计算机组成原理 P5实验报告

## 一、CPU设计方案综述

本CPU为Verilog实现的流水线MIPS32 - CPU，支持的指令集包含MIPS-lite2 **{addu, subu, ori, lw, sw, beq, lui, j, jal, jr, nop}**。,采用模块化和层次化设计。顶层有效的驱动信号仅有内置的clk时钟信号和同步复位reset信号。

模块定义

二．模块接口

1.PC\_IM

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时间信号 |
| reset | I | 复位信号 |
| WE | I | 使能 |
| npc | I | 下一周期待执行指令的32位地址 |
| pc | O | 本周起待执行指令的32位地址 |
| Instr | O | 当前执行的32位机器码指令 |

2. REG\_D (IF-ID寄存器)

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| REG\_D.v | module REG\_D(  input clk,  input reset,  input en,  input [31:0] Instr,  input [31:0] pc,  output reg[31:0] Instr\_out,  output reg[31:0] pc\_out ); |

3.GRF

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| pc | I | 当前执行的机器码指令 |
| clk | I | 时钟信号 |
| reset | I | 异步复位信号，将32个寄存器中的值全部清零  1：复位 0：无效 |
| WE | I | 写使能信号  1：可向GRF中写入数据 0：不可向GRF中写入数据 |
| a1 | I | 5位地址输入信号，指定32个寄存器中的一个，将其中存储的数据读出到RD1 |
| a2 | I | 5位地址输入信号，指定32个寄存器中的一个，将其中存储的数据读出到RD2 |
| a3 | I | 5位地址输入信号，指定32个寄存器中的一个，作为写入的目标寄存器 |
| WD | I | 32位数据输入信号 |
| RD1 | O | 输出a1指定的寄存器中的32位数据 |
| RD2 | O | 输出a2指定的寄存器中的32位数据 |

4.CMP

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| rsRD1[31:0] | I | Rs指向寄存器的值 |
| rtRD2[31:0] | I | Rt指向寄存器的值 |
| B\_type[2:0] | I | B跳转指令的类型 |
| B\_jump | O | 是否执行B跳转指令 |

5. EXT

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| EXTin | I | J型指令25-0位中存储的26位立即数 |
| EXTop | O | 1位控制信号，控制EXT对16位立即数的位扩展方式  0：零扩展 1：符号扩展 |
| EXTout | O | 32位的立即数的扩展结果 |

6. NPC

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| D\_PC | I | D级PC |
| F\_PC | I | F级PC |
| I26 | I | 26位立即数 |
| MFRSD | I | 转发PC的MUX结果（jr jalr需要转发） |
| Br | I | 选择下一条指令地址的类型 |
| B\_jump | I | 判断BRANCH类指令条件是否成立 |
| Npc | O | 更新的pc值 |

7.E\_REG(ID/EX)

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| E\_REG.v | module E\_REG (  input clk,  input reset,  input en,  input [31:0] Instr,  input [31:0] PC,  input [31:0] RF\_RD1,  input [31:0] RF\_RD2,  input [31:0] EXT,  output reg[31:0] Instr\_out,  output reg[31:0] PC\_out,  output reg[31:0] RS\_E,  output reg[31:0] RT\_E,  output reg[31:0] EXT\_E  ); |

8.ALU

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| A[31:0] | I | 32 位输入数据 1 |
| B[31:0] | I | 32 位输入数据 2 |
| ALUCtrl[2:0] | I | 控制信号  000：与  001：或  010：加  011：减  100：移位 |
| Shamt[4:0] | I | 位移量 |
| ALUout | O | ALU计算结果 |

9.M\_REG（EX/MEM）

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| M\_REG.v | module M\_REG (  input clk,  input reset,  input en,  input [31:0] Instr,  input [31:0] PC,  input [31:0] ALU,  input [31:0] RT\_E,  output reg[31:0] Instr\_out,  output reg[31:0] PC\_out,  output reg[31:0] ALU\_out,  output reg[31:0] RT\_M  ); |

10.DM

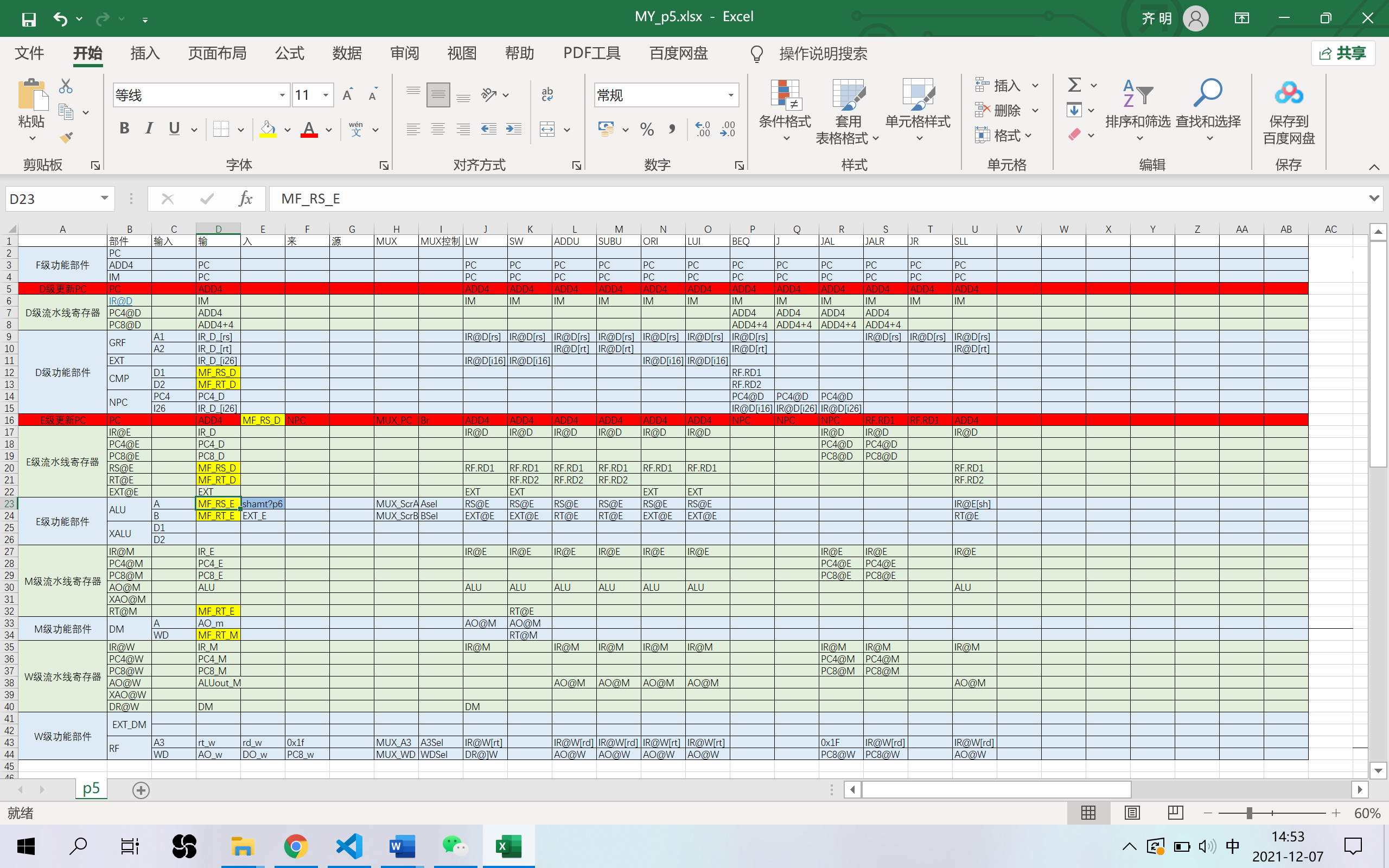
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| Addr[31:0] | I | ALUout,写入地址 |
| Bit\_type[2:0] | I | Store/load的类型 |
| Newdata | I | RD2 |
| PC[31:0] | I | 当前PC |
| WE | I | 使能 |
| Reset | I | 复位信号  1：复位  0：无效 |
| Clk | I | 时钟信号 |
| Out\_DM[31:0] | O | 加载内存的结果 |

11.W\_REG(MEM/WB)

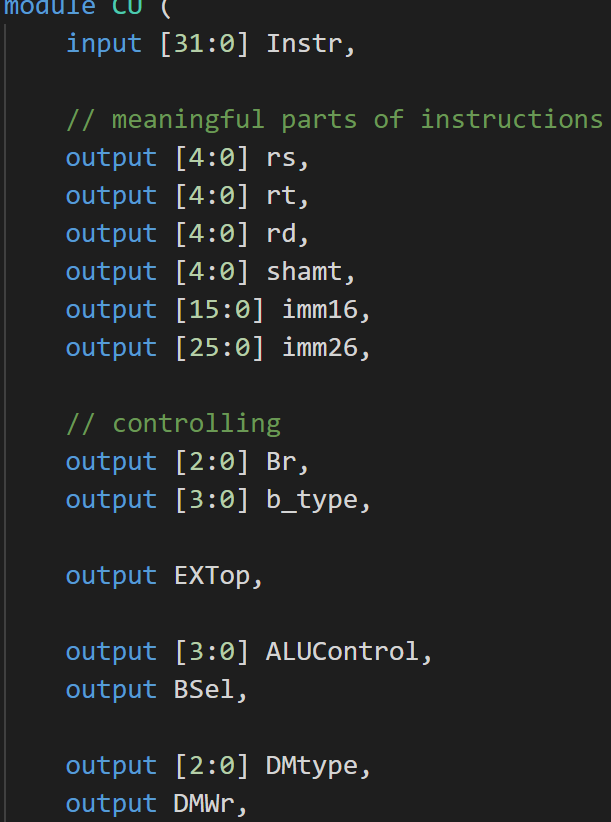
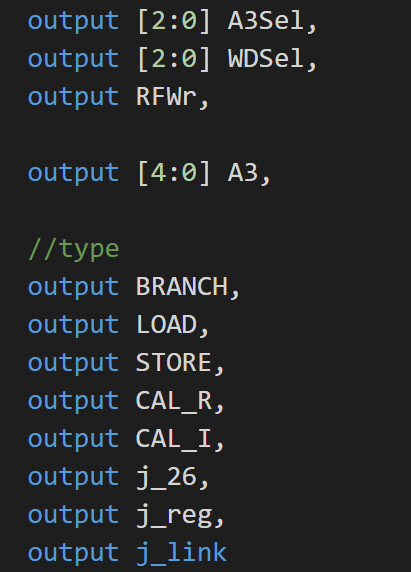
|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| W\_REG | module W\_REG (  input clk,  input reset,  input en,  input [31:0] Instr,  input [31:0] PC,  input [31:0] AO\_M,  input [31:0] DM,  output reg[31:0] Instr\_out,  output reg[31:0] PC\_W,  output reg[31:0] AO\_W,  output reg[31:0] DR\_W  ); |

三．控制器设计(CU)

含有转发数据的数据通路如表格



CU （采取分布式译码）

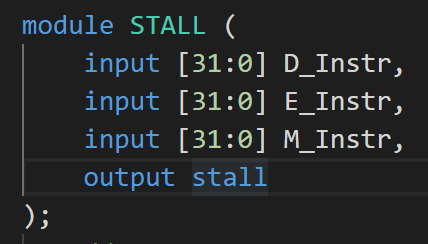
 

具体控制信号的生成，采取先对指令进行指令类型判断，再根据指令类型得到控制型号取值的方法。具体实现见CU.v。

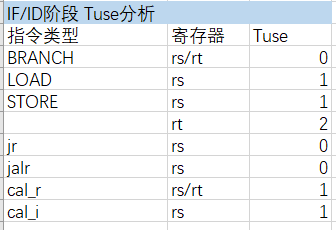
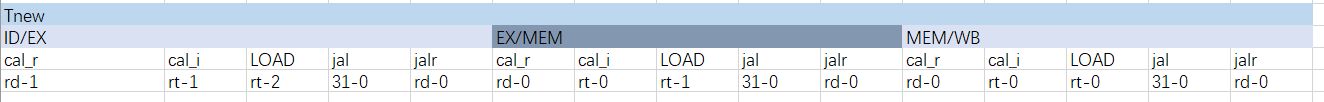
在流水线阶段，根据使用需要，将CU实例化四次。

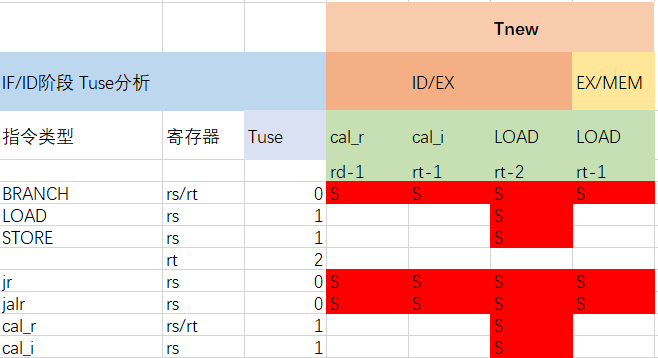
四．冒险处理单元设计(FWD,STALL)———使用**需求时间——供给时间模型**。

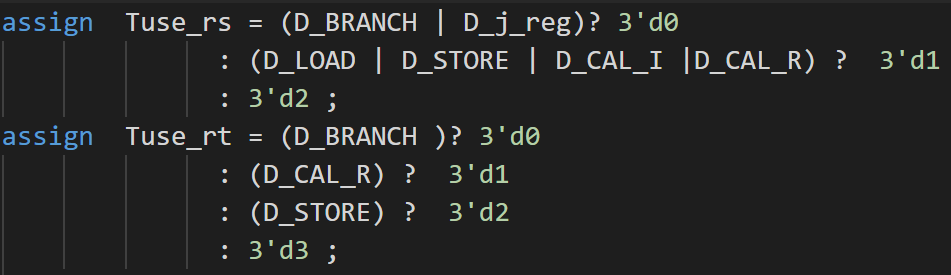
1．Stall 暂停信号的生成

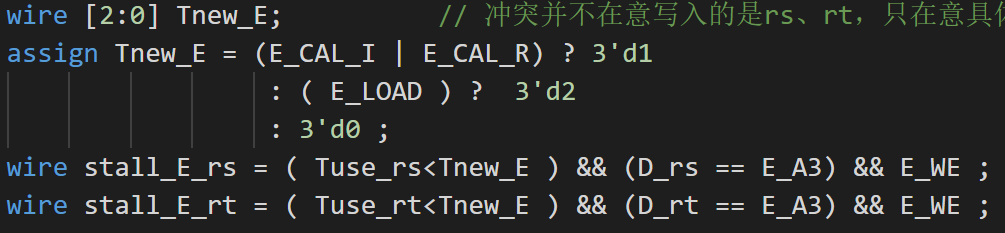


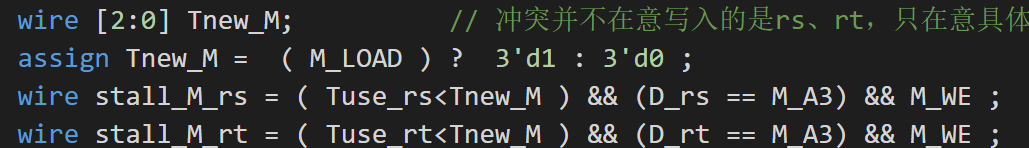
通过D、E、M级流水线寄存器的指令类型来判断stall信号，从而完成流水线的暂停（即寄存器使能置零和值清空）





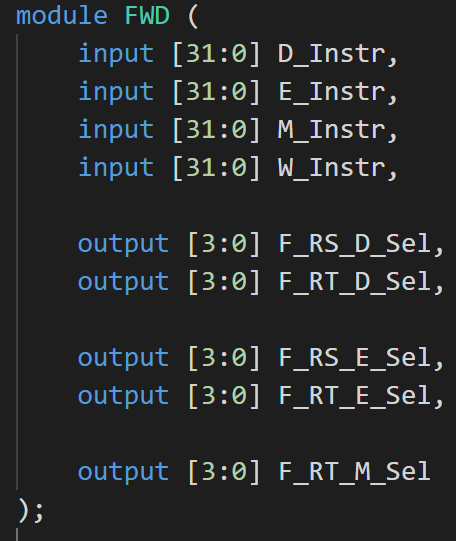




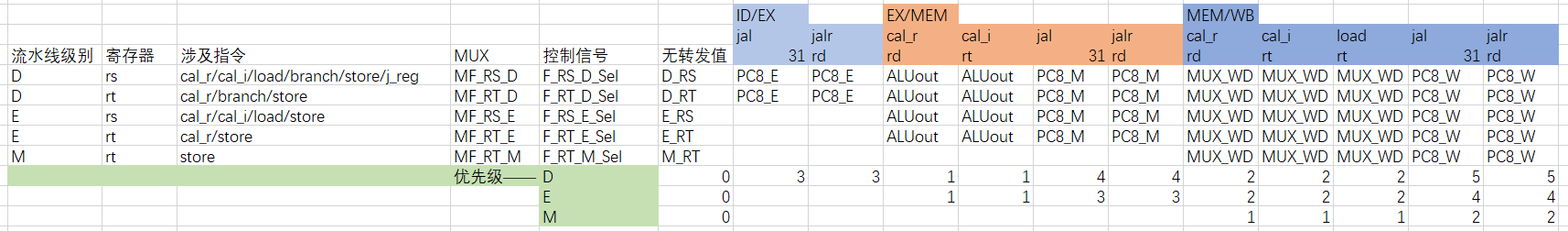
assign stall = stall\_E\_rs | stall\_E\_rt | stall\_M\_rs | stall\_M\_rt ;

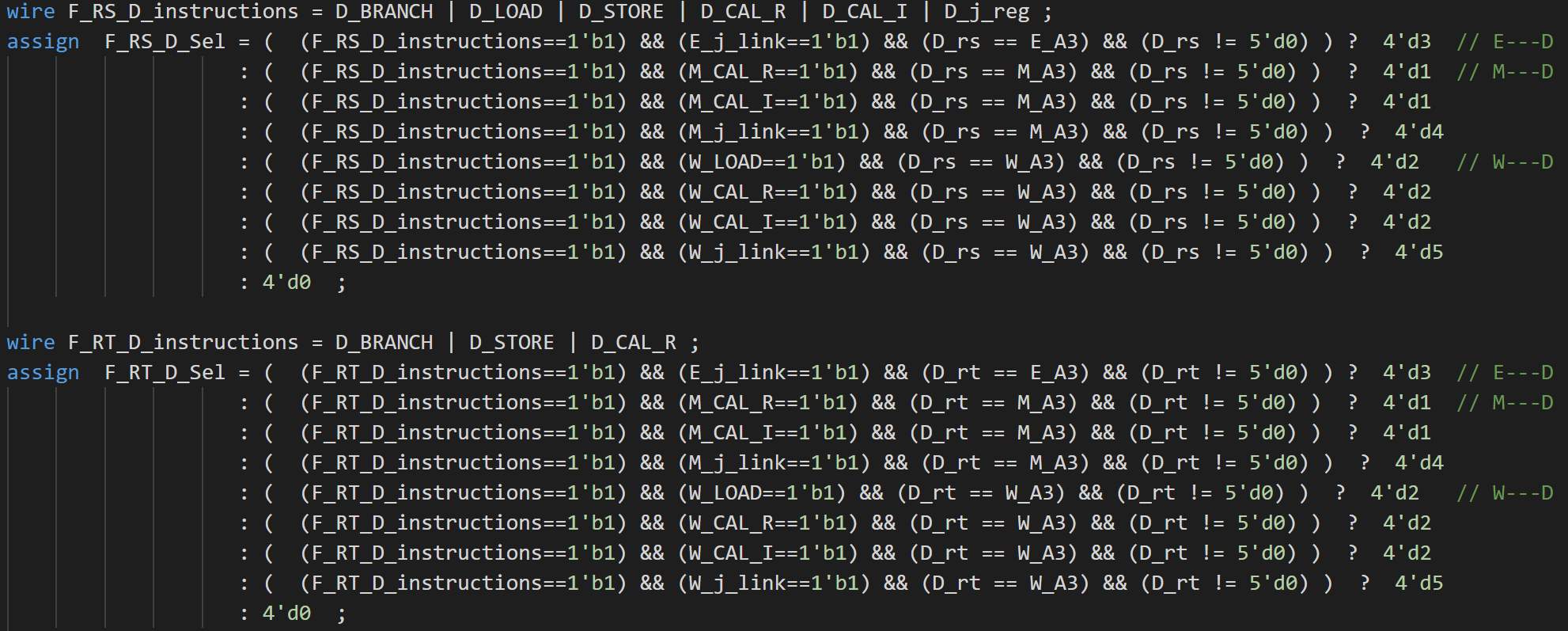
总的暂停信号通过各级rs、rt处发生的暂停信号进行按位与后得到。

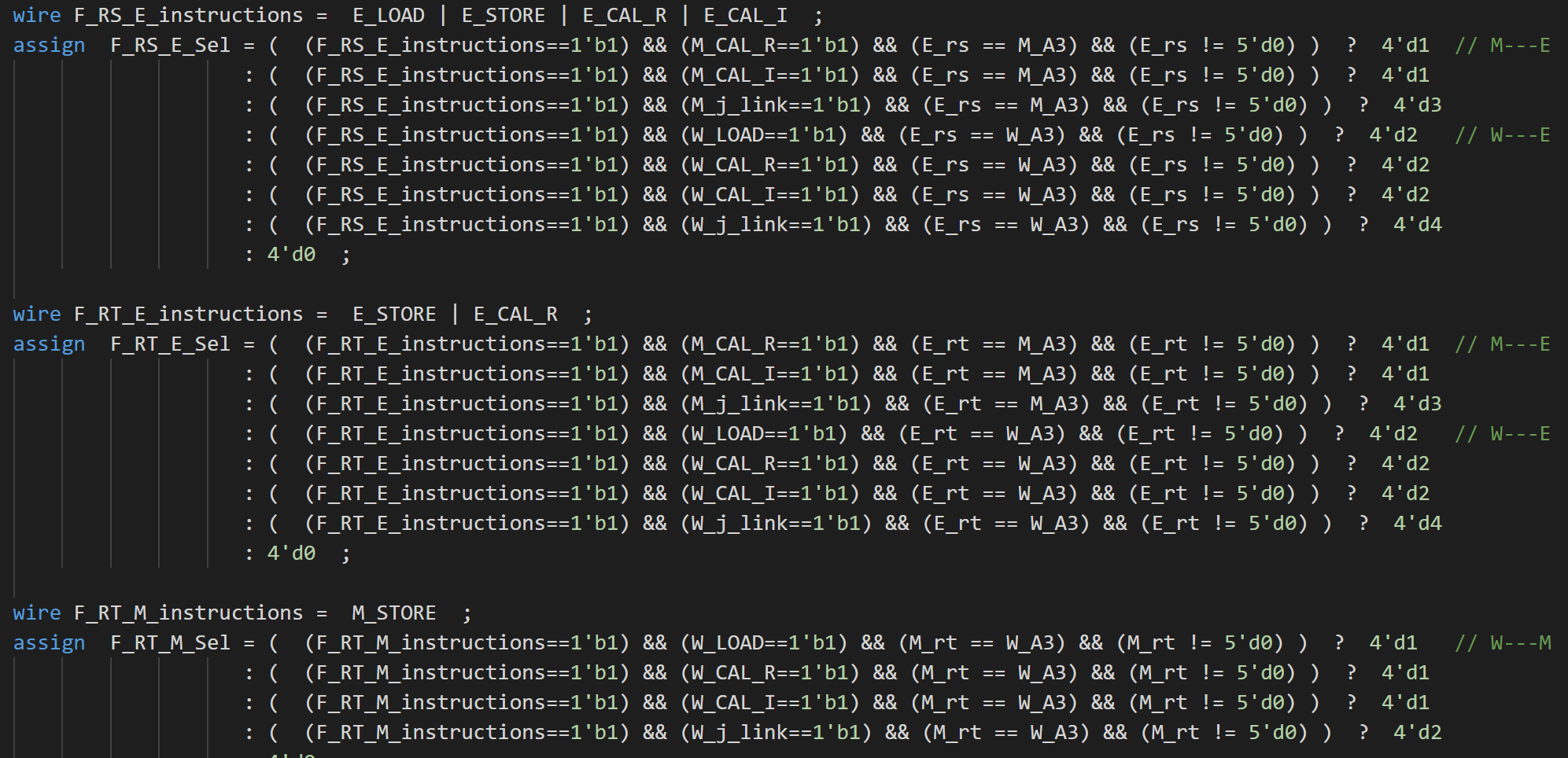
2．转发MUX的选择信号的生成



FWD.v用于生成转发MUX的选择信号，在mips.v中实例化后使用



****



五．测试程序

(1)转发机制覆盖测试

一．D级rs

1.D级rs与E级j\_link(jal/jalr)

2.D级rs与M级cal\_r

3.D级rs与M级cal\_i

4.D级rs与M级jal

5.D级rs与W级cal\_r

6.D级rs与W级cal\_i

7.D级rs与W级load rt

8.D级rs与W级jal

每一项E级rs指令又分为：cal\_r,cal\_i,load,store,beq（branch）,jr,jalr

二．D级Rt

1.D级rt与E级j\_link(jal/jalr)

2.D级rt与M级cal\_r

3.D级rt与M级cal\_i

4.D级rt与M级jal

5.D级rt与W级cal\_r

6.D级rt与W级cal\_i

7.D级rt与W级load rt

8.D级rt与W级jal

每一项又分为：cal\_r,st,beq

三．E级Rs

1.E级Rs与M级cal\_r

2.E级Rs与M级cal\_i

3.E级Rs与M级jal

4.E级Rs与W级cal\_r

5.E级Rs与W级cal\_i

6.E级Rs与W级load

7.E级Rs与W级jal

每一项又分为：cal\_r,cal\_i,load,store

四．E级Rt

1.E级Rt与M级cal\_r

2.E级Rt与M级cal\_i

3.E级Rt与M级jal

4.E级Rt与W级cal\_r

5.E级Rt与W级cal\_i

6.E级Rt与W级load

7.E级Rt与W级jal

每一项又分为：cal\_r,store

五．M级Rt

1.M级Rt与W级cal\_r

2.M级Rt与W级cal\_i

3.M级Rt与W级load

4.M级Rt与W级jal

每一项又分为：store

(2)暂停机制覆盖测试(按照指令来取的覆盖性测试)

测试目录：

一．Beq\_rs/rt

(1) E级cal\_r\_rd

(2) E级cal\_i\_rt

(3) E级load\_rt

(4) M级load\_rt

二．Cal\_r\_rs/rt

E级load\_rt

三．Cal\_i\_rs

E级load\_rt

四．load\_rs

E级load\_rt

五．store\_rs

E级load\_rt

六．jr\_rs

(1)E级cal\_r\_rd

(2) E级cal\_i\_rt

(3) E级load\_rt

(4) M级load\_rt

测试程序（Mars与CPU的输出对拍对比直接使用python正则表达式匹配程序实现）

ori $1, $1, 1

nop

nop

nop

nop

addu $2, $1, $1

addu $2, $1, $2

subu $2, $2, $1

subu $2, $1, $2

addu $2, $1, $2

nop

nop

nop

nop

lui $1, 0

lui $2, 0

nop

nop

ori $1, $1, 1

nop

nop

nop

nop

addu $2, $1, $1

nop

addu $2, $1, $2

nop

subu $2, $2, $1

nop

subu $2, $1, $2

nop

addu $2, $1, $2

nop

nop

nop

nop

lui $1, 0

lui $2, 0

nop

nop

nop

ori $1, $1, 1

nop

nop

nop

nop

addu $2, $1, $1

nop

nop

addu $2, $1, $2

nop

nop

subu $2, $2, $1

nop

nop

subu $2, $1, $2

nop

nop

addu $2, $1, $2

nop

nop

nop

nop

ori $1, 123

addu $2, $1, $1

subu $3, $1, $1

nop

nop

nop

nop

ori $1, 321

subu $2, $1, $1

addu $3, $1, $1

nop

nop

nop

ori $1, 4321

nop

addu $2, $1, $1

subu $3, $1, $1

nop

nop

nop

nop

ori $1, 54212

nop

subu $2, $1, $1

addu $3, $1, $1

nop

nop

nop

nop

lui $1, 123

addu $2, $1, $1

subu $3, $2, $1

nop

nop

nop

nop

lui $1, 223

subu $3, $2, $1

addu $2, $1, $1

nop

nop

nop

nop

lui $1, 321

nop

addu $2, $1, $2

subu $3, $1, $1

nop

nop

nop

nop

lui $1, 432

nop

subu $3, $1, $1

addu $2, $1, $3

nop

nop

nop

jal label00

addu $2, $1, $31

label00: subu $1, $31, $1

jal label01

subu $1, $31, $1

label01: addu $2, $1, $31

jal label02

nop

label02: addu $2, $1, $31

subu $1, $31, $1

ori $2, $1, 123

ori $3, $1, 432

ori $4, $1, 543

jal label03

nop

label03: subu $1, $31, $1

addu $2, $1, $31

ori $3, $2, 123

ori $4, $2, 234

ori $5, $2, 345

ori $1, $5, 12345

ori $2, $5, 43121

ori $3, $5, 12313

ori $4, $5, 12421

lui $1, 12

ori $2, $1, 123

ori $2, $1, 121

ori $2, $1, 122

ori $2, $1, 143

jal label04

ori $1, $31, 123

label04 : ori $1, $31, 124

ori $1, $31, 125

ori $1, $31, 126

nop

nop

nop

addu $2, $1, $31

beq $2, $1, label05

nop

label05:

addu $3, $1, $31

nop

beq $3, $1, label06

nop

label06:

addu $4, $1, $31

nop

nop

beq $4, $1, label07

nop

label07:

subu $2, $1, $31

beq $2, $1, label08

nop

label08:

subu $3, $1, $31

nop

beq $3, $1, label09

nop

label09:

subu $4, $1, $31

nop

nop

beq $4, $1, label10

nop

label10:

ori $1, $4, 12312

beq $1, $31, label11

ori $2, $4, 12312

label11:nop

beq $2, $31, label12

ori $3, $4, 12312

label12: nop

nop

beq $3, $31, label13

ori $4, $3, 43121

label13: nop

nop

nop

beq $4, $31, label14

nop

label14:nop

lui $1, 12312

beq $1, $31, label15

lui $2, 12312

label15:nop

beq $2, $31, label16

lui $3, 12312

label16: nop

nop

beq $3, $31, label17

lui $4, 43121

label17: nop

nop

nop

beq $4, $31, label18

nop

label18:nop

jal label19

nop

label19: beq $31, $1, label20

nop

label20: jal label21

nop

label21: nop

beq $31, $1, label22

nop

label22: jal label23

nop

label23: nop

nop

beq $31, $1, label24

nop

label24:

ori $2, $0, 16

addu $1, $31, $31

sw $1, -16($2)

sw $1, -12($2)

sw $1, -8($2)

sw $1, 12($0)

subu $1, $1, $31

sw $1, 0($0)

sw $1, 4($0)

sw $1, 8($0)

sw $1, 12($0)

ori $1, $1, 5324

sw $1, 0($0)

sw $1, 4($0)

sw $1, 8($0)

sw $1, 12($0)

lui $1, 12

sw $1, 0($0)

sw $1, 4($0)

sw $1, 8($0)

sw $1, 12($0)

jal label25

sw $31, -16($2)

label25: sw $31, 4($0)

sw $31, 8($0)

sw $31, 12($0)

ori $1, $0, 4

jal label26

addu $31, $31, $1

nop

j label27

nop

label26: jr $31

nop

label27: jal label28

addu $31, $31, $1

nop

j label29

nop

label28: nop

jr $31

nop

label29: jal label30

addu $31, $31, $1

nop

j label31

nop

label30: nop

nop

jr $31

nop

label31: nop

nop

ori $1, $0, 4

jal label32

addu $31, $31, $1

j label33

nop

label32: subu $31, $31, $1

jr $31

nop

label33: jal label34

addu $31, $31, $1

j label35

nop

label34: subu $31, $31, $1

nop

jr $31

nop

label35: jal label36

addu $31, $31, $1

j label37

nop

label36: subu $31, $31, $1

nop

nop

jr $31

nop

label37: nop

ori $9, $9, 0x3000

jal label38

subu $31, $31, $9

j label39

nop

label38: ori $31, $31, 0x3000

jr $31

nop

label39: jal label40

subu $31, $31, $9

j label41

nop

label40: ori $31, $31, 0x3000

nop

jr $31

nop

label41: jal label42

subu $31, $31, $9

j label43

nop

label42: ori $31, $31, 0x3000

nop

nop

jr $31

nop

label43: jal label44

nop

j label45

nop

label44: jr $31

nop

label45: jal label46

nop

j label47

nop

label46: nop

jr $31

nop

label47: jal label48

nop

j label49

nop

label48: nop

nop

nop

jr $31

label49: nop

ori $1, $0, 0

lw $1, 0($0)

addu $2, $1, $1

addu $3, $2, $1

addu $4, $3, $1

addu $5, $4, $1

ori $1, $0, 0

nop

nop

nop

nop

lw $1, 0($0)

subu $2, $1, $1

subu $3, $2, $1

subu $4, $3, $1

subu $5, $4, $1

lw $2, 0($0)

ori $1, $2, 123

ori $1, $2, 321

ori $1, $2, 245

ori $1, $2, 1234

lw $3, 0($0)

beq $3, $1, label50

nop

beq $3, $1, label50

nop

beq $3, $1, label50

nop

beq $3, $1, label50

nop

label50: nop

lw $4, 0($0)

sw $4, 0x2000($0)

sw $4, 0x2004($0)

sw $4, 0x2008($0)

sw $4, 0x200c($0)

ori $9, $0, 0x3000

subu $4, $4, $9

sw $4, 0x0000($4)

jal label51

sw $31, 0($0)

jal label52

sw $31, 0($0)

jal label53

sw $31, 0($0)

j label54

label51: lw $1, 0($0)

jr $1

nop

label52: lw $1, 0($0)

nop

jr $1

nop

label53: lw $1, 0($0)

nop

nop

jr $1

label54: nop

ori $2, $0, 4

addu $1, $2, $0

lw $3, 0($1)

lw $3, 4($1)

lw $3, 8($1)

lw $3, -4($1)

subu $3, $1, $1

lw $4, 0($3)

lw $4, 4($3)

lw $4, 8($3)

lw $4, 12($3)

ori $3, $0, 8

lw $2, -8($3)

lw $2, -4($3)

lw $2, 0($3)

lw $2, 4($3)

sw $3, 0($0)

lw $1, 0($0)

lw $2, 0($1)

lw $2, -4($1)

lw $2, 4($1)

lw $2, -8($1)

ori $2, $0, 123

lui $2, 0

lw $3, 0($2)

lw $3, 4($2)

lw $3, 8($2)

lw $3, 12($2)

jal label55

lw $4, -0x3000($31)

label55: lw $4, -0x3000($31)

lw $4, -0x3000($31)

lw $4, -0x3000($31)

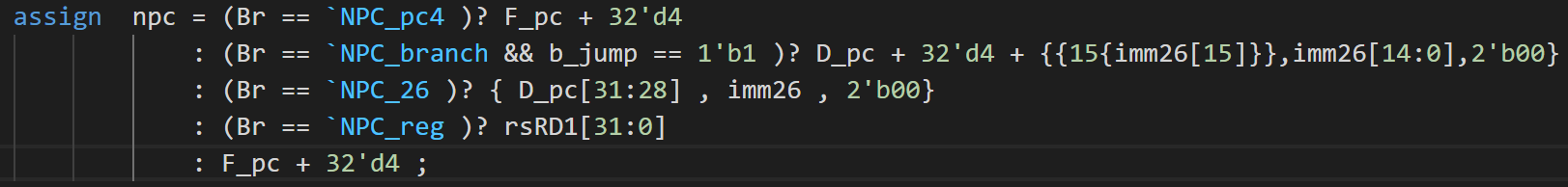
sw $1, 0($4)

六．思考题

## 流水线冒险

1. 在采用本节所述的控制冒险处理方式下，PC 的值应当如何被更新？请从数据通路和控制信号两方面进行说明。

答：



首先，我们在D阶段根据D\_Instr进行NEW\_PC的判断。

控制信号与单周期类似，分为顺序执行、branch跳转、j跳转到imm26对应的地址和j跳转到寄存器存储的地址这四种。其中顺序执行时基于流水线中最新的指令进行的（也就是F级指令），而其他各类跳转都是基于D\_Instr执行的。跳转指令之后的nop延迟槽，可以保证跳转地址的正确性。

关于数据通路：branch类跳转指令的比较被提前到D级的CMP部件执行；j跳转到寄存器存储的地址、地址采用D级rs经过转发后的量；j跳转到imm26对应的地址的采用D级Instr中的立即数部分作为基准。

1. 对于 jal 等需要将指令地址写入寄存器的指令，为什么需要回写 PC+8 ？

答：因为MIPS是采用延时槽技术，在D段决定是否进行跳转，这个时候下一条指令已经进入了流水线，所以在跳转指令的下一条指令必须被执行，这就是延时槽技术。故采用的地址为：PC+8。

## 数据冒险的分析

1. 为什么所有的供给者都是存储了上一级传来的各种数据的**流水级寄存器**，而不是由 ALU 或者 DM 等部件来提供数据？

答：转发信号作为接收者的输入信号，从流水线寄存器传来可以保证所有部件的输入信号只在时钟上升沿更新，这样数据安全性更高更稳定；同时这样也能保证转发数据的来源同样是经过转发了的数据。

## AT 法处理流水线数据冒险

1. “转发（旁路）机制的构造”中的 Thinking 1-4；

**Thinking 1**：如果不采用已经转发过的数据，而采用上一级中的原始数据，会出现怎样的问题？

**Answer 1： 会出现Tuse<Tnew的问题；上一级中的原始数据可能包含组合逻辑部分，会在非clk上升沿更新，使流水线数据安全性降低。**

**Thinking 2**：我们为什么要对 GPR 采用内部转发机制？如果不采用内部转发机制，我们要怎样才能解决这种情况下的转发需求呢？

**Answer 2：GRF在一个时钟周期里是读写两用的；上升沿写入下降沿读出（半周期写入半周期读出）。**

**Thinking 3**：为什么 0 号寄存器需要特殊处理？

**Answer 3：若当前位点的读取寄存器地址和某转发输入来源的写入寄存器地址均为0，则当前位点的读取值一定为0，但是转发来源的写入值不一定为0，这样会造成错误。**

**Thinking 4**：什么是“最新产生的数据”？

**Answer 4:** 在当前指令之前、离当前指令最近的指令产生的数据，也就是后续最近的流水线寄存器中的有效数据。

1. 在 AT 方法讨论转发条件的时候，只提到了“供给者需求者的A相同，且不为 0”，但在 CPU 写入 GRF 的时候，是有一个 we 信号来控制是否要写入的。为何在 AT 方法中不需要特判 we 呢？为了**用且仅用** A 和 T 完成转发，在翻译出 A 的时候，要结合 we 做什么操作呢？

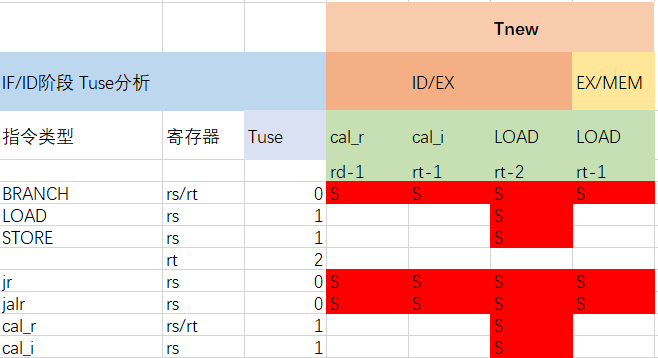
答：因为我实现转发机制时，转发值还并没有真正的写入寄存器，转发使用的正是还在流水线中流水的数据。我在实现转发时，对转发数据的使用的的地址依照指令功能的需要、做出了rs、rt的区分，也就是对aRead的选择已经做出了限制。当aRead和转发提供者的aWrite 相等时，就已经满足了AT法转发的条件。

## 在线测试相关说明

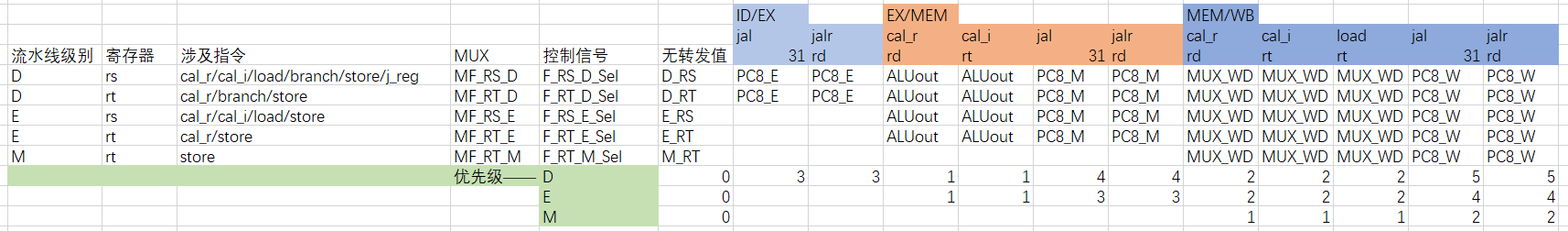
在本实验中你遇到了哪些不同指令类型组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？

如果你是手动构造的样例，请说明构造策略，说明你的测试程序如何保证**覆盖**了所有需要测试的情况；如果你是**完全随机**生成的测试样例，请思考完全随机的测试程序有何不足之处；如果你在生成测试样例时采用了**特殊的策略，**比如构造连续数据冒险序列，请你描述一下你使用的策略如何**结合了随机性**达到强测的效果。

暂停↓



转发↓



上述两表是AT法分析指令数据冲突在MIPS—2的指令集下的全部情况，对每一种情况分别构造测试程序分析即可。（见前文的测试程序部分）