实验六

一、实验名称

简单的类 MIPS 多周期流水化处理器实现

二、实验目的

理解 CPU 的 pipeline，对 Data Hazard， Branch Hazard 有初步认识

三、实验范围

1. ISE 的使用

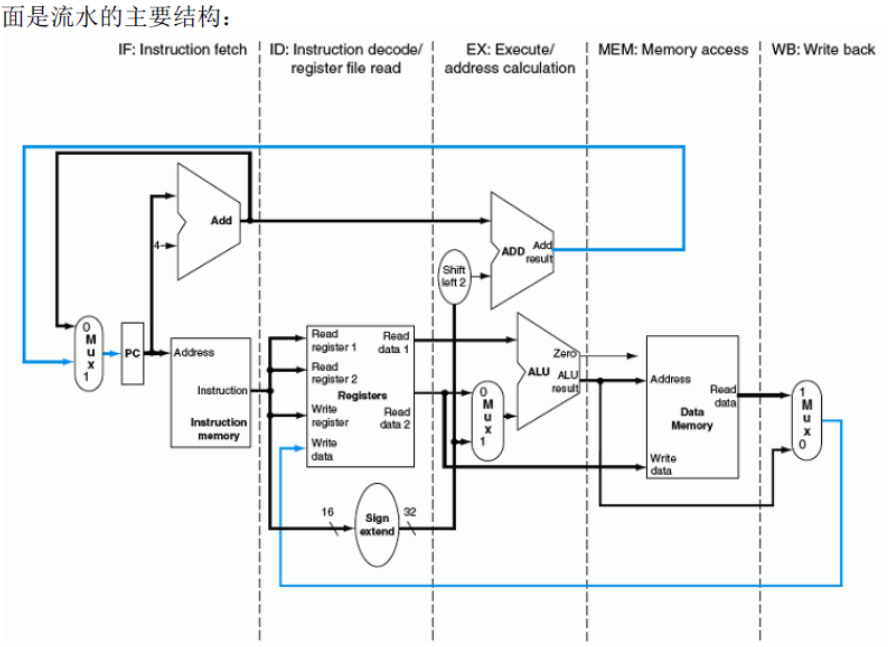
2. VirtexII Pro 实验板的使用

3. 使用 VerilogHDL 进行逻辑设计

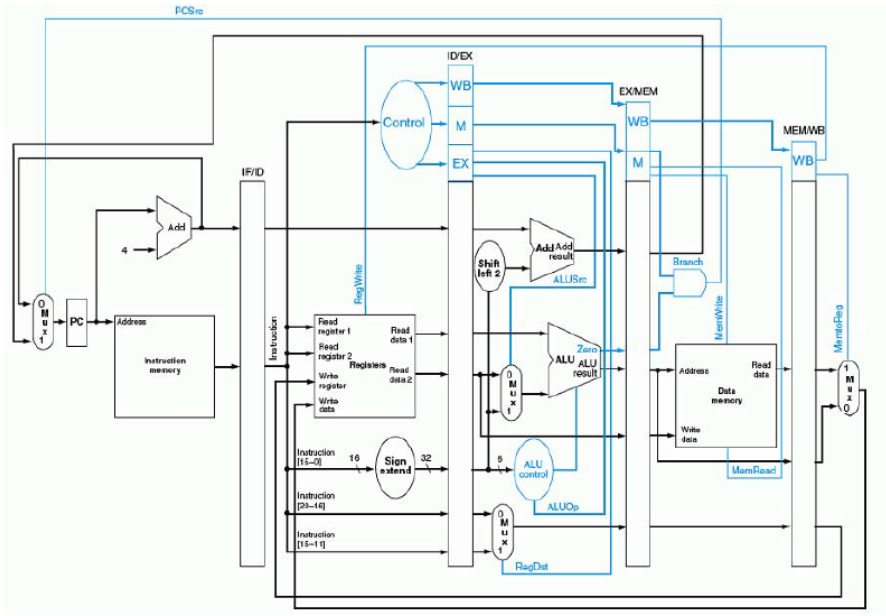
四、主要的设计思想和测试仿真

以下是流水线的主要结构

下面是流水的主要结构：



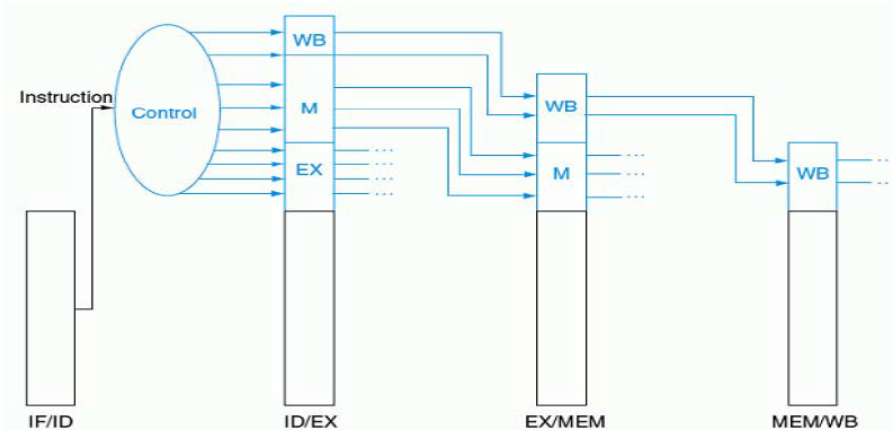
将单周期 CPU 进行分割，插入 4 级寄存器，将其分割为 IF，ID，EX，M，WB 五大部分：



首先需要了解流水线处理器的基本原理：将一条 MIPS 指令的每个步骤都作为一个周期来执行：IF 从指令存储器中读取指令， ID 进行译码和读取寄存器的操作， EX 执行 ALU 计算，MEM 读取存储器或写存储器， WB 将结果写回寄存器。

如此一来，在一条指令执行时，其他的单元也不会空闲，可以执行下面指令的操作，能够大大提高吞吐率。

在这里引入了 4 个中间寄存器单元：



虽然流水线的处理使得效率提高了，但是同时由于结构较为复杂

带来一些问题，比如数据冒险和控制冒险。但是由于时间和能力有限，我并没有解决冒险的问题，只是完成了流水线的连线，能够进行基本指令的进行。

仍然采用开始两次实验的模块，在此基础上在 top 模块逐步实现。

连线时主要的困难还是对于流水线上理解的问题，中间寄存器需要保存一些中间数据，这些数据为 reg 类型，后面需要使用。而中间寄存器产生的控制信号，这些是 wire 类型，用于各个单元之间的连线。

比如在 MEM/WB 中

```
//wire
  wire [31:0] MEM_ReadData;
  wire MEM_PCSrc;
//reg
  reg [31:0] MEM_WB_ReadData;
  reg [31:0] MEM_WB_AluRes;
  reg [4:0] MEM_WB_WriteReg;
  reg MEM_WB_MemToReg;
  reg MEM_WB_RegWrite;
```

而对于连接前两次实验的模块，这与实验五完全一样，故不再赘

述。

测试仿真（普通的指令，数据冒险通过代码中添加阻塞，其实并没有实质解决，没有涉及控制冒险）：

10101100000000100000000000000011 //sw $1, 3($0)

10001100000000010000000000000101 //lw $1, 5($0)

10001100000001000000000000000010 //lw $4, 2($0)

00000000000000000000000000000000 //nop

00000000000000000000000000000000 //nop

00000000000000000000000000000000 //nop

00000000100000010001000000100000 //$2 = $4 + $1

00000000001001000001100000100010 //$3 = $1 - $4

00000000000000000000000000000000 //nop

00000000000000000000000000000000 //nop

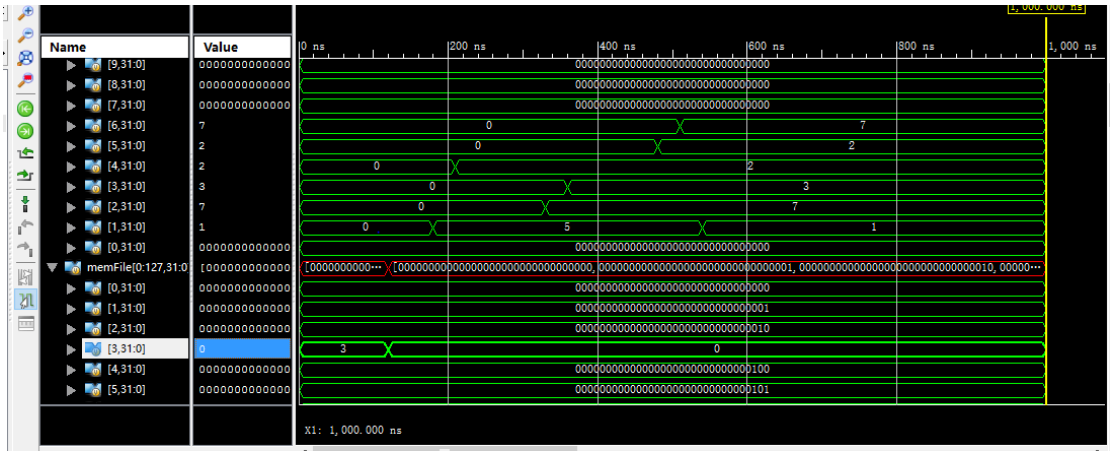
00000000000000000000000000000000 //nop

00000000011001000010100000100100 //$5 = $ 3 & $4

00000000001001000011000000100101 //$6 = $1 | $4

00000000001000100000100000101010 //$1 = $1 slt $2

仿真波形如下：



五、上板验证

上板验证的思路与实验五基本一样。

同样是通过两个开关控制显示 PC 或 3 个寄存器的值的变化。

六、心得体会

这次的实验六由于在实验五的铺垫之下完成地还算没有太大的困难，调试遇到一些问题的时候有实验五的经验一点点的改正，完成了流水线的基本连线，能够运行一些基本的指令。

由于各种原因，我还是放弃了解决数据冒险，控制冒险的尝试。感觉在实验五上花去很多的时间，留给实验六的时间本就不多，本身实验六连线就已经足够复杂。但我看到许多同学已经完成了 forward 机制，不得不佩服他们。而且事实上，设置实验六的主要目的就是要让我们更加深刻理解冒险并解决它，否则就和实验五没什么区别了。但是，最后还是没能完整地完成实验六，确实是一个遗憾，希望自己能够慢慢思考将其完成，或者询问已完成的同学他们的想法，在此基础上继续完善。

就这样，这次计算机组成实验结束了，虽然并不尽如人意，但还是靠着努力基本完成了要求。可以说，从一开始实验三，四的单个模块，到完成整个处理器的总体设计，最后上板验证，这个要求是一步一步加强的，从底而上对代码提出越来越高的要求，所以说在一开始就要尽可能保证代码的正确性，这样在后面的编程过程中不至于还要来改正底层的代码。这也是我第一次接触硬件相关的编程，觉得蛮有趣，但需要付出大量的努力，希望能继续努力，学到更多的东西。