

计算机组成原理实验三实验报告

**学 院： 计 算 机 工 程 学 院**

**班 级： 计算2114**

**姓 名: 庄佳强 章立早**

**学 号： 202121331104 202121331118**

二〇二二 年 11 月10 日

# 实验一.A

# 一 题目要求

* 1. 介绍  
      在前两个实验当中，你已经能够根据功能需求实现出一个能够解决问题的软件/硬件系统，我们接下来的主要任务是学习如何提升计算系统的性能。为了提升计算系统的性能，我们首先将从CPU执行时间的经典模型开始讨论。经典模型告诉我们，在不修改程序指令数和CPI的前提下，我们可以通过提升主频达到提升系统性能的目的。因此，在这个实验中，我们可以通过应用流水线的技巧提升主频。你将使用自己编写的向量点积测试程序作为负载，比较单周期处理器和流水线处理器之间的性能差异。
  2. 设计你的流水线MIPS处理器  
      你可以拿出你在实验2当中设计的那个能够运行向量点积测试程序的单周期处理器，并再次确保它能够运行出正确的结果。在优化之前，先想办法记录下你设计的单周期处理器运行测试程序所需要的时钟周期数。（你可以在程序末尾添加一条特别的指令，再利用一个计数器进行计数，并且当程序运行到末尾这条特殊指令的时候终止计数）。  
      流水线处理器的设计有很多不同的技巧，我们在课堂上讨论了其中的几种：流水级的划分、数据旁路、分支跳转预测等等。你可以选择只使用其中一种策略，也可以将几种不同的策略搭配起来使用。请注意，无论你如何搭配，你都应该能够得到与之前完全一致的计算结果。
  3. 评估你的流水线处理器的性能  
      当你设计好你的流水线处理器之后，再次运行程序，用同样的计时方法记录下你设计的流水线处理器运行测试程序所需要的时钟周期数。当然，这个新的时钟周期数并不能反映程序的响应时间，还需要考虑主频的提升。为了简化讨论，我们在这个实验当中规定：N级流水线能够将主频提升N倍。比如，你只是在取指令和译码之间划分了流水级，那么这个划分就可以认为将主频提升了两倍。  
      我们将使用以下公式计算性能提升的倍数：  
       
      （程序P在单周期处理器上执行的时钟周期数）/（程序P在流水线处理器上执行的时钟周期数）×流水线的级数

# 二 设计思路

#### 改造部分：

在实验二的基础上，添加三级流水线。

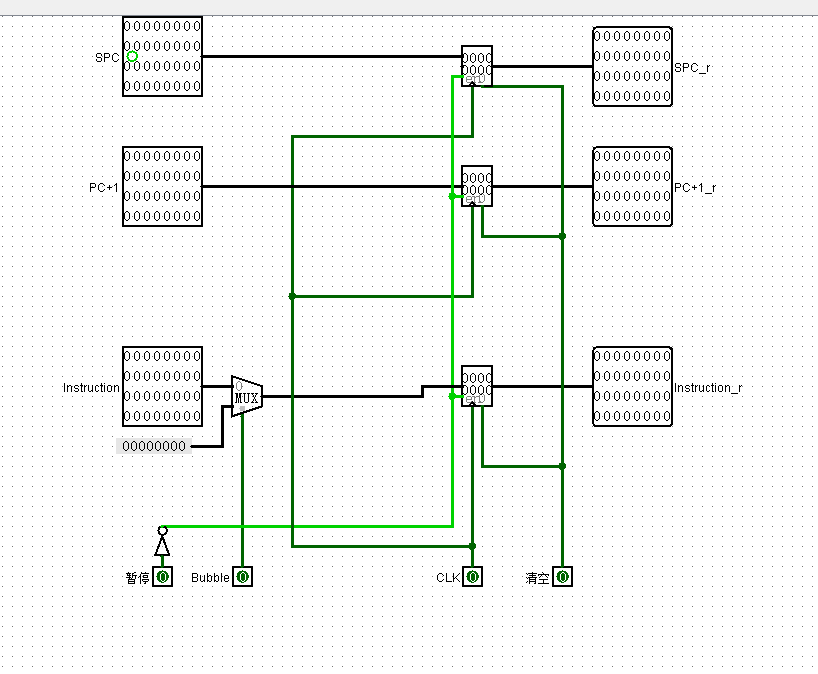
在译码阶段可以添加IF/ID，实现取指令和译码指令阶段隔开一个周期，在取数据阶段添加数据旁路和数据冒险，通过重定向加快数据的读取。加快程序性能。

在ALU之前加入ID/EX,用于存储指令的数据状态，使信号指令延后一个周期。来保证译码和处理指令阶段可以分开。提高程序性能。

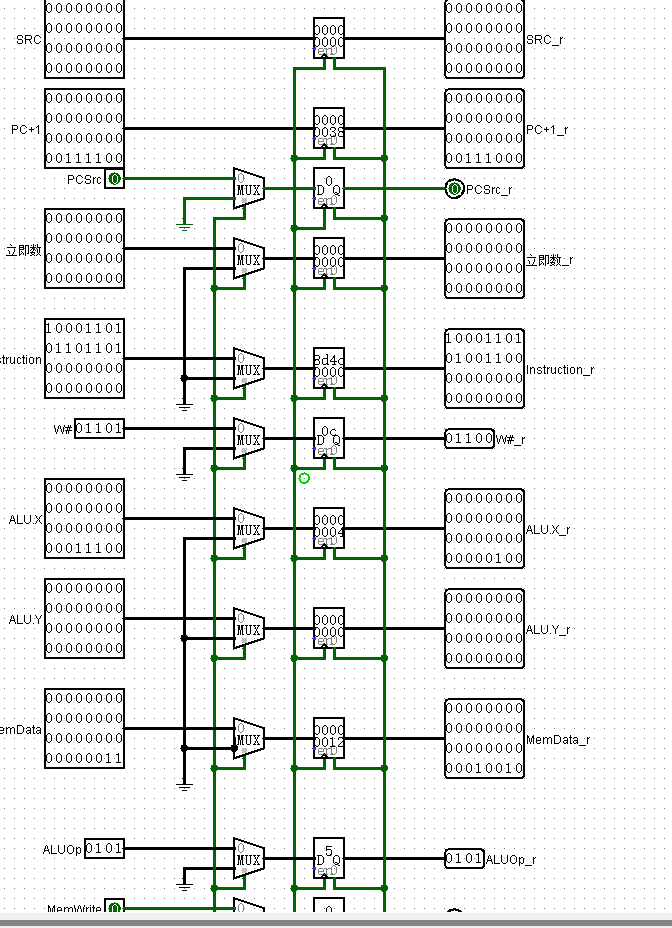
而IF/ID和ID/EX来自课堂中的编码包，修改后使用。

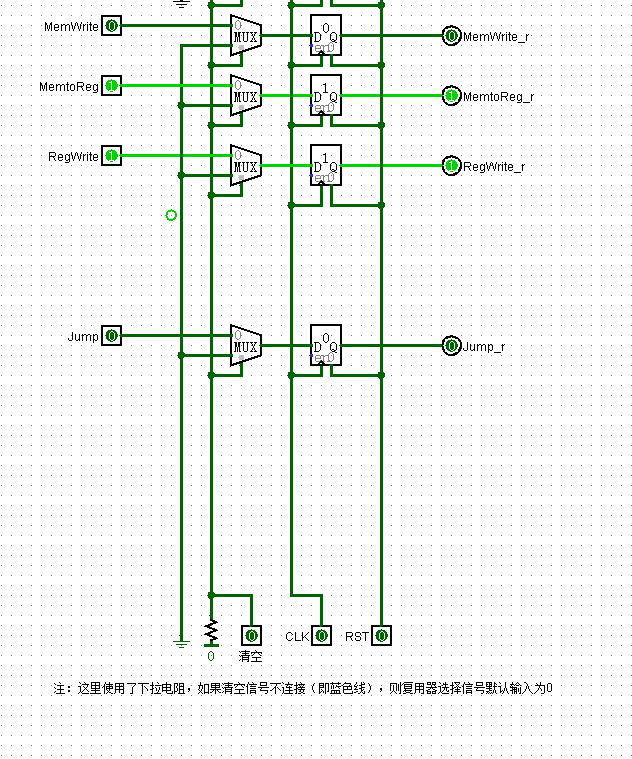
# 三 成果展现

## IF/ID部分

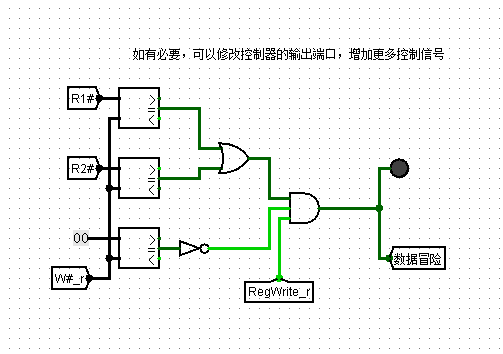


### ID/EX部分：

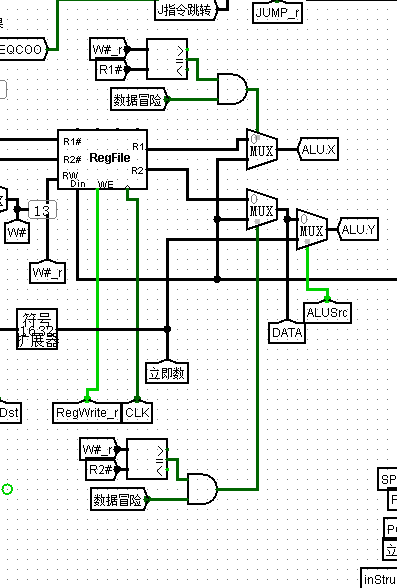




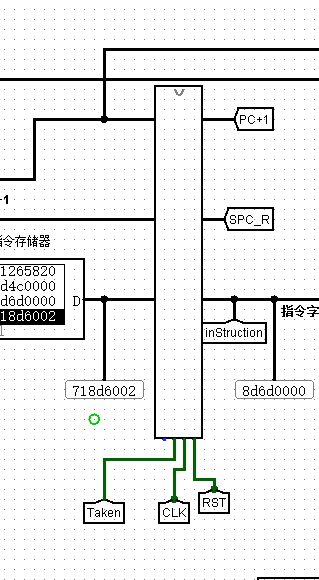
### 数据冒险：



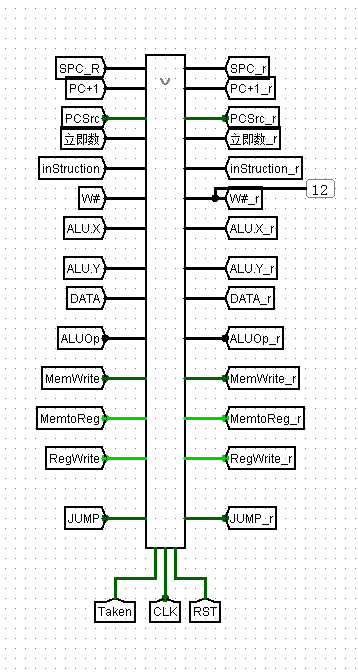
### 数据旁路：



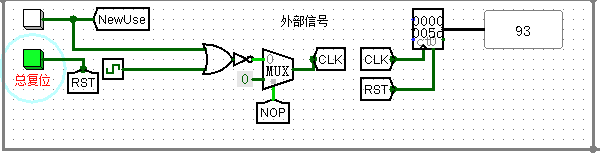
### IF/ID连接部分：

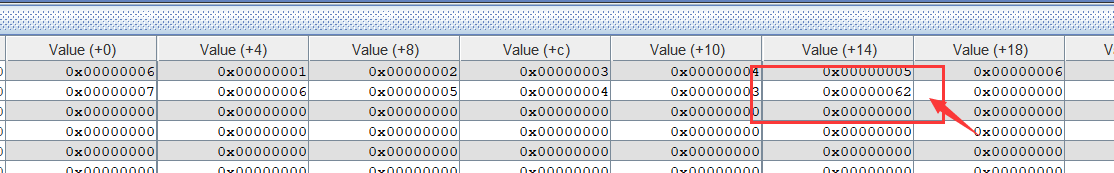


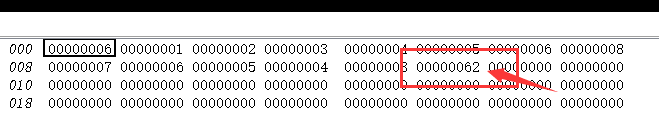
### ID/EX连接部分：



运算部分：



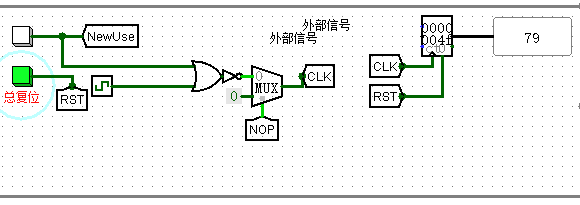


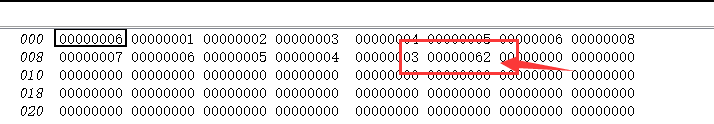


### 性能提升部分：

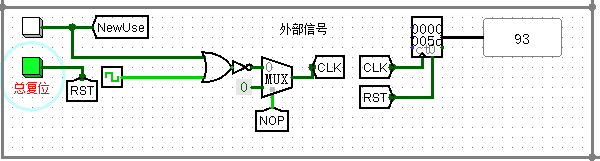
#### 在未使用流水线时：

整体下来总的clk为：79





#### 使用三级流水线时：



#### 性能提升的倍数：

79÷93×3≈2.55

可得性能提升倍率为：2.55

# 四 实验细节

1. 应该把所有的跳转指令都归到一起为Taken,来实现IF/ID中的bubble和ID/EX中的清空。
2. 当PC走到的5c时，表示程序存储器到了结束，应该把CLK记录停止，用NOP来标记这个情况，控制累加器停止计数。
3. 原来所有的接口都应该换成ID/EX转化后的接口，否则会导致程序进程异常。

# 五 实验总结

1. 一开始无从下手，只能看以前的上课的源码包，发现数据旁路和这次的实验很像，就把里面的IF/ID和ID/EX搬运过来修改使用，最后实践可行。
2. 修改后已开启其实不能使用，在反复确认关键连线无误，修改后的组件也可以使用后，我又反复查看数据旁路的源码。使用对照法，发现一开始数据旁路的RegDst和Regwrite信号就是亮着的，这可能会影响后续的RegFile 数据的写入和数据冒险，但后面实践后发现，并没有用处，可能是在这个计算中没有用处。
3. 反复对比后发现本质上的不同，我只能用数据观察法，用探测器观察数据的变化，最后在看W#,我发现W#\_r一直为0，查看W#的连线发现，W#被选择器盖住了，其实没有连上，连上后一切就正常。
4. 通过这一个小小的BUG,我学到了处理错误的不同解法,即通过不同的角度思考问题。