

Department of
**Computer Science
& Engineering**



计算机组成实验指导书-LAB6

1. OVERVIEW

1.1 实验名称

简单的类 MIPS 多周期流水化处理器实现

1.2 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、控制竞争和结构冒险
- 3) 在 2) 的基础上，增加 Forwarding 机制解决数据竞争，减少因数据竞争带来的流水线停顿延时，提高流水线处理器性能 (选做)
- 4) 在 3) 的基础上，通过 predict-not-taken 或延时转移策略解决控制竞争，减少控制竞争带来的流水线停顿延时，进一步提高处理器性能（选作）
- 5) 在 4) 的基础上，将 CPU 支持的指令数量从 16 条扩充为 31 条，使处理器功能 更加丰富（选做）


1.3 实验报告与验收办法

需提交纸电子版报告，实验完毕需验收登记

2. 新建工程



2.1 实验描述

2.1.1 新建工程

 New Project



New Project Summary

-  A new RTL project named 'lab06' will be created.
-  The default part and product family for the new project:
 - Default Part: xc7k325tffg676-2
 - Product: Kintex-7
 - Family: Kintex-7
 - Package: ffg676
 - Speed Grade: -2

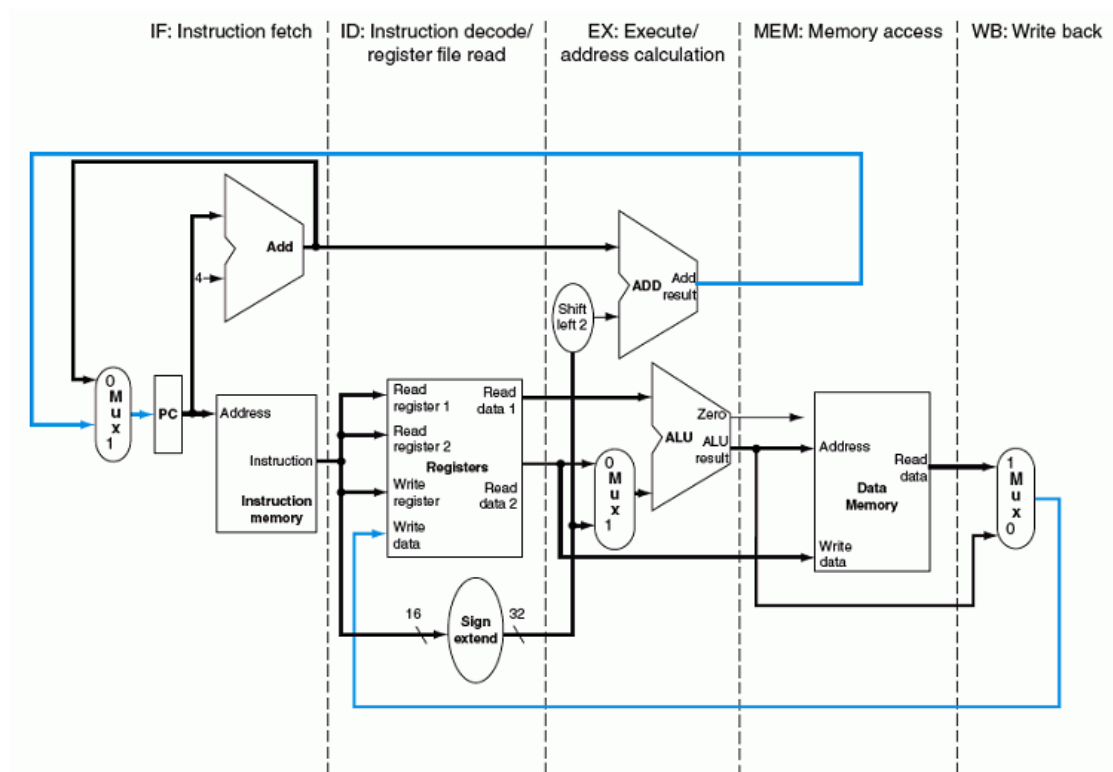
3. TOP 模块

3.1 实验描述

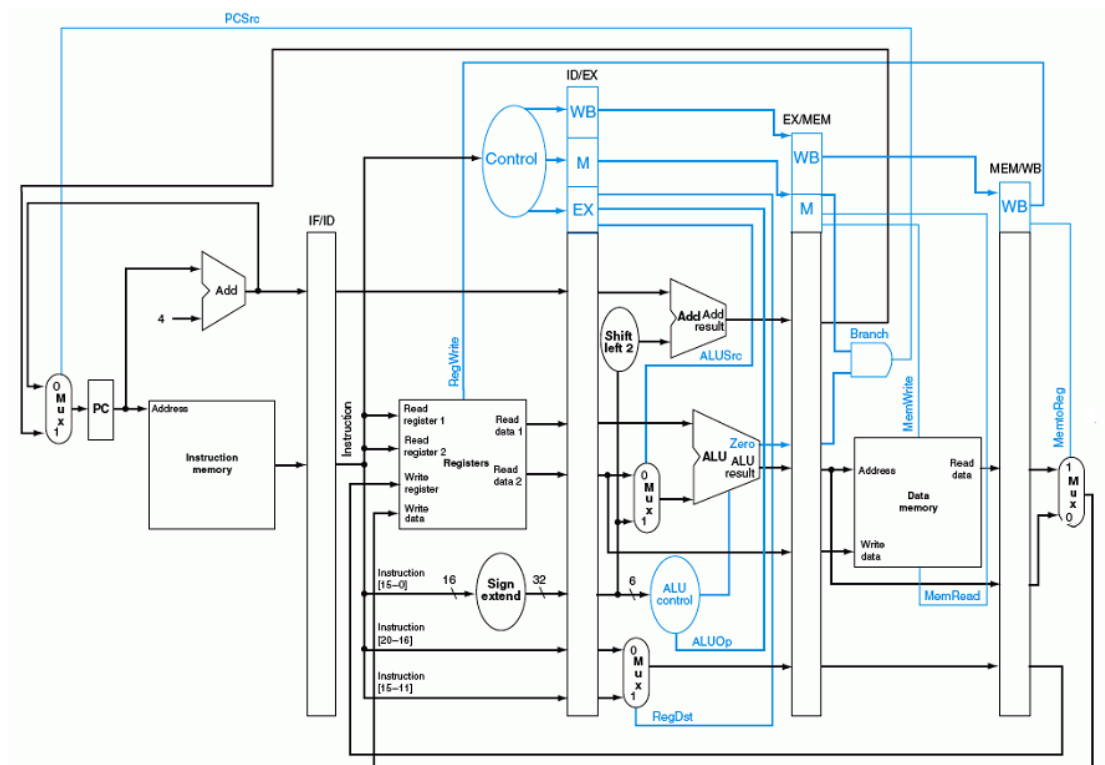
由于前几次实验已经完成了 CPU 各部件的主要功能，因此只需要完成 Top 模块（包括修改 Control 模块，以及修改模块间互联的定义等）。

3.1.1 模块描述

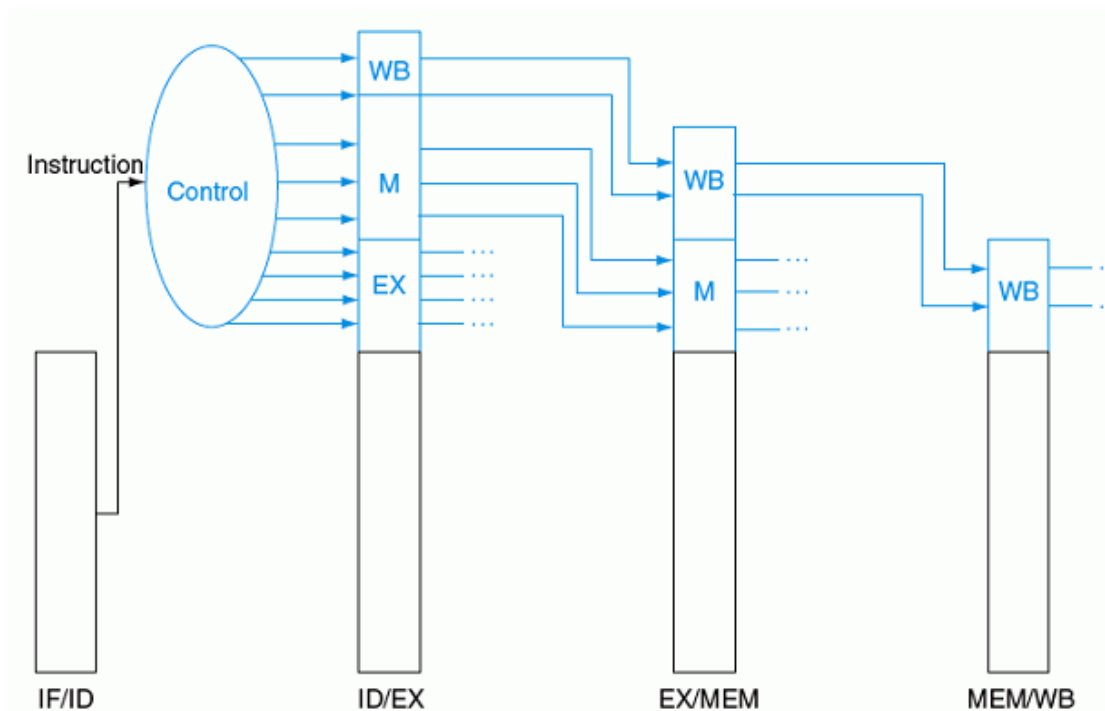
下面是流水的主要结构：



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：



其中 Control 的输出需要被加入流水线寄存器保存下来，以供后续每级流水使用。如下所示：



Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg Write	Mem to Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

3.1.2 添加先前已经完成的诸模块

包括指令内存模块

3.1.3 新建 Top 模块

编写 Top 模块:

注意, 由于各种变量名称极为复杂, 推荐在着手编码之前为自己选择一套命名规范。

另外, 由于 MEM 级的 Branch 会影响 PCSrc 的值, 从而影响下次 PC, 因此需要为 Control 加入 RESET 功能, 将 Branch 置零。

3.1.4 仿真测试

1. 编写汇编代码, 推荐使用自己编写的测试程序, 也可以使用下面的程序, 注意, 需要去除中间的 Data Hazard:

```
lw $1, 40($0) ; 1
lw $2, 44($0) ; 5
lw $3, 48($0) ; 8
add $4, $1, $2 ; $4=6
sub $5, $3, $1 ; $5=7
and $6, $2, $1 ; $6=1
lw $10, 40($0) ; 1
lw $10, 40($0) ; 1
lw $10, 40($0) ; 1
or $7, $3, $1 ; $7=9
slt $8, $3, $1 ; $8=0
beq $0, $0, end; to end
add $9, $7, $8 ; $9=9, not executed
end:
lw $10, 40($0) ; 1
lw $10, 40($0) ; 1
lw $10, 40($0) ; 1
lw $10, 40($0) ; 1
lw $10, 40($0) ; 1
```

2. 将上述代码转为二进制 Codes, 保存为文件, 文件名自定, 这里假定是 instruction.txt;
将下列数据

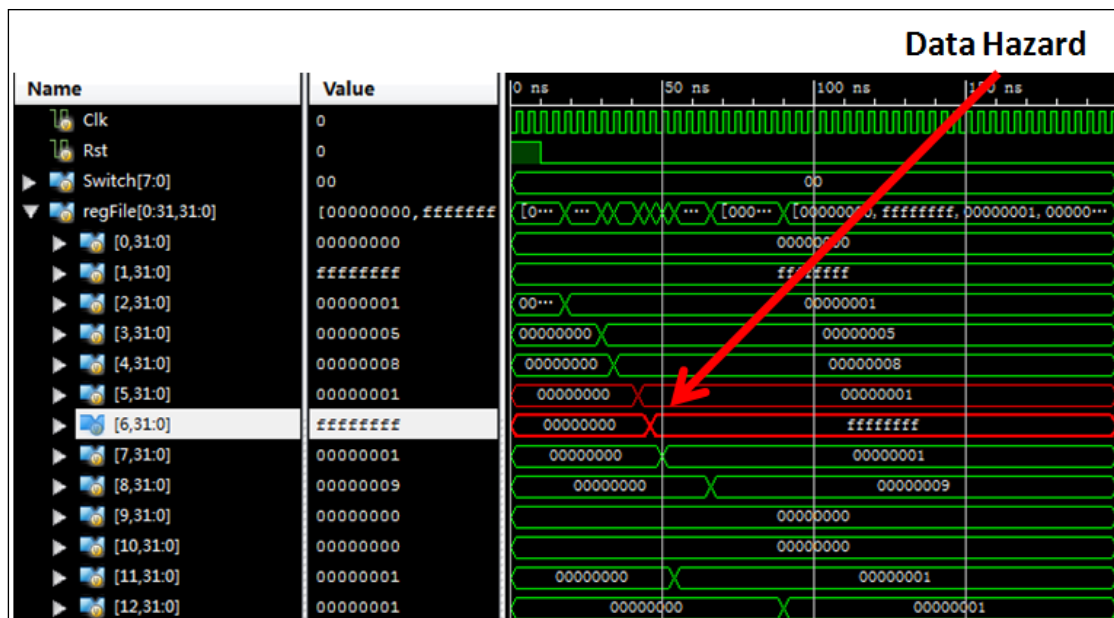
1
5
8

假设保存为 data.txt。

将 instruction.txt 和 data.txt 放在工程目录下，也可在 Top 中加入下面代码：

```
Initial begin
  $readmemb("instruction.txt", InstMemFile);
  $readmemb("data.txt", memFile);
end
```

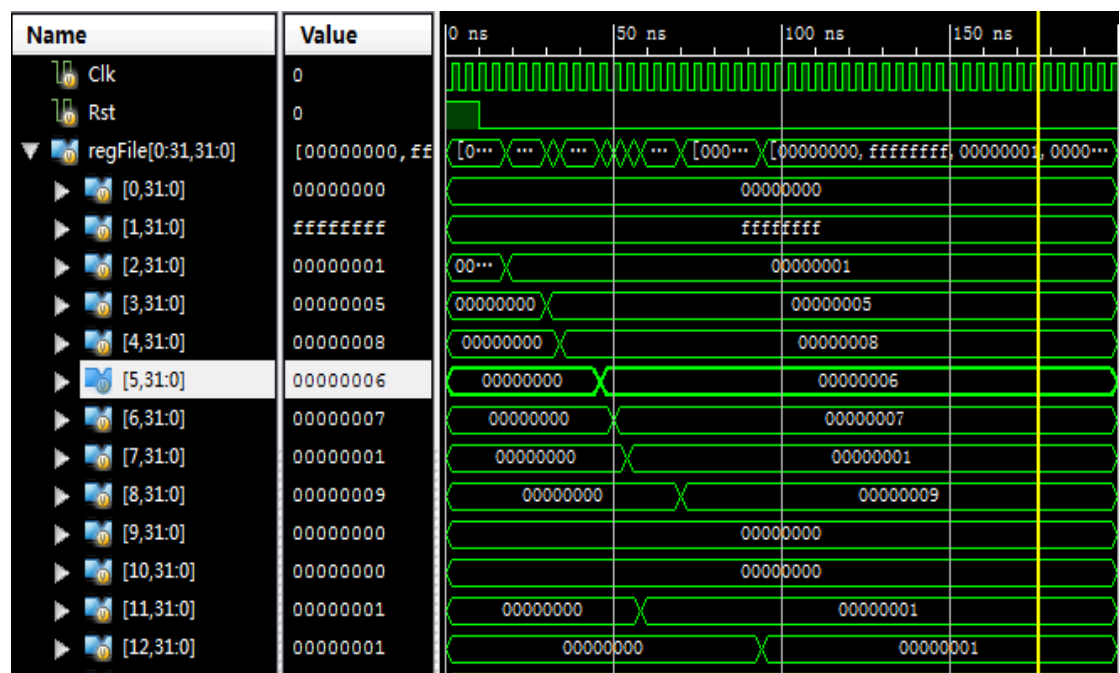
3. 编写 Top 层的测试文件，添加时钟激励和其它输入信号并初始化
4. 下面给出一个参考样例：



出现了 Data Hazard

观察上述波形可知，\$5 寄存器的值是 0x00000001 和\$6 寄存器的值是 0xffffffff；但由汇编指令计算可知，\$5 寄存器的值应该是 0x00000006，\$6 寄存器的值应该是 0x00000007。这是由于流水化处理后，当前指令用到了前面指令尚未计算完成的寄存器值，导致计算结果出错，这就是 Data Hazard。

5. 修改指令后，重新仿真，可看到如下样例波形：



用简单方法消除了 Data Hazard

3.2 实验报告