Lab5.1 流水线CPU在线虚拟仿真实验 课堂练习

#### 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. 理解流水线硬件结构及对冒险处理方法的多样性。

#### 2、课堂练习

2.1练习一

在 RISC V仿真平台

<https://webriscv.dii.unisi.it/index.php>

或

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

选择 RV64IM，Forwarding activate，branch hazard handling Flush Instruction模式

输入代码段（参考课本习题 4.22）：

sd x29,1024(x16) //sw x29,1024(x16) 选择RV32IM时候可用sw替换

ld x29,1024(x16) //lw x29,1024(x16) 选择RV32IM时候可用lw替换

sub x17,x15,x14

beq x17,x0,label

add x15,x11,x14

sub x15,x30,x14

label:nop

1. 单步执行仿真，给出execution table截图。从这个图可以看出倒数第3和2句（beq x17,x0,label和add x15,x11,x14）被执行了什么特殊处理？为什么？

* 从图中可以看到，beq x17,x0,label被执行了stall处理，因为此时a7的值还在exe阶段进行计算，所以还无法通过前递获取，必须进行stall。
* add x15,x11,x14被执行了flush操作。这是因为上一指令的beq分支预测错误，所以需要将beq下一条指令flush，跳转到beq指令中PC地址。

表格

描述已自动生成

2）在ripes仿真环境下做一次五级流水线CPU仿真，参照表1前三列的前三个时钟执行过程画出完整五级流水线（F、D、X、M、W）执行表。与上面的执行表截图作对比。分析有什么异同。

同：两处执行所需的时钟周期均为11，且均发生了flush。

异：Ripes上发生了两处flush，而webrisc-v 则只发生了一次flush，而且有一处stall。

Ripes发生两次flush是因为Ripes的分支跳转判断是在exe阶段才判断的，这自然也不用stall。

表1

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction\clock cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| sd x29,1024(x16) | F | D | X | M | W |  |  |  |  |  |  |
| ld x29,1024(x16) |  | F | D | X | M | W |  |  |  |  |  |
| sub x17,x15,x14 |  |  | F | D | X | M | W |  |  |  |  |
| beq x17,x0,label |  |  |  | F | D | X | M | W |  |  |  |
| add x15,x11,x14 |  |  |  |  | F | D |  |  |  |  |  |
| sub x15,x30,x14 |  |  |  |  |  | F |  |  |  |  |  |
| label:nop |  |  |  |  |  |  | F | D | X | M | W |

2.2 练习二

在五段流水线体系中执行以下RISC-V程序指令段：

addi x6,x0,1024

addi x2,x2,18

add x2,x3,x4

sw x2,0(x6)

lw x7,4(x6)

add x1,x1,x7

add x3,x5,x7

sub x3,x3,x7

lw x4,8(x6)

add x4,x4, x4

add x4,x6,x7

填写表2.1回答：

1）哪几条标号指令存在数据冲突（hazard）？

2）哪几条标号指令数据冲突不能通过旁路技术（forward）解决？为什么？

3）如果流水线时钟周期为100ns，计算完成这些指令所需时间。

**一共需要17个时钟周期，所以需要1700ns。**

表2.1

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 程序段 | 是否有冲突 | 是否可通过旁路解决冲突，为什么 |
| 1 | addi x6,x0,1024 | 无 | / |
| 2 | addi x2,x2,18 | 无 | / |
| 3 | add x2,x3,x4 | 无 | / |
| 4 | sw x2,0(x6) | 有 | 可以，上一条指令执行完exe时该指令进入exe，可以得到上一条指令前递的数据。 |
| 5 | lw x7,4(x6) | 无 | / |
| 6 | add x1,x1,x7 | 有 | 不可以，lw和add指令的相关性在时间上是逆向的。 |
| 7 | add x3,x5,x7 | 无 | / |
| 8 | sub x3,x3,x7 | 有 | 可以，上一条指令执行完exe时该指令进入exe，可以得到上一条指令前递的数据。 |
| 9 | lw x4,8(x6) | 无 | / |
| 10 | add x4,x4, x4 | 有 | 不可以，lw和add指令的相关性在时间上是逆向的。 |
| 11 | add x4,x6,x7 | 无 | / |

4）分析程序段，以表2.2为模板画出你认为可能的五段流水线执行表表2.2-1。并在ripes平台上完成仿真，以表2.2为模板画出实际仿真过程的执行表标为表2.2-2，再通过webrisc-v平台进行仿真，截图得到仿真过程的执行表标为表2.2-3，做对比分析。和你认为的执行过程是否一致？对比两个平台上的仿真执行表的异同，理解同样支持RISCV指令而以不同流水线硬件结构实现时其实际执行过程的区别。

**同：花费的时钟周期总数均为17，且均发生了stall。两者的lw指令的前递均发生在MEM/WB寄存器**

**异：，****webrisc-v的lw会使下一条指令stall在ID阶段，而ripes的stall指令会使下一条指令stall在EX阶段。这说明webrisc-v指令在ID 阶段就已经用到了前递的寄存器值，而ripes指令则是先读取当前寄存器的值，在EXE阶段再与前递来的值比较进行计算。**

表2.2-1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction\clock cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| addi x6,x0,1024 | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |
| addi x2,x2,18 |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |
| add x2,x3,x4 |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |
| sw x2,0(x6) |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |
| lw x7,4(x6) |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |
| add x1,x1,x7 |  |  |  |  |  | F | - | D | E | M | W |  |  |  |  |  |  |
| add x3,x5,x7 |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |
| sub x3,x3,x7 |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |
| lw x4,8(x6) |  |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |
| add x4,x4,x4 |  |  |  |  |  |  |  |  |  |  | F | - | D | E | M | W |  |
| add x4,x6,x7 |  |  |  |  |  |  |  |  |  |  |  |  | F | D | E | M | W |

表2.2-2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction\clock cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| addi x6,x0,1024 | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |
| addi x2,x2,18 |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |
| add x2,x3,x4 |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |
| sw x2,0(x6) |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |
| lw x7,4(x6) |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |
| add x1,x1,x7 |  |  |  |  |  | F | D | - | E | M | W |  |  |  |  |  |  |
| add x3,x5,x7 |  |  |  |  |  |  | F | - | D | E | M | W |  |  |  |  |  |
| sub x3,x3,x7 |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |
| lw x4,8(x6) |  |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |
| add x4,x4,x4 |  |  |  |  |  |  |  |  |  |  | F | D | - | E | M | W |  |
| add x4,x6,x7 |  |  |  |  |  |  |  |  |  |  |  | F | - | D | E | M | W |

表2.2-3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction\clock cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| addi x6,x0,1024 | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |
| addi x2,x2,18 |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |
| add x2,x3,x4 |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |
| sw x2,0(x6) |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |
| lw x7,4(x6) |  |  |  |  | F | D | E | M | W |  |  |  |  |  |  |  |  |
| add x1,x1,x7 |  |  |  |  |  | F | - | D | E | M | W |  |  |  |  |  |  |
| add x3,x5,x7 |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |  |
| sub x3,x3,x7 |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |  |
| lw x4,8(x6) |  |  |  |  |  |  |  |  |  | F | D | E | M | W |  |  |  |
| add x4,x4,x4 |  |  |  |  |  |  |  |  |  |  | F | - | D | E | M | W |  |
| add x4,x6,x7 |  |  |  |  |  |  |  |  |  |  |  |  | F | D | E | M | W |

#### 3、附录 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 。

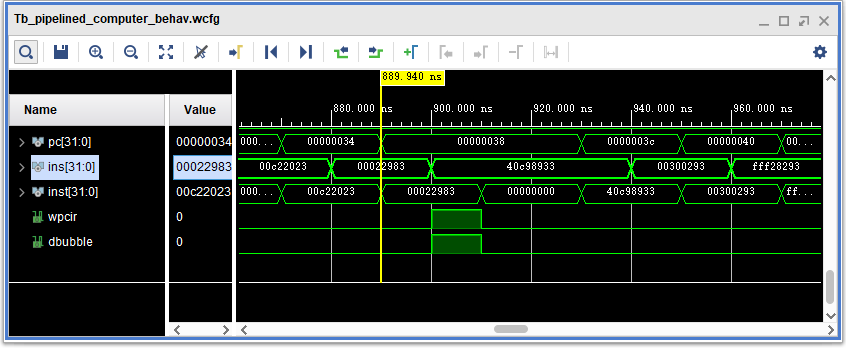


图1 pc出现持续2个时钟的bubble

1. **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。

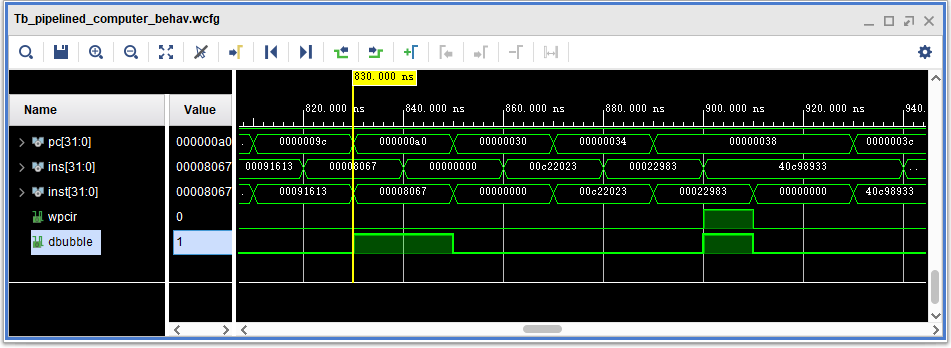


图2从指令寄存器读出的 ins 被flush操作设置为0

1. **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 紫色框中的多路选择器。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。

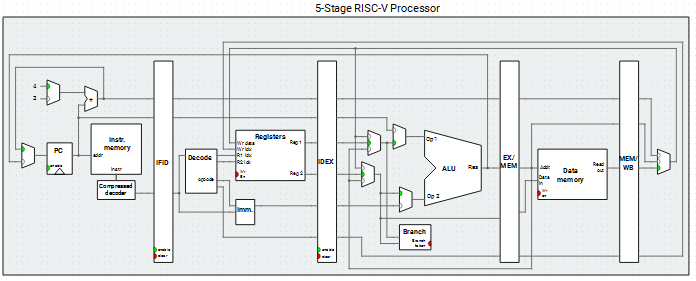


图3 一种五级流水线结构中的forward功能模块示意

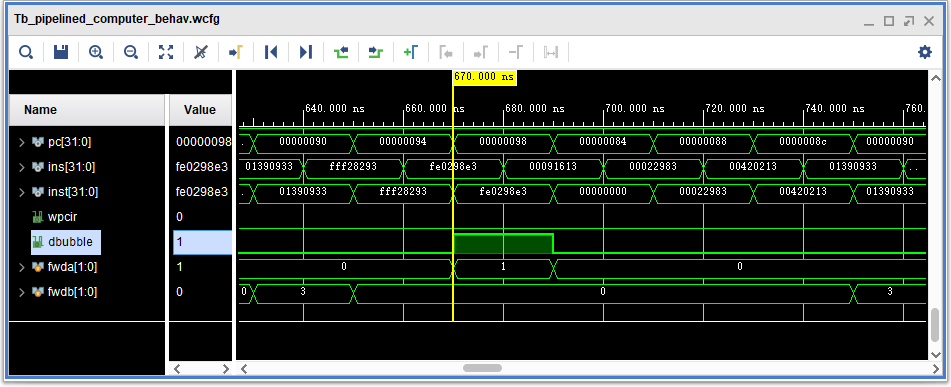


图4 另一种五级流水线结构中的forward功能实现中的控制信号示意