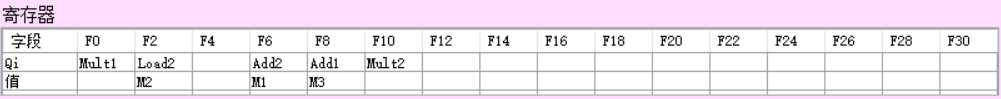
（2）对于与上面相同的延迟时间和代码段：

①给出在第3个时钟周期时保留站的内容。

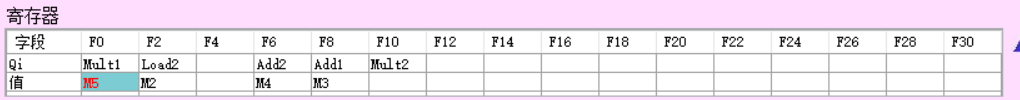
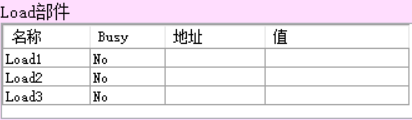


②步进5个时钟周期，给出这时保留站、Load缓冲器以及寄存器状态表中的内容。



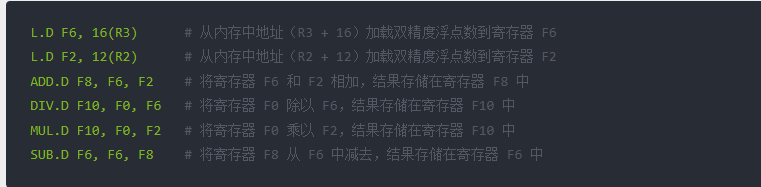


③再步进10个时钟周期，给出这时保留站、Load缓冲器以及寄存器状态表中的内容。

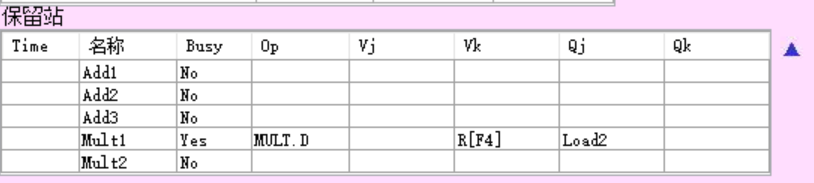


（3）假设浮点功能部件的延迟为：加减法3个时钟周期，乘法8个时钟周期，除法40个时钟周期。自己编写一段程序（要在实验报告中给出），重复上述步骤（2）的工作。

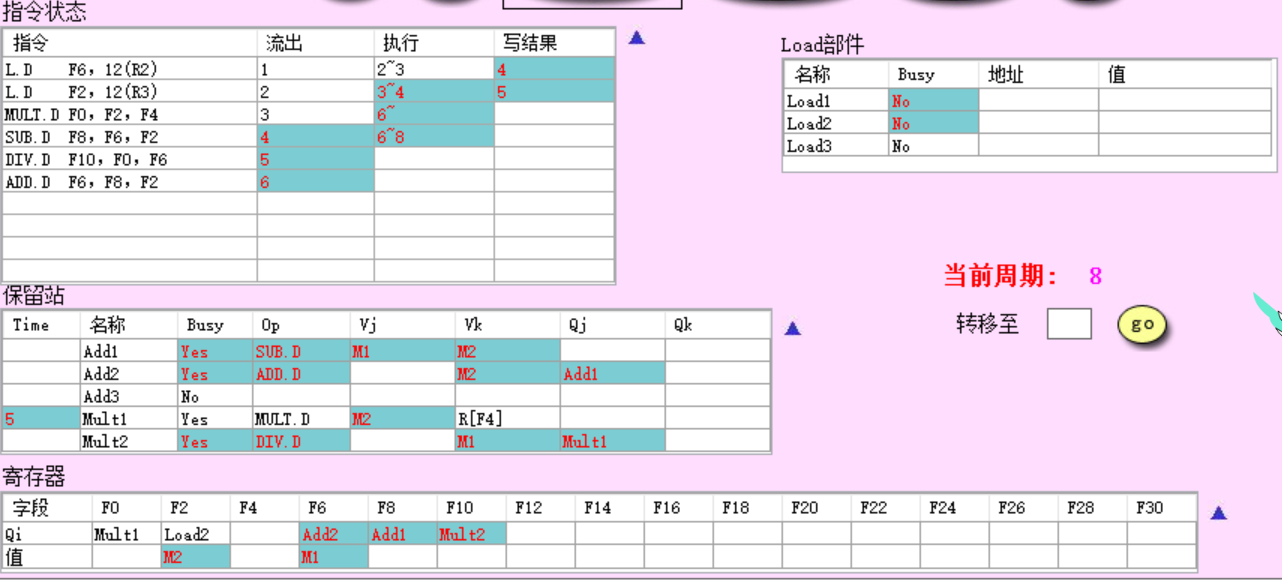
**实验程序：**



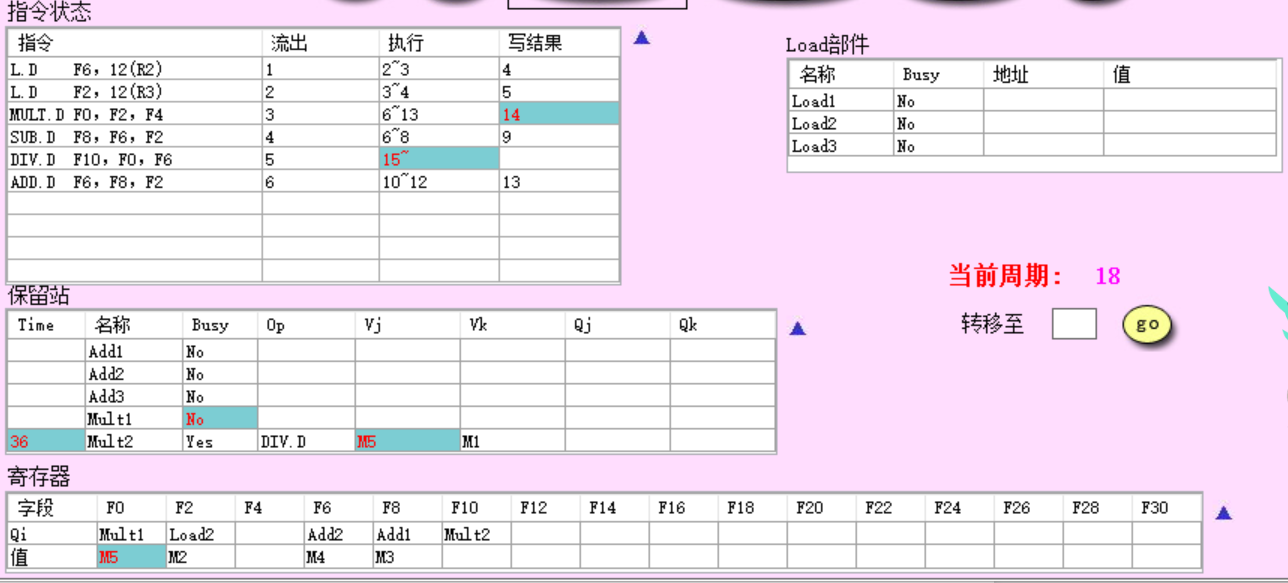
1. 给出在第3个时钟周期时保留站的内容。



1. 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



1. 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。



1. **实验总结**

我了解到了Tomasulo算法是一种动态调度指令执行的高级硬件架构，旨在提高指令级并行性（ILP），下面是我的一写总结。

1. 寄存器重命名：

寄存器重命名是Tomasulo算法的核心特性之一。通过可以为每个指令生成一个独立的寄存器标签，而不是使用物理寄存器名，这样算法消除了数据相关性的假依赖。这种重命名使得指令可以并发执行，而无需等待先前指令的结果。

例如，如果有两条指令 A 和 B，B 需要 A 的结果，传统情况下会就存在数据相关性，这导致 了B 必须等待 A 执行完成。然而，在Tomasulo算法中，B 可以使用重命名后的标签，而不必等待 A 完成，从而提高了并发度。

2. 预留站：

Tomasulo算法使用预留站（Reservation Stations）作为执行单元。每个执行单元都有一个预留站，用于存储指令的操作数和运算结果。这允许多个指令同时在不同的执行单元中执行，而不会互相阻塞。

3. 公共数据总线：

公共数据总线（Common Data Bus，CDB）用于广播计算出的结果。当一个指令完成计算时，它会将结果广播到CDB上，其他指令通过CDB获取需要的数据。这种广播机制消除了数据的点对点传递，促进了更灵活的数据传输。

4. 异步乱序执行：

Tomasulo算法采用异步乱序执行的方式，即指令的发射和完成顺序不一致。只要操作数就绪，指令就可以发射，并在执行单元中独立执行。这种灵活性允许处理器更有效地利用可用资源，提高整体性能。

5. 强调ILP和流水线：

Tomasulo算法的设计旨在最大程度地发挥指令级并行性。通过强调ILP和流水线执行，它有效地克服了传统处理器中存在的数据相关性带来的性能瓶颈。

6. 推测执行：

Tomasulo算法的灵活性还使其适用于推测执行。通过在分支指令上进行推测性执行，可以在等待分支决定的同时执行其他指令。如果推测错误，算法可以通过回滚机制来纠正，以最小化对性能的影响。

所以，Tomasulo算法的创新性在于其对数据相关性的处理方式，通过寄存器重命名、预留站和公共数据总线的结合使用，以及乱序执行的思想，为处理器提供了更高的并行度和更灵活的执行模型，这为高性能计算提供了强大的支持。