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。

#### 2、课堂练习

2.1代码段1的练习

1. **关于不同硬件结构对控制冒险处理方式的理解**

在 RISC V仿真平台<https://webriscv.dii.unisi.it/index.php>，选择 RV32IM，Forwarding activate，branch hazard handling Flush Instruction模式，输入以下代码段，执行单步仿真，截图execution table。

sw x29,1024(x16)

lw x29,1024(x16)

sub x17,x15,x14

beq x17,x0,label

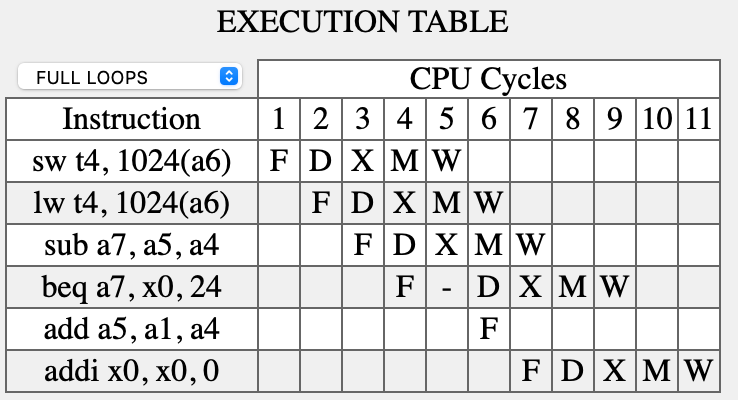
add x15,x11,x14

sub x15,x30,x14

label:nop

从截图可以看出倒数第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？ （10分）

答：



1. beq x17, x0, label指令被执行了forward和stall处理，因为上一句指令sub x17,x15,x14对x17进行了修改，发生数据冒险，执行了前递;同时beq指令stall了一个时钟周期来等待sub x17, x15, x14进行EX阶段产生ALU输出。
2. add x15, x11, x14指令被执行flush处理，因为beq a7, x0, 24判断应该采取分支，分支预测不正确，该指令错误进入IF阶段，发生控制冒险，需要被flush。

再在Ripes仿真平台输入这段代码，选择 32bit 5-stage processor模式，也即含有Forwarding和branch hazard detection/elimination（通过flush指令方式）模式，执行单步仿真，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。分析倒数第3和2句（add x15,x11,x14和 sub x15,x30,x14）被执行了什么特殊处理？为什么？（10分）

表1 5-stage processor模式

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction\clock cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| sw x29,1024(x16) | F | D | X | M | W |  |  |  |  |  |  |
| lw 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 |

答：

add x15, x11, 14和sub x15, x30, x14被flush,因为分支预测不正确，它们进入了ID、IF阶段但不应被执行，发生了控制冒险，需要被flush来解决控制冒险。

与前一个仿真平台的仿真执行表截图有什么不同？试着分析一下原因。理解同样支持RISCV指令而采用不同流水线硬件结构实现的CPU的实际执行过程的区别。 （10分）

答：

不同：beq指令没有stall一个时钟周期，且后两条指令被flush。原因：Ripes的实现中在EX阶段才进行分支判断，而Web仿真在ID阶段进行分支判断。所以Ripes的beq指令不需要stall一周期，但需要flush掉进入IF/ID阶段发生控制冒险的两条指令；Web仿真需要stall一周期等待上一条指令的ALU输出稳定，再在ID阶段判断并选择是否flush掉IF阶段的指令。两者实现稍有差别但时钟周期数相同。

**2、关于数据冒险的处理，有无forwarding处理的执行过程对比**

选择 32bit 5-stage processor w/o forwarding unit模式，也即不含有forwarding，只含有branch hazard detection/elimination模式

做一次五级流水线CPU单步仿真，参照表2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。分析beq x17,x0,label 语句及其后的2条语句在哪些时钟周期被执行了什么特殊处理？为什么？ （10分）

表2 5-stage processor w/o forwarding unit模式

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction\clock cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| sw x29,1024(x16) | F | D | X | M | W |  |  |  |  |  |  |  |  |
| lw 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 |

答：

beq语句stall了两个周期，因为beq使用了上一条sub语句中被修改的x17，发生了数据冒险，因为没有forwarding模块，必须等待上一条指令经过WB阶段将x17的值写回寄存器才能进入EX阶段进行分支判断，因此要stall两个时钟周期；而后两条语句被flush，因为在EX阶段判断分支应该采取，分支预测错误，发生了控制冒险，需要将两条进入IF,ID阶段但不应被执行的指令flush.

**3、关于对无控制冒险检测的硬件可能执行结果不正确的理解**

选择 32bit 5-stage processor 和5-stage processor w/o hazard detection模式，也即均含有Forwarding但是含或者不含有branch hazard detection/elimination（通过flush指令方式）模式

**在首句前添加addi x15,x15,1**，分别做一次单步仿真，查看执行后x15的值在两种模式下是否一样，分析原因。 （10分）

答：

不同。without hazard detection模式中x15的值为1，with hazard detection模式中x15的值为0。原因：因为without hazard detection中执行beq时没有检测到beq对sub指令修改的x17有数据依赖，存在数据冒险，所以没有发生前递，beq指令执行时读取的是RegFile中尚未更新的x17值即0，导致分支判断认为需要采取分支，后两条指令被flush，没有对x15进行修改，x15最终值为1；而有hazard detection时检测到数据冒险，将sub中x17的值前递，分支判断正确，认为不应该采取分支，没有进行flush，最终add x15, x11, x14、sub x15, x30, x14被执行，x15最终值为0.

注意：Ripes平台的without hazard detection应该指没有数据冒险检测，因为观察到流水线执行表中它仍然能执行flush操作。对于Web平台，因为这里的beq指令只要有forwarding就会被判断不采取分支，所以两者执行结果没有区别。

参照表3、4前三列的前三个时钟执行过程画出两个模式下完整五级流水线（F、D、X、M、W）执行表。解释执行过程为什么不同，哪种情况执行结果是正确的。执行不正确的那种模式下，如果要得到正确结果，可以怎么修改程序？（10分）

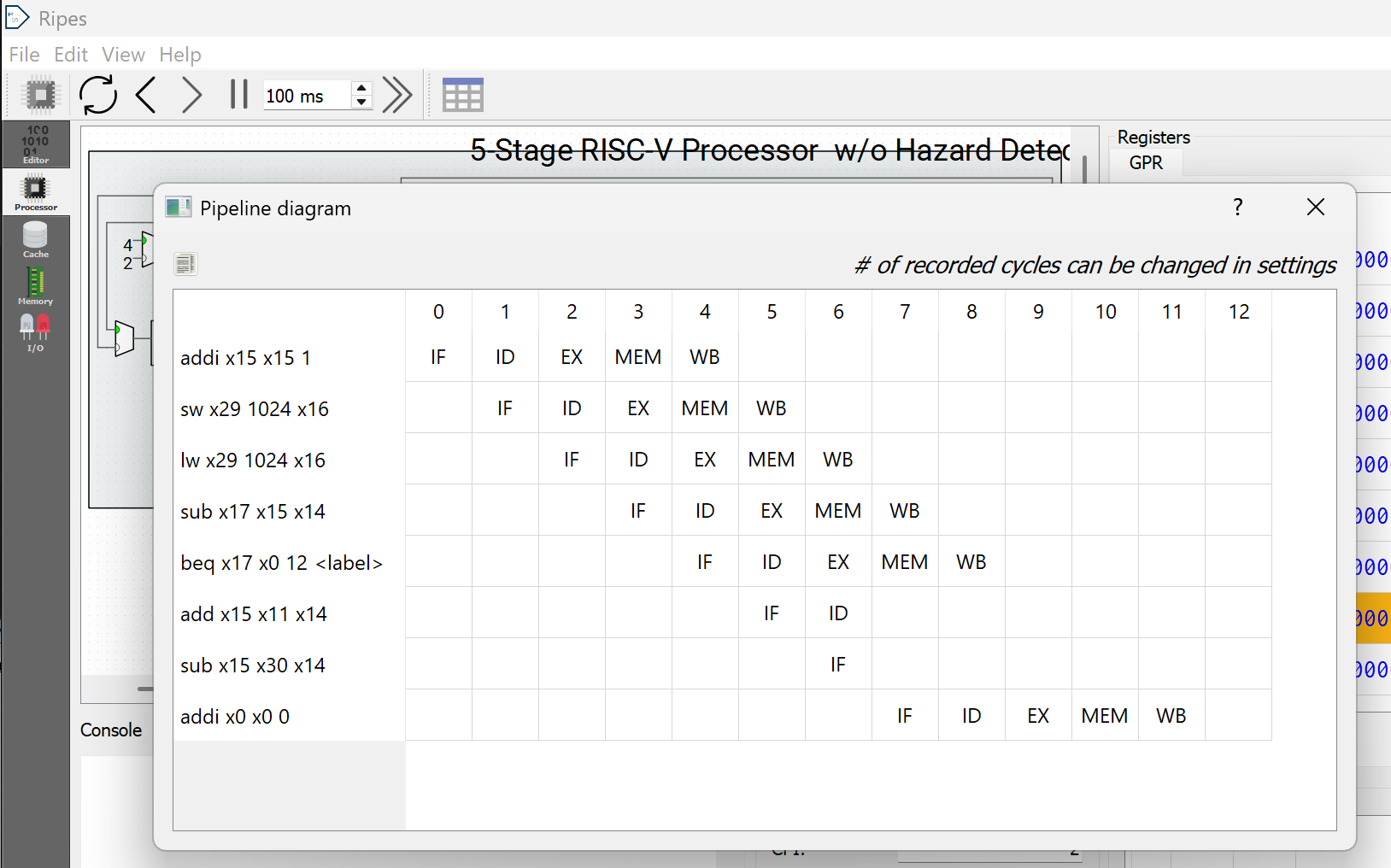


表3 5-stage processor w/o hazard detection模式

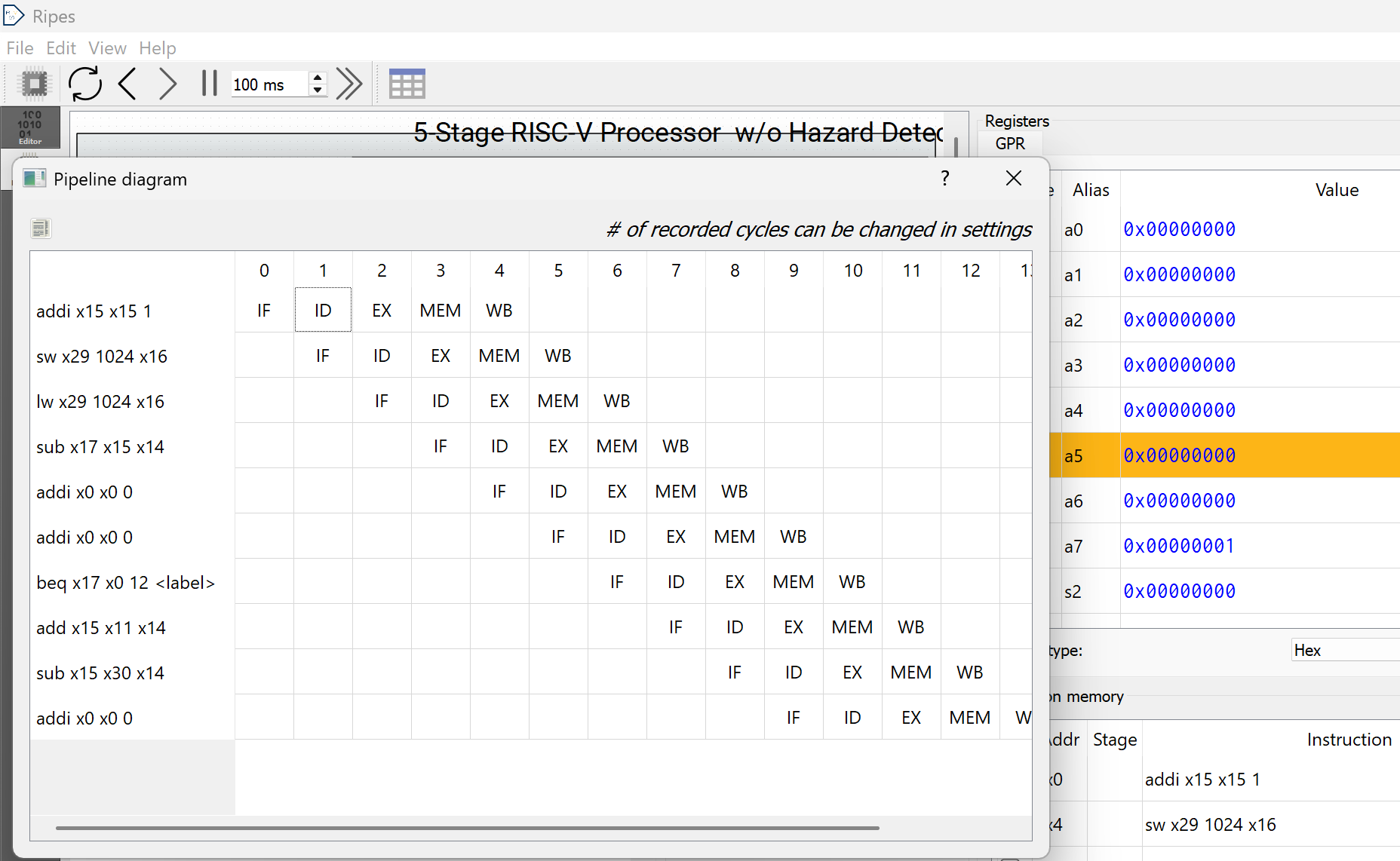
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction\clock cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| addi x15,x15,1 | F | D | X | M | W |  |  |  |  |  |  |  |
| sw x29,1024(x16) |  | F | D | X | M | W |  |  |  |  |  |  |
| lw 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 |

表4 5-stage processor模式

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction\clock cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| addi x15,x15,1 | F | D | X | M | W |  |  |  |  |  |  |  |
| sw x29,1024(x16) |  | F | D | X | M | W |  |  |  |  |  |  |
| lw 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 | X | M | W |  |  |
| sub x15,x30,x14 |  |  |  |  |  |  | F | D | X | M | W |  |
| label:nop |  |  |  |  |  |  |  | F | D | X | M | W |

答：

表4的执行结果是正确的。表3因为没有进行数据冒险检测，所以没有正确执行前递，导致分支判断错误。要使结果正确可以在beq指令前面加入2条nop指令，强制beq在sub指令完成对x17值的写回(通过WB阶段)以后再进入EX阶段。如下图



2.2 代码段二的练习

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）？（10分）

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？（10分）

3）如果流水线时钟周期为100ns，计算完成这些指令所需时间。（10分）

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 | 否 |  |
| 2 | addi x2,x2,18 | 否 |  |
| 3 | add x2,x3,x4 | 是 | 可以 |
| 4 | sw x2,0(x6) | 是 | 可以 |
| 5 | lw x7,4(x6) | 否 |  |
| 6 | add x1,x1,x7 | 是 | 需要停顿一周期，因为要等待M阶段读取 |
| 7 | add x3,x5,x7 | 否 |  |
| 8 | sub x3,x3,x7 | 是 | 可以 |
| 9 | lw x4,8(x6) | 否 |  |
| 10 | add x4,x4, x4 | 是 | 需要停顿一周期，因为要等待M阶段读取 |
| 11 | add x4,x6,x7 | 是 | 可以 |

答：

1. 17 \* 100ns = 1700ns

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。和你认为的执行过程是否一致？（10分）

表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 |

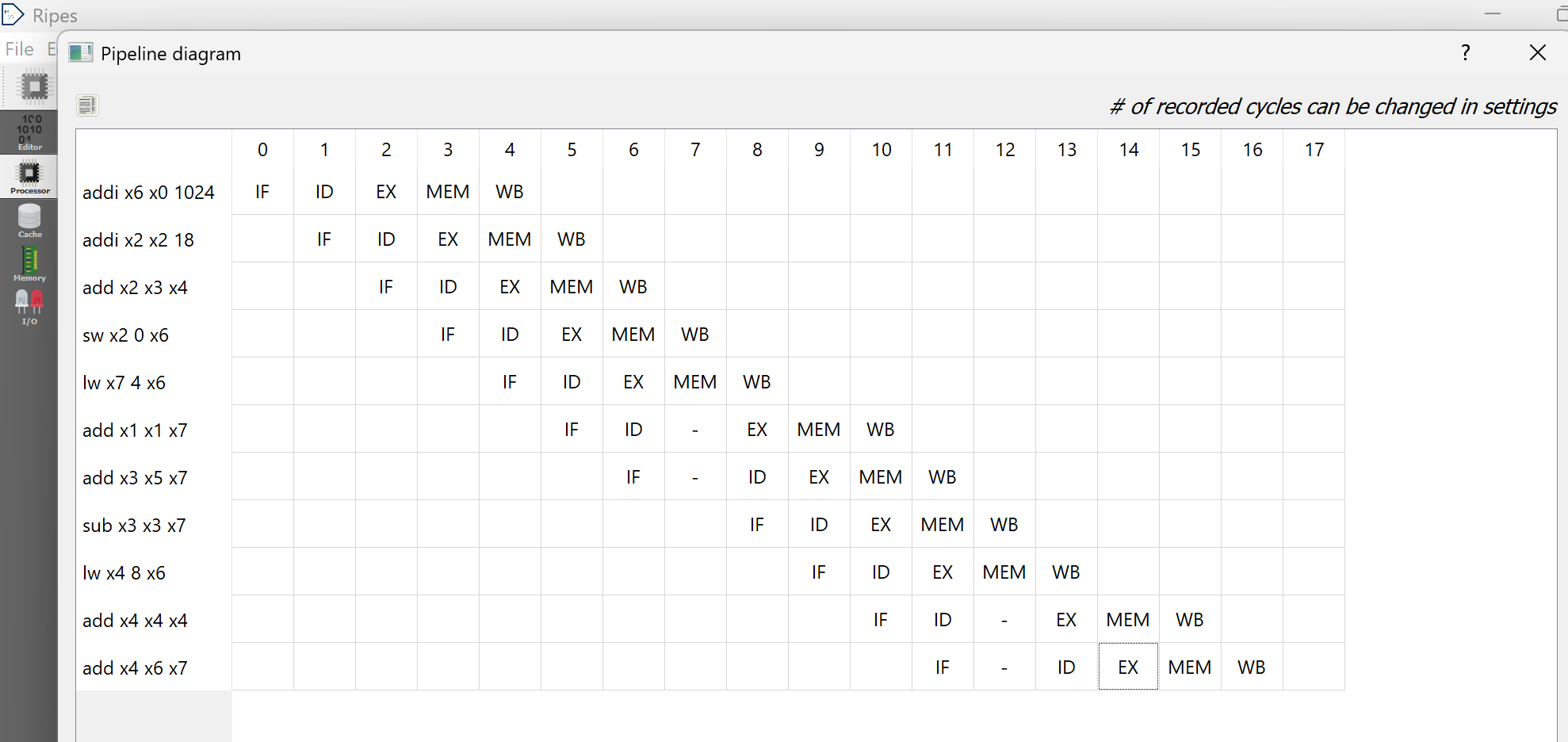


表2.2-2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

答：

一致。

#### 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 。

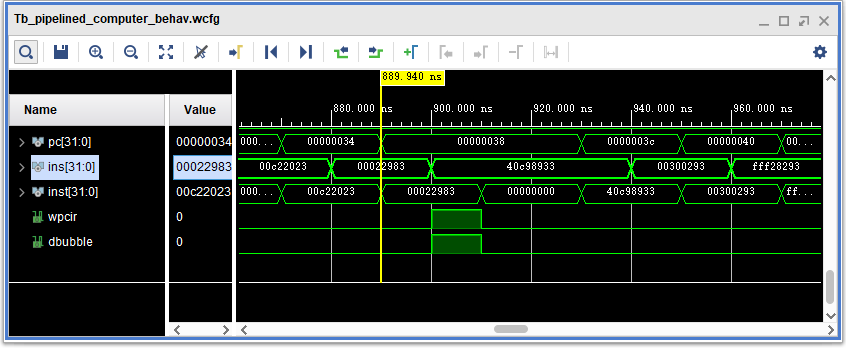


图1 pc出现持续2个时钟的bubble

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。

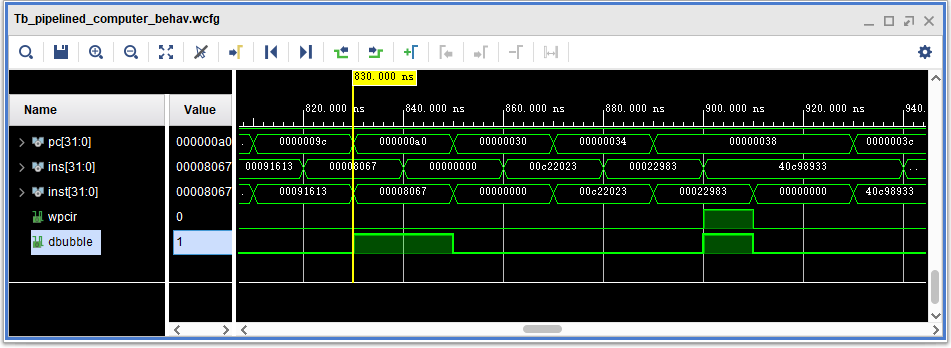


图2从指令寄存器读出的 ins 被flush操作设置为0

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。

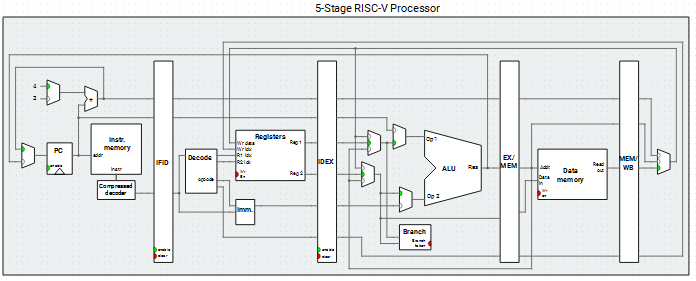


图3 一种五级流水线结构中的forward功能模块示意

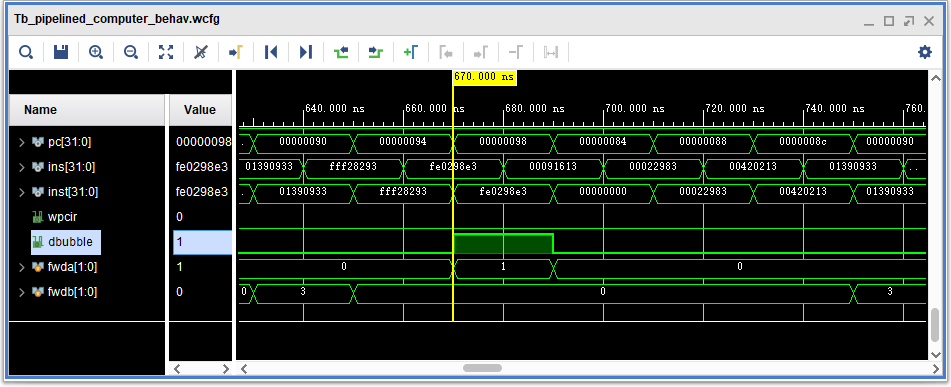


图4 另一种五级流水线结构中的forward功能实现中的控制信号示意