

SHANGHAI JIAO TONG UNIVERSITY



计算机系统结构实验报告

实验名称: 简单的类MIPS多周期流水化处理器实现

姓 名： 洪瑄锐

学 号： 517030910227

班 级： F1703302

手 机： 15248246044

邮 箱： [1204378645@qq.com](mailto:1204378645@qq.com)

目录

[1. 实验目的 3](#_Toc8850511)

[2. 实验原理与代码实现 3](#_Toc8850512)

[2.1 指令流水线的实现 3](#_Toc8850513)

[2.1.1 主要思想 3](#_Toc8850514)

[2.1.2 编写代码 4](#_Toc8850515)

[2.2控制冒险与数据冒险 6](#_Toc8850516)

[2.2.1 数据冒险 6](#_Toc8850517)

[2.2.2控制冒险 6](#_Toc8850518)

[3. 仿真测试 7](#_Toc8850519)

[4. 感想 9](#_Toc8850520)

[5. 参考文献 9](#_Toc8850521)

# 1. 实验目的

1. 理解CPU的Pipeline，了解流水线的相关和冒险（hazard）。
2. 设计流水线CPU，支持Stall。通过检测竞争并插入停顿（Stall）机制解决数据冒险、控制冒险和结构冒险。

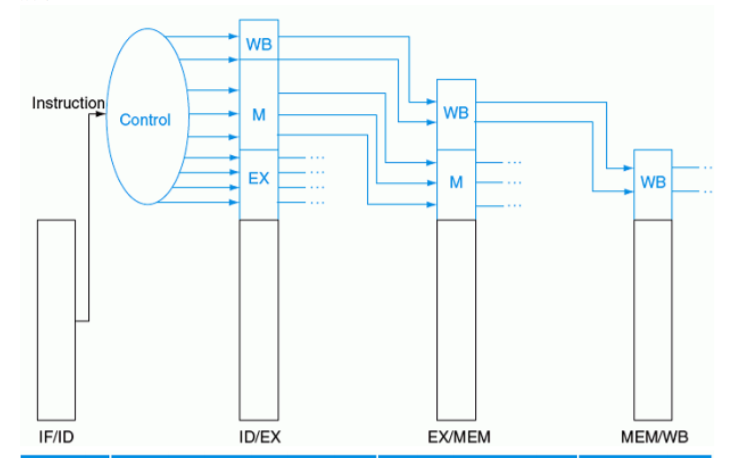
# 2. 实验原理与代码实现

## 2.1 指令流水线的实现

2.1.1 主要思想

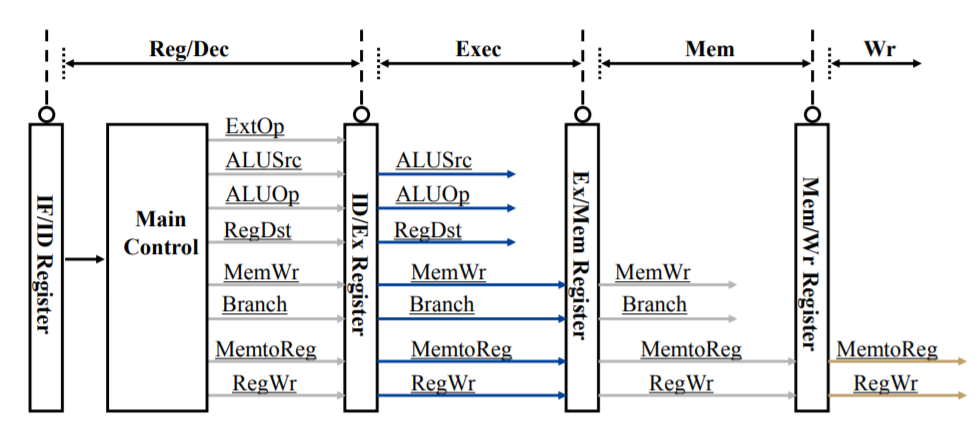
一条指令的执行分为取指（IF），译码（ID），执行（EX），访存（MEM），写回（WB）五个步骤，在单周期CPU中当一条指令执行完所有步骤下一条指令才开始取指，但对于多周期CPU，一条指令在当前周期取指完毕进入下一步骤译码时，该指令的下一条指令立即开始取指，构成指令流水线，大大提高了CPU运行效率。

与单周期CPU实现不同的是，多周期CPU需要段寄存器传递控制信号，主控制单元在译码阶段产生所有的控制信号，EX段需要的控制信号在一周期后使用，Mem段需要的控制信号在两周期后使用，Wr段需要的控制信号在三周期后使用。



如图所示，我们共需要四种段寄存器，连接IF与ID的段寄存器，连接ID与EX的段寄存器，连接EX与MEM的段寄存器，连接MEM与WB的段寄存器。其中ID/EX需要存储ID，EX，MEM，WB所用的控制信号，EX/MEM需要存储EX，MEM,WB所用的控制信号，MEM/WB需要存储MEM，WB需要的控制信号。

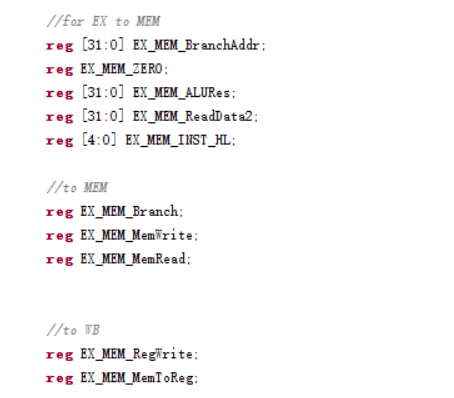
**传递信号如图：**



2.1.2 编写代码

定义各个存储控制信号的寄存器变量，由于各种变量名称极为复杂，所以这里采用 **“源寄存器\_目标寄存器\_信号名称”**的方式，源与目标针对的是信号的传递方向。

**部分代码：**

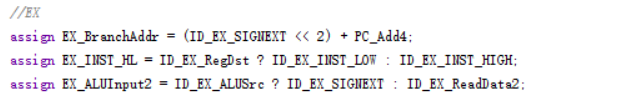


（这里定义了保存从EX传递到MEM,WB的信号的寄存器）

* **模块连接：**

同lab05一样，我们需要利用wire将各个模块的输入输出端口连接起来，比如带符号扩展模块的输出可能是ALU模块的操作数输入。

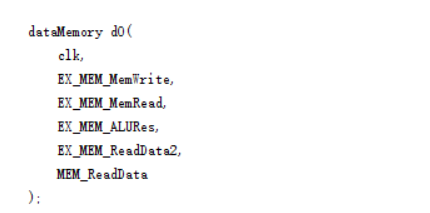
**部分代码：**



* **实例化：**

注意模块端口与寄存器变量一一对应。

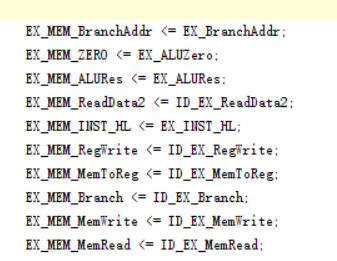
**部分代码：**



* **段寄存器之间的信号传递：**

在lab06中，需要在各个段寄存器之间传递信号的值。因此在更新PC的同时，还要将部分寄存器的值传给下一阶段的寄存器。

**部分代码：**



## 2.2控制冒险与数据冒险

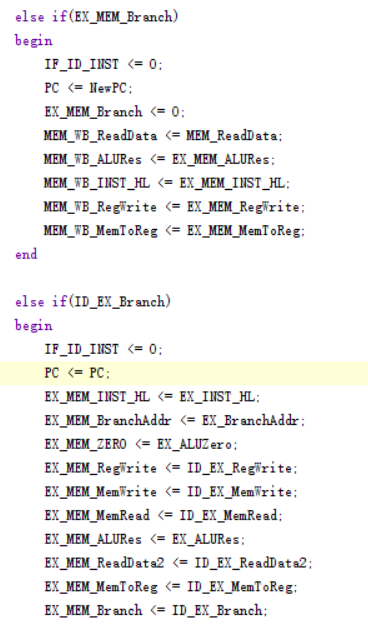
2.2.1 数据冒险

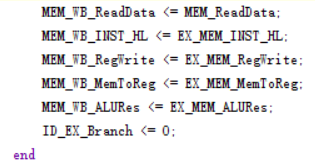
数据冒险：当一条指令需要用到前面某条指令的结果，从而不能重叠执行时，就发生了数据冒险。比如lw指令后为add指令，且add指令的输入来自于lw刚刚写入的寄存器，由于指令流水线，add指令取操作数时lw还没有写入寄存器，导致数据冒险。这里采用在产生数据冒险的指令之间插入空指令进行停顿的做法。

2.2.2控制冒险

控制冒险：当流水线遇到分支指令和其他能够改变PC值的指令时，就会发生控制冒险，因为对于跳转指令，下一个指令的PC值并非原PC+4,但是在指令流水线中，PC默认PC=PC+4，所以在编写代码时需要对分支指令进行辨别和处理。这里采用的方法是检测是否有跳转指令，如果有那么暂停流水线，直到branch的结果和真正的跳转之后的指令地址出现再继续流水线。

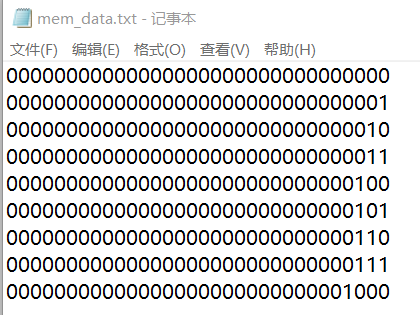
**处理代码：**



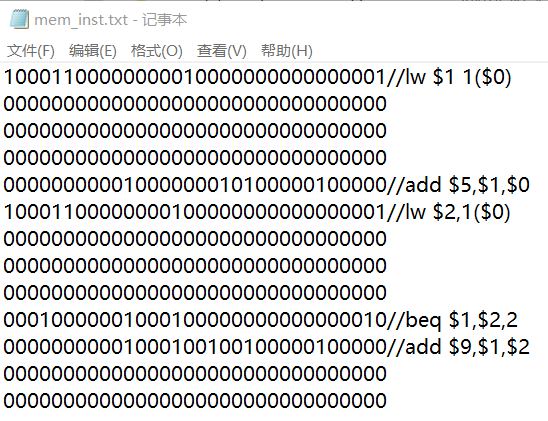


# 3. 仿真测试

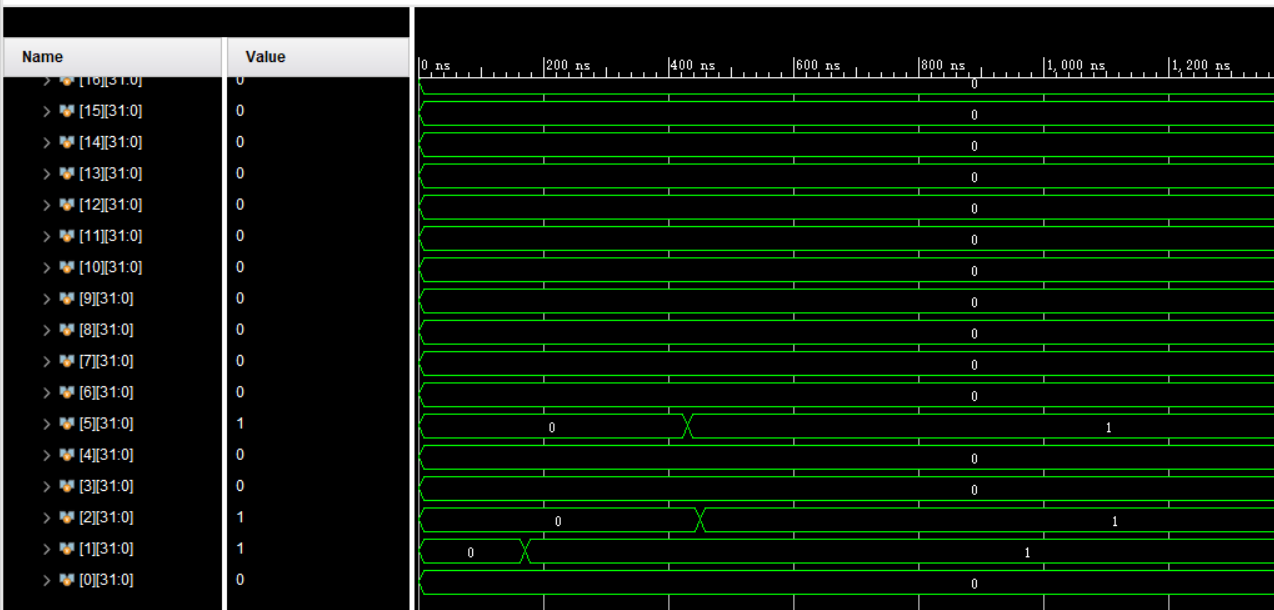
* mem\_data.txt：



* mem\_inst.txt



* **仿真结果：**



可以看出5号寄存器写入1，避免了数据冒险，9号寄存器无变化，说明beq跳转指令成功跳转，没有执行下一条add指令。仿真成功。

# 4. 感想

lab06相较lab05难度又有了很大的提高，lab06实现了指令的流水线，相当于性能得到了提高。在编写代码的过程中，我认识到严谨地做好每一步的重要性，不仅要思维严谨，变量定义和端口相连也要严谨，否则就会出错。在做实验的过程中，我对cpu的认识不再只局限于书本之上，当我真正实现了简单的cpu时，我才觉得自己真正地对计算机系统结构这部分的内容懂得了，掌握了。特别感谢老师和同学在实验中给予的教导和启发。

# 5. 参考文献

《cs145实验指导书lab06》