版本

更新记录	文档名	实验指导书_ lab5		
	版本号	0.3		
	创建人	计算机组成原理教学组		
	创建日期	2022/1/1		
更新历史				
序号	**更新日期**	**更新人**	**版 本号**	**更新内容**
1	2022/1/1	陈颖琪	0.1	RISC-V 流水线CPU模型虚拟仿真
2	2022/2/28	陈颖琪	0.2	添加简短测试代码段
3	2022/4/20	陈颖琪	0.3	修订实验步骤5
4	2022/5/7	陈颖琪	0.4	修订拓展思考，概念回顾

文档错误反馈:

本文档出现错误请联系:

Yingqichen@sjtu.edu.cn

实验 5　基于 RISC-V 流水线 CPU 的指令执行过程在线仿真与冒险处理方式研究

1、实验目的

1、 理解流水线 CPU 指令执行过程。
2、 理解流水线冒险处理的概念。
3、 理解不同流水线硬件结构对冒险处理方法的区别。

2、流水线冒险处理概念回顾

1、 暂停流水线 stall

用于控制暂停 PC 值的更新，PC 值维持上一个时钟状态，波形上可见 pc 信号出现持续 2 个时钟周期的 bubble。参见图 1 。

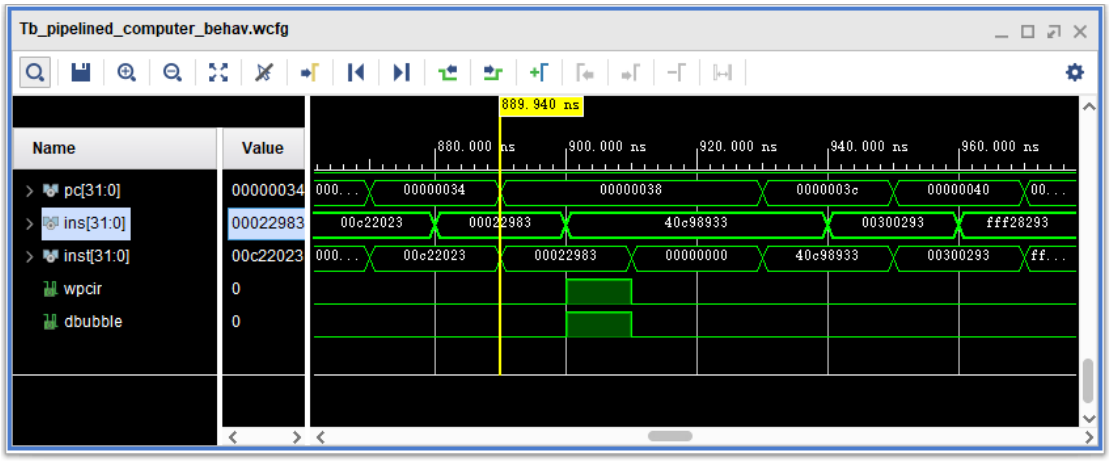


图 1 pc 出现持续 2 个时钟的 bubble

2、 刷新流水线 flush

清除已经从指令存储器读取到的指令，不执行指令操作，清零指令数据，或者指令译码产生的控制信号全部清 0。参见图 2。

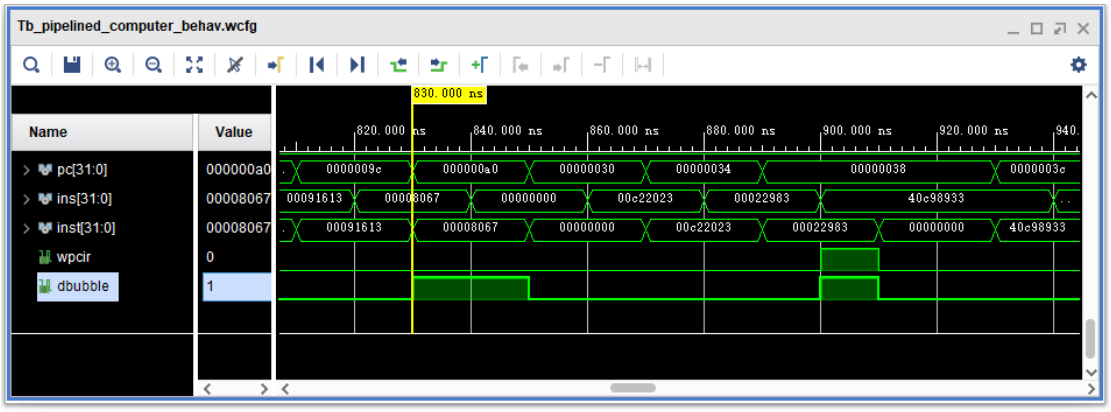


图 2 从指令寄存器读出的 ins 被 flush 操作设置为 0

3、 **前推 forward**

提前将本条指令经过 ALU 运算得到的数据，或者从数据存储器读出来的数据，不经过写入寄存器堆，就直接送到 ALU 输入端，提供给下一条或两条指令用于该指令的执行过程。

图 3 所示为 ripes 软件中实现的一种结构的示意图。图 4 为另一种实现结构中具体的控制信号波形示意。该结构将紫色框的逻辑前移至 ID 阶段，可理解为 fwda、fwdb 信号就是用于控制类似图 3 红框中的多路选择器。fwda 信号的值为 0，1，2，3 分别对应了多路选择器选取的数据来源如图 3 中红、蓝、黄、绿四条彩色数据通路所示。

由此也可理解实现的具体结构可以是有多种不同的形式的。但都能够完成对 RISC-V 架构的支持。体现在实现细节的复杂性以及效率等方面上有所差别。例如同样程序段执行所用的时钟数可能不一样。

也可查看本次练习要用到的另一个仿真平台所示的结构框图对比思考，加深对以上概念的理解。

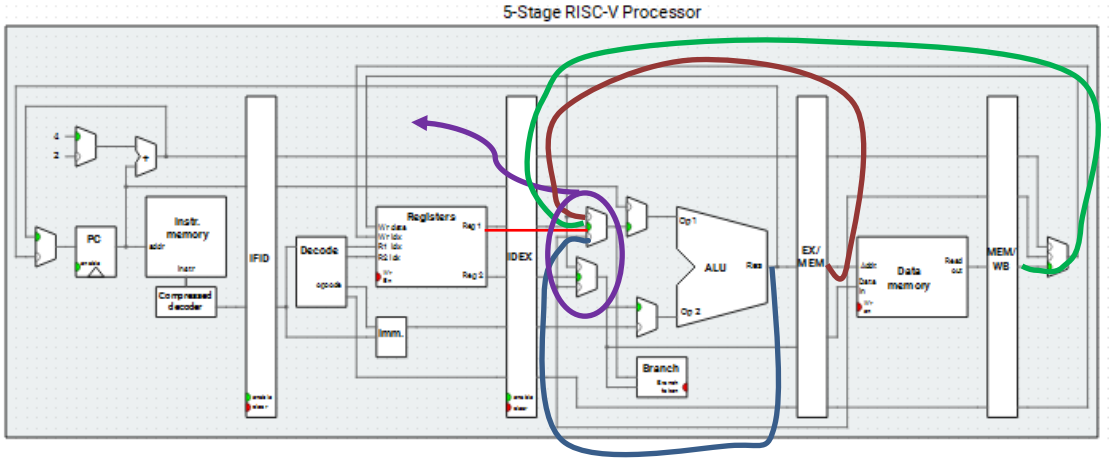


图 3　一种五级流水线结构中的 forward 功能模块示意

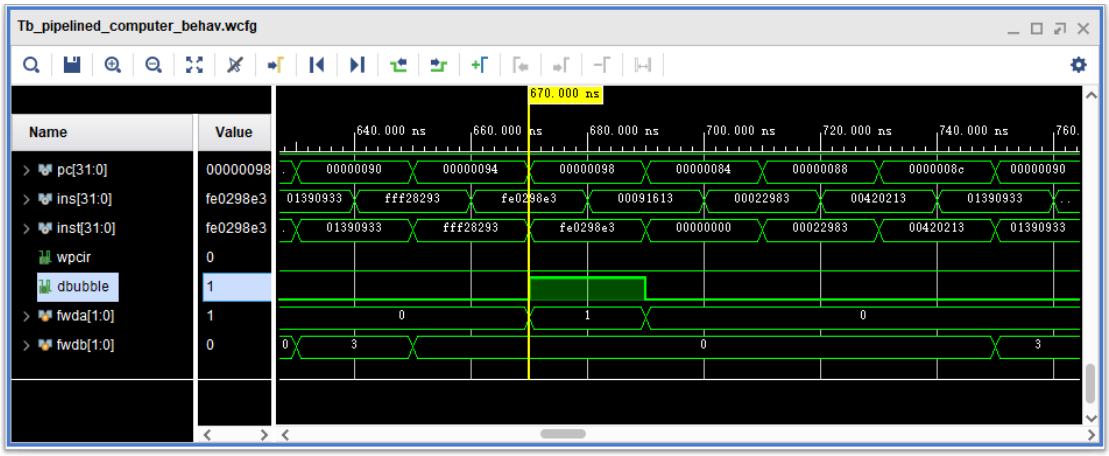


图 4 另一种五级流水线结构中的 forward 功能实现中的控制信号示意

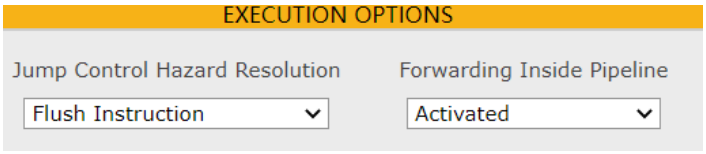
3、实验要求与步骤

3.1 测试代码 1 的仿真

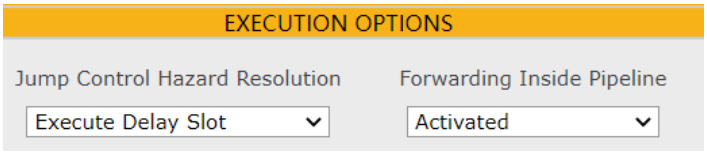
选择仿真平台 2，完成对下面所列的测试代码 1 的仿真并填写表 1 和表 2。分四种模式分析五级流水线 CPU 执行过程：模式 1：with forward with flush，模式 2：no forward with flush，模式 3：with forward no flush，模式 4：no forward no flush。

1、 在界面上选择不同的冒险处理方式的执行选项，例如：

模式 1：



模式 3：



你可自行选择更多选项组合。

2、分不同模式执行仿真，查看程序执行后的 execution table，并截图，指出执行的周期数，对比分析不同模式下执行周期数不同的原因。指出被执行了 forward 或者 flush 操作的具体指令和原因。 完成表 1。

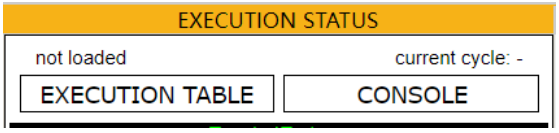


表 1

代码段 1	模式 1： with forward with flush	模式 2： no forward with flush	模式 3： with forward no flush	模式 4： no forward no flush
执行周期数	**19**	**20**	**19**	**22**
被执行 forward 的指令，执行 forward 次数和原因	addi x6,x6,2 beq x6,x0,fi 执行 **1** 次 不需要等待 addi 指令将 x6 写回寄存器就直接从 ALU 将结果前递给 beq 使用。	/	addi x6,x6,2 beq x6,x0,fi 执行 **1** 次 不需要等待 addi 指令将 x6 写回寄存器就直接从 ALU 将结果前递给 beq 使用。	/
被执行 flush 操作的指令和原因	j loop fi: add x4,x4,x5 执行 **3** 次 默认分支不跳转，当发现分支需要跳转时清除已有指令。	j loop fi: add x4,x4,x5 执行 **3** 次 默认分支不跳转，当发现分支需要跳转时清除已有指令。	/	/

4、 分析解释不同选择方式下，执行同一段程序的差别。指出不同模式执行后用到的几个寄存器的值，分析执行结果是否正确。如果不正确，是由什么引起的？如何修改可以得到正确结果？给出修改后的执行结果。

执行对代码段 1 仿真后，分析并完成表 2。

表 2

代码段 1	分析：单周期 CPU 架构下，执行需（ 70 ）个时钟周期， 执行后，x6=（ 0 ），x5=（ 6 ），x4=（ 6 ）			
实际执行仿真情况	模式 1： with forward with flush	模式 2： no forward with flush	模式 3： with forward no flush	模式 4： no forward no flush
Reg X6=	0	0	-1	-1
Reg X5=	6	6	6	6
Reg X4=	6	6	15	15
执行结果正确与否	正确	正确	不正确	不正确
若不正确 如何修改	\	\	由于没有 flush，导致默认的分支不跳转预判每次都被完整执行。即每轮循环都会错误执行一遍 fi: add x4,x4,x5 指令（导致 x4 错误）；且 addi x6,x6,-1 指令也被多执行了一次（导致 x6 错误）。 一种解决方案是增加冗余的指令。如在 beq 语句后加 addi x6, x6,0；在 j 语句后加 addi x4, x4, 0。（时钟周期数会增加）	

5、 自行寻找你认为可能出现数据冲突和控制冒险的代码段，完成同上的仿真，查看流水线 CPU 处理冒险的过程，验证你的理解是否正确。（选做）

3.2 测试代码 2 的仿真

1、分析测试代码段 2，如果考虑五级流水线 CPU 执行过程中对数据和控制冒险冲突的处理，也即采用 with forward with flush 的模式，指出具体 PC 为哪些的指令会被做数据冒险处理？PC 为哪些的指令会被做控制冒险处理？被做数据冒险处理对应的操作是什么？被做控制冒险处理对应的操作是什么？对测试代码段 2 进行单步仿真，在你指出的指令的执行时间点做截图，验证你的想法。并填写表 3。

表 3

PC 值	指令	冒险的种类	执行的操作
04	ori x4, x10, 0	数据冒险	从 00 指令前递 x10 的值
30	sw x12, 0(x4)	控制冒险	flush 清除指令
8c	add x18, x18, x19	数据冒险	从 84 指令前递 x19 的值
94	bne x5, x0, loop	数据冒险	从 90 指令前递 x5 的值，并 stall 一个周期
98	slli x12, x18, 0	控制冒险	flush 清除指令
38	sub x18, x19, x12	数据冒险	从 34 指令前递 x19 的值，并 stall 一个周期
40	loop2:addi x5, x5, -1	数据冒险	从 3c 指令前递 x5 的值
44	ori x18, x5, -1	数据冒险	从 40 指令前递 x5 的值
48	xori x18, x18, 1365	数据冒险	从 44 指令前递 x18 的值
50	andi x20, x19, -1	数据冒险	从 4c 指令前递 x19 的值
54	or x16, x20, x19	数据冒险	从 50 指令前递 x20 的值
5c	and x17, x20, x16	数据冒险	从 54 指令前递 x16 的值
68	shift:addi x5, x0, -1	控制冒险	flush 清除指令
6c	slli x18, x5, 15	数据冒险	从 68 指令前递 x5 的值
70	slli x18, x18, 16	数据冒险	从 6c 指令前递 x18 的值
74	srai x18, x18, 16	数据冒险	从 70 指令前递 x18 的值
78	srli x18, x18, 15	数据冒险	从 74 指令前递 x18 的值
80	sum:add x18, x0, x0	控制冒险	flush 清除指令

2、对测试代码段 2 选择不同流水线模式完成仿真，各个模式下第一次执行完最后的死循环跳转指令分别用了多少个时钟？对比并分析用不同模式以及用单周期 CPU 执行该段程序的过程。（选做）。

4、测试代码

测试代码段 1：

```
addi x6,x6,2
loop: beq x6,x0,fi
addi x6,x6,-1
addi x5,x5,3
j loop
fi: add x4,x4,x5
```

测试代码段 2：此处所列测试代码 2 与 lab2 的 2.1.2 节所列代码一致。数据空间基地址换为了 1024，便于在仿真平台上观察数据存储器内数据变化。

```
lui x10, 0
ori x4, x10, 1024
addi x25, x0, 1
addi x26, x0, 2
addi x27, x0, 3
```

```
addi x28, x0, 4
sw x25, 0(x4)
sw x26, 4(x4)
sw x27, 8(x4)
sw x28, 12(x4)
addi x5, x0, 4
call:
jal sum
sw x12, 0(x4)
lw x19, 0(x4)
sub x18, x19, x12
addi x5, x0, 3
loop2:
addi x5, x5, -1
ori x18, x5, -1
xori x18, x18, 1365
addi x19, x0, -1
andi x20, x19, -1
or x16, x20, x19
xor x18, x20, x19
and x17, x20, x16
beq x5, x0, shift
j loop2
shift:
addi x5, x0, -1
slli x18, x5, 15
slli x18, x18, 16
srai x18, x18, 16
srli x18, x18, 15
fi:
j fi
sum:
add x18, x0, x0
loop:
lw x19, 0(x4)
addi x4, x4, 4
add x18, x18, x19
addi x5, x5, -1
bne x5, x0, loop
slli x12, x18, 0
jr ra
```

5、拓展思考

请思考图 1 也即 lab3 实验要求书中的单周期 cpu 结构图图 2，按照 5 级流水线如何划分？试着在图上标画。其中左侧四选一 MUX 应该算作哪一级？右上方两个加法器可以在 ID 级里面实现吗？最右侧的二选一 MUX 是否一定要放在最后一级里面实现？如果放在 EXE 级里是否可以？forward 操作是否可以在 ID 级实现？stall 操作应该控制什么模块？flush 操作应该控制什么模块？图中蓝色粗线代表了第一级 PC 寄存器位置。可类似画出后面的几级寄存器应该加在哪里。

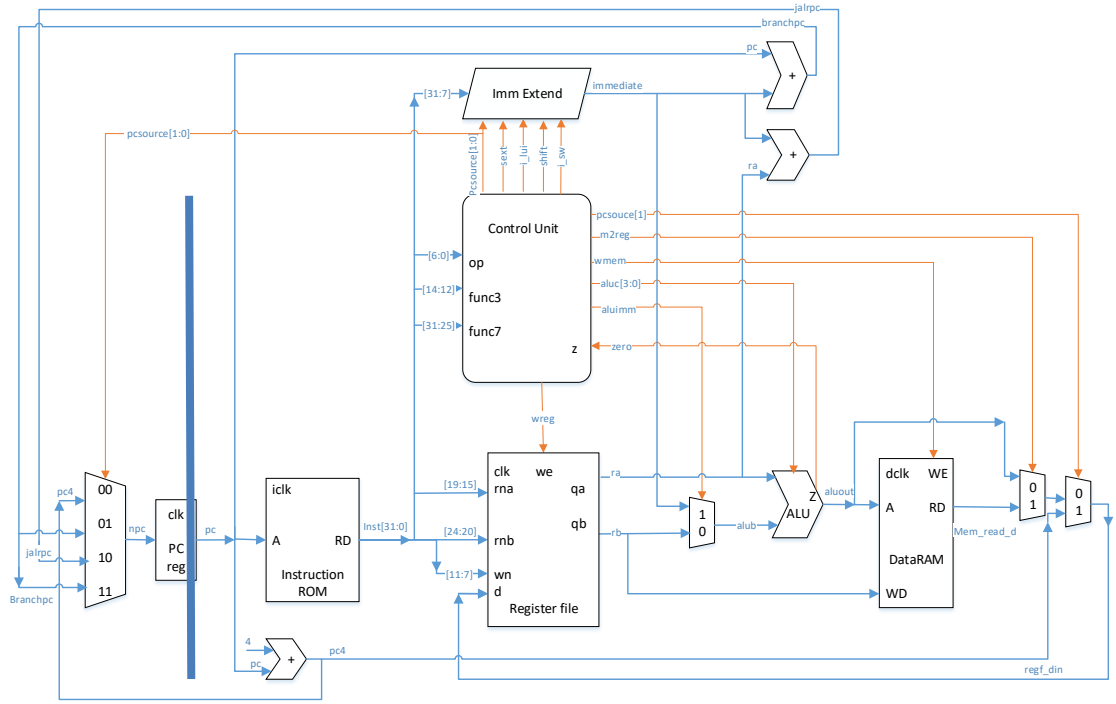


图 1:（lab3 要求书中的图 2）支持 22 条指令的单周期 CPU 框架图

6、仿真平台 2 备用链接

仿真平台 2：WebRISCV

RISC_V 架构 RV32/64IM 五级流水线 CPU 模型在线仿真平台。

https://webriscv.dii.unisi.it/index.php （选择 RV32IM 模式）

或者

http://10.119.1.50:81/ （在校外连接时需要开启交大 VPN）