计算机组成原理 P6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-lite3。采用模块化和层次化设计。

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 |

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  ); |

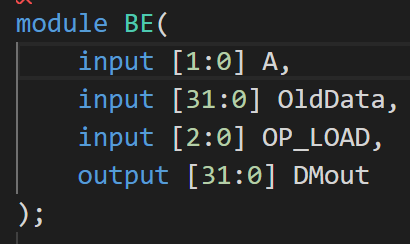
8.ALU

|  |  |  |
| --- | --- | --- |
| 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..BE (E级的存储数据扩展模块)

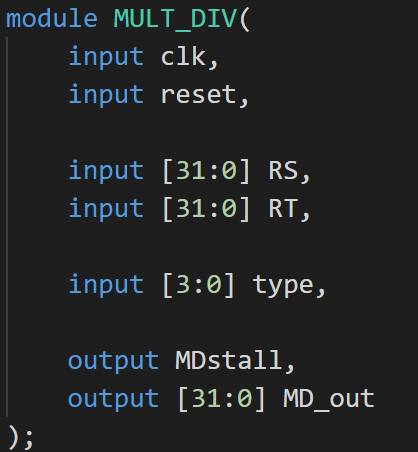


|  |  |  |
| --- | --- | --- |
| A | I | 被加载的内存的地址的末两位 |
| OldData | I | 被加载内存的地址存储的整字数据 |
| OP\_LOAD | I | Load指令类型的控制信号 |
| DM\_OUT | O | 加载内存的结果 |

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  ); |

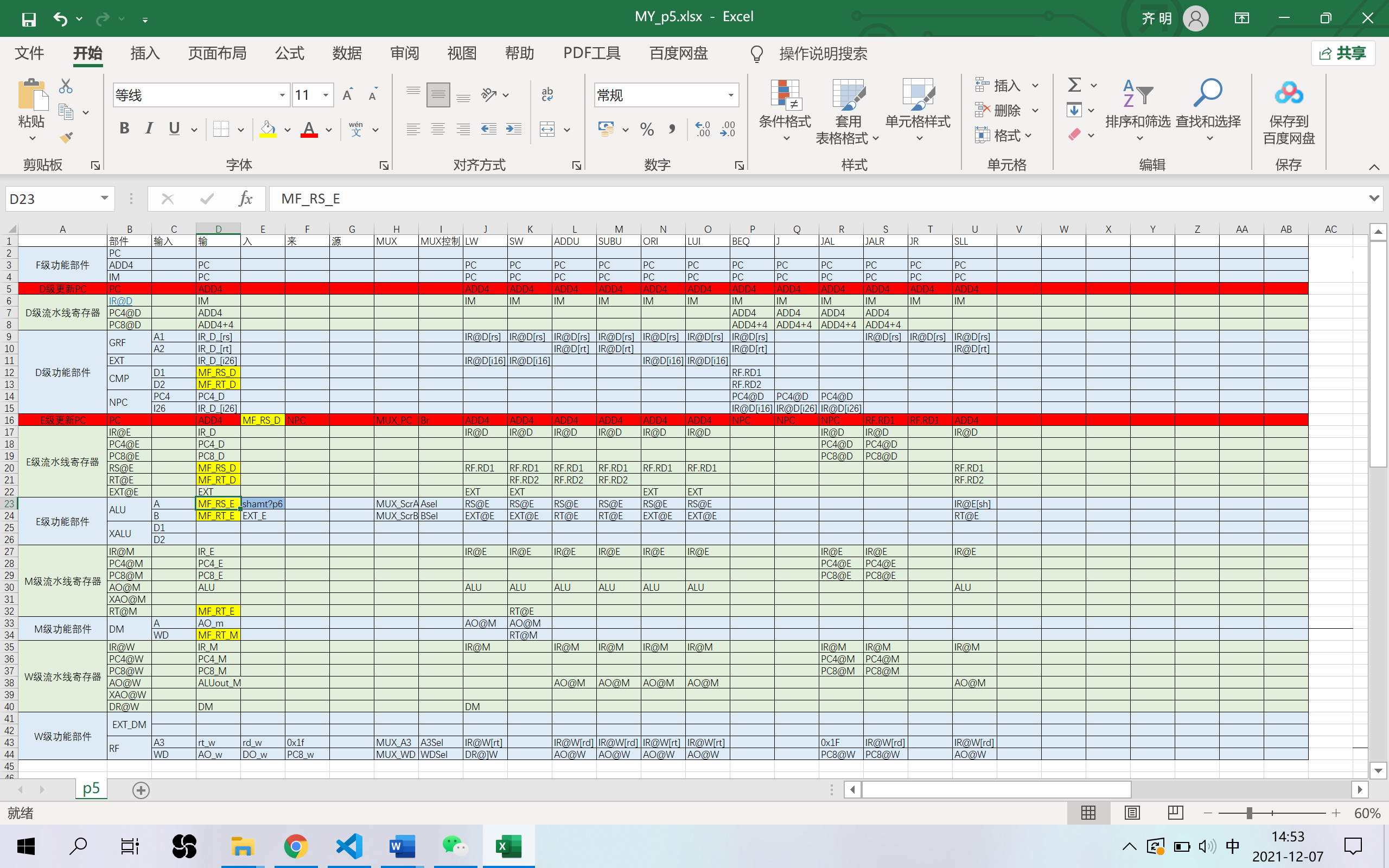
12.MULT\_DIV （乘除模块）



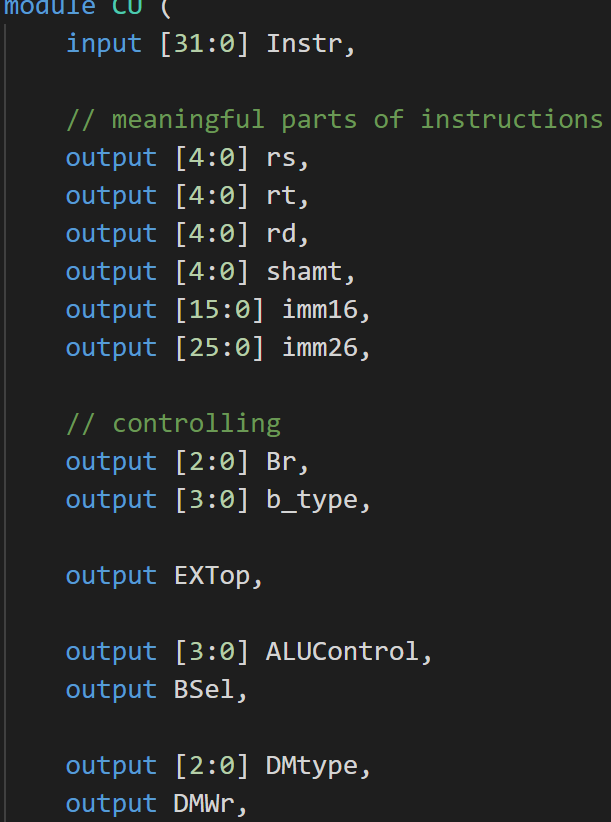
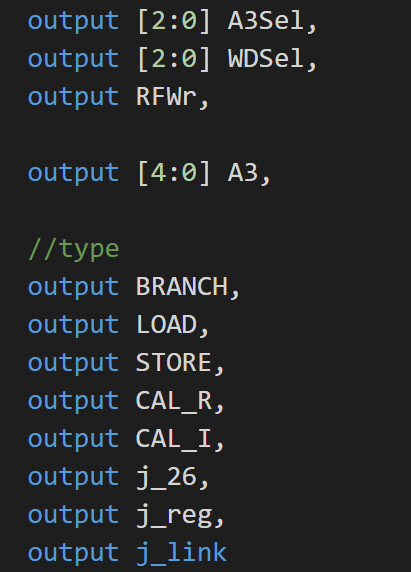
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 功能描述 |
| RS[31:0] | I | 32 位输入数据 1 |
| RT[31:0] | I | 32 位输入数据 2 |
| TYPE[3:0] | I | 乘除单元指令控制信号控制信号 |
| Clk | I | 时钟信号 |
| reset | I | 复位信号 |
| MDstall | O | 标志乘除模块正在启动（start）或者正在工作延迟中（busy） |
| MD\_OUT | 0 | 乘除模块的输出 |

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）

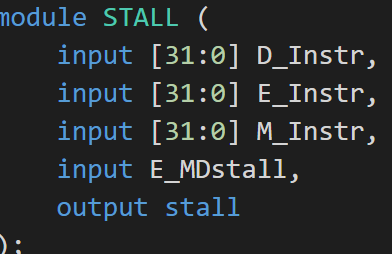
 

具体控制信号的生成，采取先对指令进行指令类型判断，再根据指令类型得到控制型号取值的方法。具体实现见CU.v。

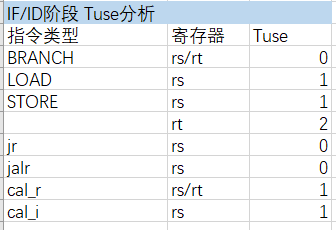
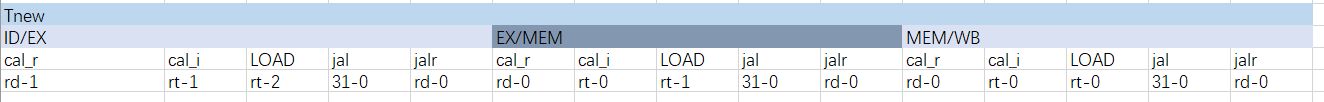
在流水线阶段，根据使用需要，将CU实例化四次。

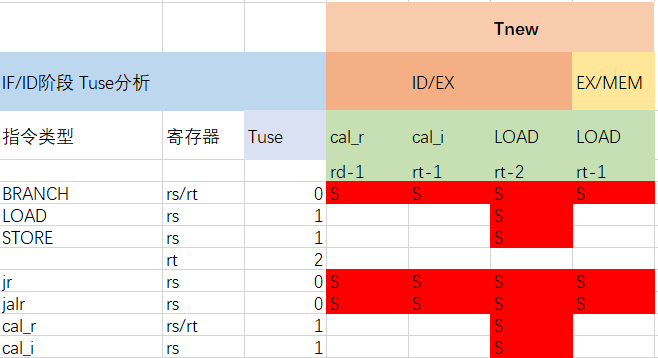
四．冒险处理单元设计(FWD,STALL)———使用**需求时间——供给时间模型**。

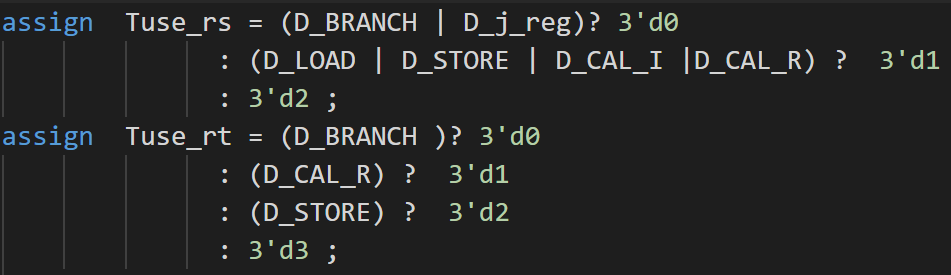
1．Stall 暂停信号的生成

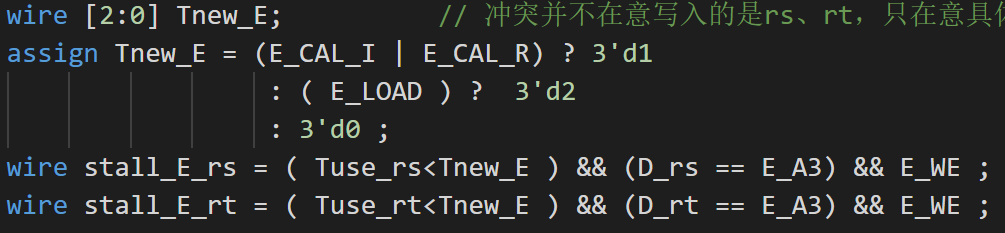


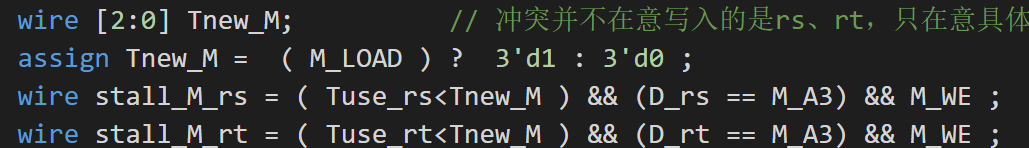
通过D、E、M级流水线寄存器的指令类型来判断stall信号，从而完成流水线的暂停（即寄存器使能置零和值清空）









相较P5新增对乘除模块暂停的判断

    // as data giver(mf) : Tnew

    // as data user (md,mt) : Tuse

    // consider inner problems :  MUL\_DIV is busy but we are going to use this module.

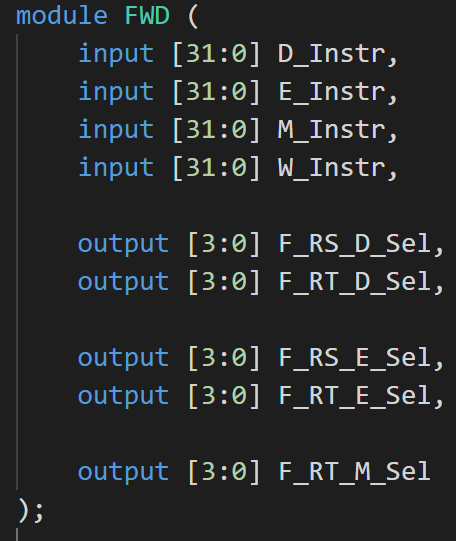
1. mf 为GRF提供数据，作为数据的提供者，判断其Tnew（在E阶段Tnew为1，在M阶段Tnew为0）
2. md、mt 是数据的使用者，判断其Tuse，均在E级使用数据，故在D级Tuse均为1；md的rs和rt均需要使用数据、而mt只有rs需要使用数据
3. 还需考虑专属于MultDiv模块的冲突。当E级的乘除模块处于启动start或者工作busy状态、而D级指令又是乘除类指令时，若不在D级将指令暂停，则乘除模块内部会造成冲突（主要是hi、lo两寄存器的冲突），故对该种情况造成的暂停进行特判：

wire stall\_MD = E\_MDstall && ( D\_md | D\_mf | D\_mt) ;

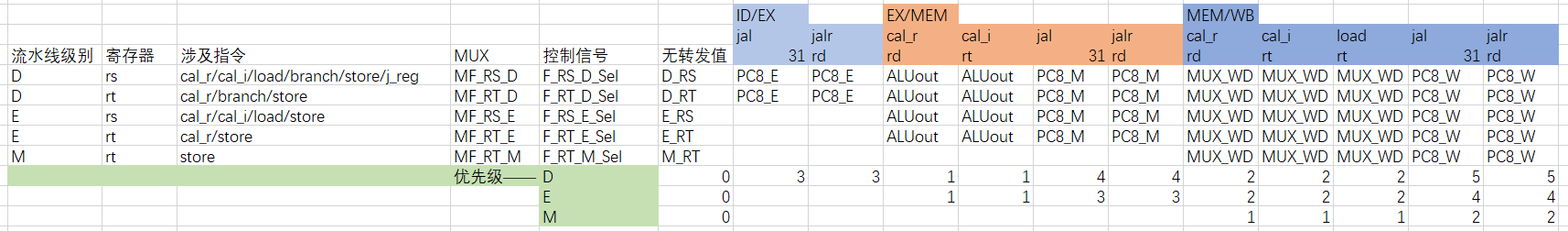
assign stall = stall\_E\_rs | stall\_E\_rt | stall\_M\_rs | stall\_M\_rt | stall\_MD;

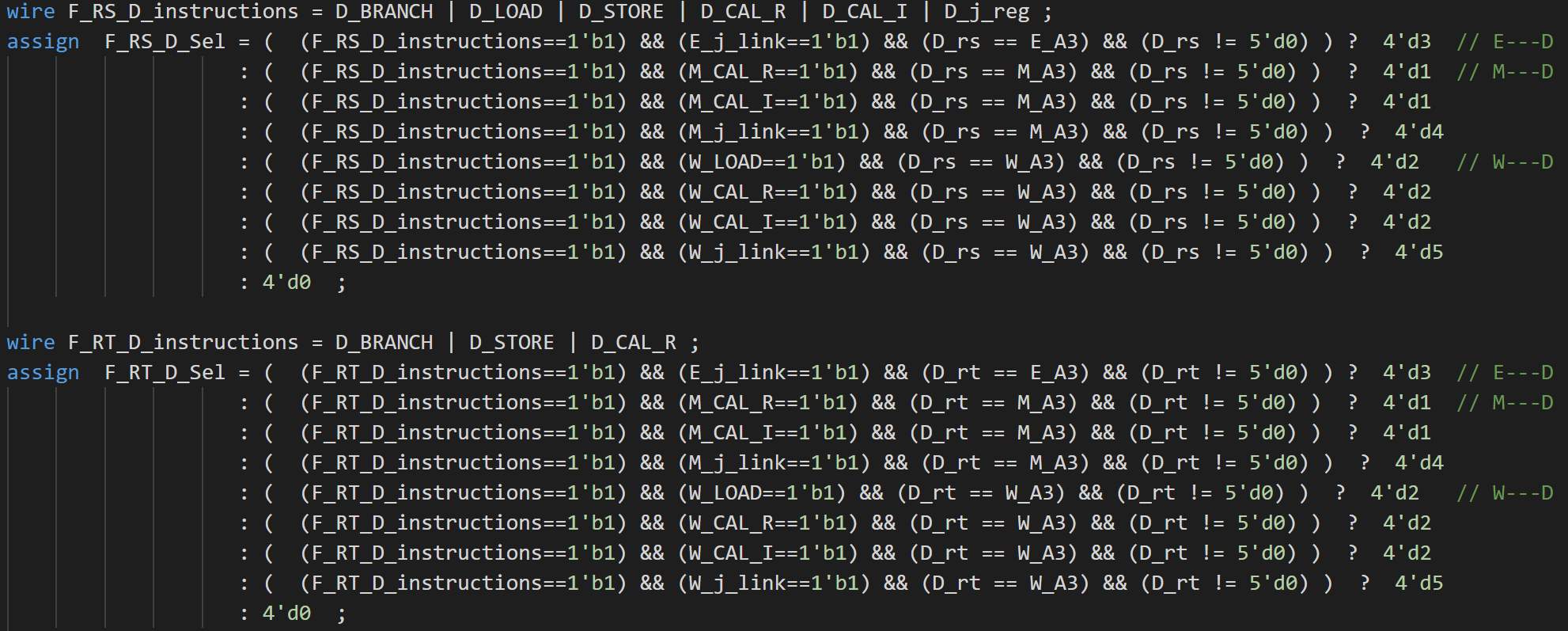
总的暂停信号通过各级rs、rt处发生的暂停信号进行按位与后得到。

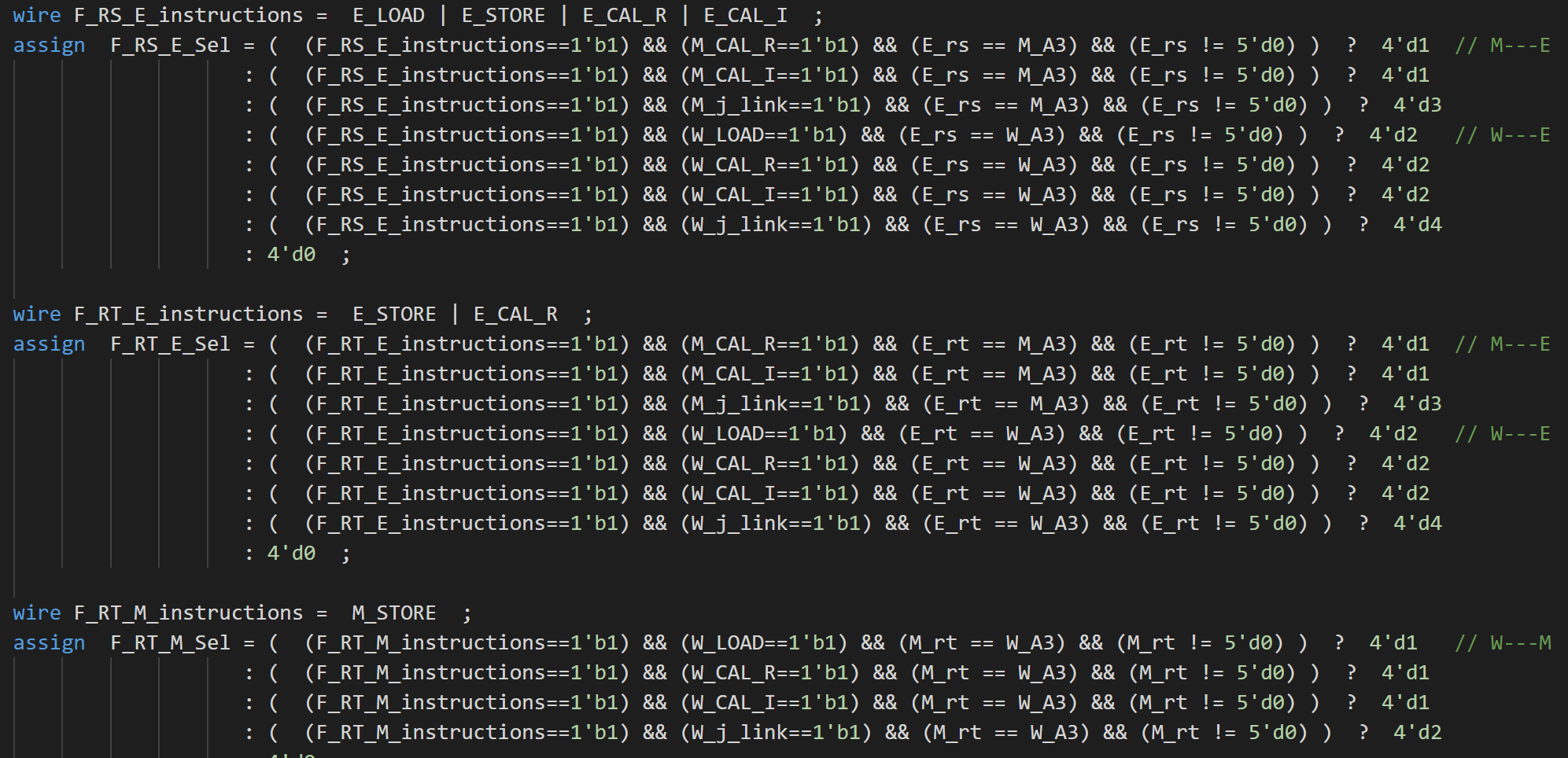
2．转发MUX的选择信号的生成



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

在P5的基础上，需要额外测试乘除模块的冲突；

Tuse Tnew分析时，md类指令可与cal\_R等效，mt类指令可与cal\_I等效；

然后再测试当E级的乘除模块处于启动start或者工作busy状态、而D级指令又是乘除类指令这种冲突

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）

（1）P5 测试转发功能

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)

（2）测试p6新增指令行为

Addi addiu

ori $2,$zero,0x1

addiu $3,$2,3

addiu $4,$2,-3

ori $2,$zero,0x1

addi $3,$2,3

addi $4,$2,-3

add addu sub subu

lui $7 ,0xffff

ori $7 ,$7,0xffff

ori $8 ,$zero,0x2

add $9,$7,$8

ori $10 ,$0,0xffff

ori $11 ,$0,0x2

addu $12,$10,$11

ori $13 ,$0,0x2

ori $14 ,$0,0xff

sub $15,$13,$14

ori $16 ,$0,0x33

ori $17 ,$0,0x22

subu $18,$16,$17

and andi slt slti sltu sltiu

ori $7 ,$zero ,0x1111

ori $8 ,$zero ,0x2223

li $9,0x12131111

and $10, $7,$8

andi $11, $7,34

andi $12 ,$8,563

ori $13 ,$zero ,2

ori $14 ,$zero ,1

ori $15 ,$zero ,2

ori $16 ,$zero ,3

slt $17,$13,$14 # >

slt $18,$13,$15 # =

slt $19,$13,$16 # <

sltu $17,$13,$14 # >

sltu $18,$13,$15 # =

sltu $19,$13,$16 # <

slti $17,$14 ,34 # >

slti $18,$15 ,2 # =

slti $19,$16 ,4 # <

sltiu $17,$14 ,34 # >

sltiu $18,$15 ,2 # =

sltiu $19,$16 ,4 # <

BRANCH

ori $1 ,$zero ,5

ori $2 ,$zero ,5

ori $3 ,$zero ,6

beq $1 ,$2 ,cut1

lui $10 ,0xffff

cut1:

lui $11 ,0xffff

beq $1 ,$3 ,cut2

lui $12 ,0xffff

cut2:

lui $13 ,0xffff

###################################################

ori $1 ,$zero ,5

ori $2 ,$zero ,5

ori $3 ,$zero ,6

bne $1 ,$3 ,cut3

lui $10 ,0xffff

cut3:

lui $11 ,0xffff

bne $1 ,$2 ,cut4

lui $12 ,0xffff

cut4:

lui $13 ,0xffff

###############################################################

ori $1 ,$zero ,5

ori $2 ,$zero ,0

ori $3 ,$zero ,-3

blez $1 ,cut5

lui $10 ,0xffff

cut5:

lui $11 ,0xffff

blez $2 ,cut6

lui $12 ,0xffff

cut6:

lui $13 ,0xffff

blez $3 ,cut7

lui $14 ,0xffff

cut7:

lui $15 ,0xffff

#############################################################################33333

ori $1 ,$zero ,5

ori $2 ,$zero ,0

ori $3 ,$zero ,-3

bgtz $1 ,cut8

lui $10 ,0xffff

cut8:

lui $11 ,0xffff

bgtz $2 ,cut9

lui $12 ,0xffff

cut9:

lui $13 ,0xffff

bgtz $3 ,cut10

lui $14 ,0xffff

cut10:

lui $15 ,0xffff

#############################################################################33333

ori $1 ,$zero ,5

ori $2 ,$zero ,0

ori $3 ,$zero ,-3

bltz $1 ,cut11

lui $10 ,0xffff

cut11:

lui $11 ,0xffff

bltz $2 ,cut12

lui $12 ,0xffff

cut12:

lui $13 ,0xffff

bltz $3 ,cut13

lui $14 ,0xffff

cut13:

lui $15 ,0xffff

#############################################################################33333

ori $1 ,$zero ,5

ori $2 ,$zero ,0

ori $3 ,$zero ,-3

bgez $1 ,cut14

lui $10 ,0xffff

cut14:

lui $11 ,0xffff

bgez $2 ,cut15

lui $12 ,0xffff

cut15:

lui $13 ,0xffff

bgez $3 ,cut16

lui $14 ,0xffff

cut16:

lui $15 ,0xffff

逻辑运算

and $10, $7,$8

andi $11, $10,34

andi $12 ,$11,563

or $10, $7,$8

ori $11, $10,45

ori $12 ,$11,11

xor $10, $7,$8

xori $11, $10,43

xori $12 ,$11,595

nor $10, $7,$8

nor $11, $10, $10

nor $12 ,$11, $11

load 、 store

.text

ori $8, $zero, 0

lui $9,0x1234

ori $9, $9, 0x56b8

sw $9, 0($8)

sh $9, 4($8)

sh $9, 2($8)

sb $9, 0($8)

sb $9, 1($8)

sb $9, 2($8)

sb $9, 3($8)

lb $11, 0($8)

lbu $12, 1($8)

lb $13,2($8)

lb $14,3($8)

lh $15, 0($8)

lhu $16, 2($8)

add $17,$16,$16

移位

ori $8, $zero, 1

sll $8, $8, 2

ori $8, $zero, 0x1000

srl $8, $8, 2

li $8, 0xf0000000

sra $8, $8, 2

li $9 ,2

ori $8, $zero, 1

sllv $8, $8,$9

ori $8, $zero, 0x1000

srlv $8, $8, $9

li $8, 0xf0000000

srav $8, $8, $9

乘除指令

ori $1, $0, 4

ori $2, $0, 5

ori $3, $0, 6

ori $4, $0, 12

sw $3, 0($0)

sw $2, 4($0)

sw $2, 8($0)

sw $2, 12($0)

sw $1, 16($0)

sw $4, 20($0)

sw $2, 24($0)

sw $4, 28($0)

sw $3, 32($0)

sw $3, 36($0)

sw $2, 40($0)

sw $2, 44($0)

sw $2, 48($0)

sw $4, 52($0)

sw $1, 56($0)

sw $3, 60($0)

sw $2, 64($0)

sw $1, 68($0)

sw $2, 72($0)

sw $2, 76($0)

sw $4, 80($0)

sw $1, 84($0)

sw $3, 88($0)

sw $3, 92($0)

sw $1, 96($0)

sw $2, 100($0)

sw $2, 104($0)

sw $2, 108($0)

sw $1, 112($0)

sw $1, 116($0)

sw $1, 120($0)

sw $4, 124($0)

lbu $1, 0($2)

beq $1, $2, TAG1

lbu $4, 0($1)

lw $3, 0($1)

TAG1:

lui $2, 13

sb $3, 0($3)

lui $2, 7

and $3, $2, $3

TAG2:

mtlo $3

addi $2, $3, 11

mult $2, $2

lui $3, 6

TAG3:

mtlo $3

lui $1, 14

sll $0, $0, 0

slti $4, $1, 15

TAG4:

mtlo $4

lbu $1, 0($4)

sltiu $4, $4, 7

addiu $4, $4, 12

TAG5:

subu $2, $4, $4

lui $2, 3

sll $0, $0, 0

div $2, $2

TAG6:

mflo $1

bgtz $4, TAG7

sb $4, 0($4)

beq $1, $1, TAG7

TAG7:

lb $1, 0($1)

bne $1, $1, TAG8

sh $1, 0($1)

bgtz $1, TAG8

TAG8:

mthi $1

mflo $4

lbu $4, 0($4)

mtlo $4

TAG9:

beq $4, $4, TAG10

mthi $4

mthi $4

lw $3, 0($4)

TAG10:

xor $4, $3, $3

bne $4, $3, TAG11

mtlo $4

mfhi $4

TAG11:

sltu $2, $4, $4

mflo $1

sb $4, 0($2)

mflo $2

TAG12:

bgtz $2, TAG13

ori $2, $2, 13

slti $2, $2, 10

bgez $2, TAG13

TAG13:

sh $2, 0($2)

mthi $2

bltz $2, TAG14

mtlo $2

TAG14:

lw $4, 0($2)

bltz $4, TAG15

mfhi $2

sltiu $4, $4, 10

TAG15:

mflo $2

sb $2, 0($2)

srl $4, $2, 0

mtlo $2

TAG16:

lui $4, 7

ori $2, $4, 15

bgez $4, TAG17

mthi $2

TAG17:

ori $4, $2, 6

sllv $4, $4, $2

bne $4, $2, TAG18

lui $1, 2

TAG18:

sll $0, $0, 0

mflo $3

lui $1, 12

mthi $1

TAG19:

mfhi $2

mfhi $1

multu $1, $1

mfhi $1

TAG20:

sw $1, -144($1)

lui $4, 8

sll $0, $0, 0

lhu $1, -144($1)

TAG21:

slt $2, $1, $1

mult $2, $2

lui $4, 3

bgez $2, TAG22

TAG22:

mfhi $2

xori $4, $2, 5

beq $4, $2, TAG23

mult $4, $4

TAG23:

lui $2, 6

div $2, $4

mfhi $2

subu $3, $2, $2

TAG24:

beq $3, $3, TAG25

slti $4, $3, 13

mthi $4

bne $3, $3, TAG25

TAG25:

sll $1, $4, 1

mfhi $4

mfhi $1

blez $1, TAG26

TAG26:

sb $1, 0($1)

bne $1, $1, TAG27

lb $4, 0($1)

lb $3, 0($4)

TAG27:

lui $3, 15

mfhi $2

mthi $3

beq $3, $3, TAG28

TAG28:

mtlo $2

lbu $4, 0($2)

ori $2, $2, 15

blez $4, TAG29

TAG29:

sltu $2, $2, $2

lw $2, 0($2)

mthi $2

lb $2, -400($2)

TAG30:

lui $4, 9

lw $2, 112($2)

xor $3, $2, $2

mflo $4

TAG31:

mtlo $4

bgez $4, TAG32

lui $2, 2

multu $4, $4

TAG32:

mfhi $4

sw $4, -400($4)

mtlo $4

slti $2, $2, 3

TAG33:

mfhi $4

lui $3, 14

and $1, $3, $2

lhu $4, 0($1)

TAG34:

sll $0, $0, 0

lh $3, -400($4)

lw $1, -400($4)

sra $1, $1, 2

TAG35:

lh $1, 0($1)

lbu $2, 0($1)

bltz $2, TAG36

mtlo $1

TAG36:

lui $1, 0

sb $1, 0($2)

lui $1, 3

mtlo $1

TAG37:

blez $1, TAG38

mtlo $1

sll $0, $0, 0

sll $0, $0, 0

TAG38:

beq $4, $4, TAG39

lh $4, -400($4)

mfhi $1

divu $1, $4

TAG39:

bne $1, $1, TAG40

sll $0, $0, 0

sll $0, $0, 0

sll $0, $0, 0

TAG40:

xori $1, $1, 8

bne $1, $1, TAG41

mtlo $1

blez $1, TAG41

TAG41:

lui $4, 8

blez $1, TAG42

mfhi $3

blez $1, TAG42

TAG42:

sh $3, -400($3)

sw $3, -400($3)

srl $3, $3, 9

beq $3, $3, TAG43

TAG43:

lui $3, 15

mthi $3

sll $0, $0, 0

mtlo $3

TAG44:

sll $0, $0, 0

mflo $1

multu $3, $1

mtlo $1

TAG45:

mfhi $2

sb $1, -225($2)

lui $4, 6

lui $1, 2

TAG46:

multu $1, $1

bne $1, $1, TAG47

lui $1, 9

mflo $2

TAG47:

bne $2, $2, TAG48

sh $2, 0($2)

lui $3, 7

mthi $2

TAG48:

addu $3, $3, $3

sra $1, $3, 6

bgtz $3, TAG49

sltiu $4, $3, 13

TAG49:

mflo $4

bgez $4, TAG50

lui $2, 6

bgez $4, TAG50

TAG50:

mflo $3

lui $1, 2

lui $4, 6

mtlo $4

TAG51:

bne $4, $4, TAG52

mflo $1

mfhi $2

sra $4, $2, 0

TAG52:

sb $4, 0($4)

sh $4, 0($4)

srav $2, $4, $4

nor $2, $4, $4

TAG53:

divu $2, $2

mtlo $2

andi $4, $2, 1

lui $1, 13

TAG54:

sll $0, $0, 0

multu $1, $1

divu $1, $1

lui $1, 8

TAG55:

blez $1, TAG56

slti $4, $1, 5

sh $4, 0($4)

bgtz $4, TAG56

TAG56:

lui $3, 4

mult $3, $3

sll $0, $0, 0

mflo $4

TAG57:

lb $3, 0($4)

slti $2, $3, 11

bltz $2, TAG58

srl $4, $3, 13

TAG58:

sh $4, 0($4)

mflo $4

mfhi $1

mtlo $4

TAG59:

mthi $1

mthi $1

andi $4, $1, 7

lui $4, 1

TAG60:

sll $0, $0, 0

sll $0, $0, 0

sh $1, 0($1)

sll $0, $0, 0

TAG61:

xori $2, $1, 0

srlv $2, $1, $2

mtlo $1

slti $3, $2, 14

TAG62:

blez $3, TAG63

mflo $4

mfhi $2

div $2, $3

TAG63:

bne $2, $2, TAG64

subu $3, $2, $2

mult $2, $3

divu $2, $2

TAG64:

sh $3, 0($3)

lbu $2, 0($3)

sw $3, 0($3)

bgez $2, TAG65

TAG65:

ori $3, $2, 14

sb $3, 0($3)

xor $1, $3, $3

lui $1, 15

TAG66:

lui $3, 0

bgez $1, TAG67

sll $0, $0, 0

sltu $3, $3, $1

TAG67:

lui $2, 5

mtlo $3

lw $2, 0($3)

mthi $2

TAG68:

lui $1, 10

bne $1, $2, TAG69

sltiu $3, $1, 3

mtlo $2

TAG69:

beq $3, $3, TAG70

multu $3, $3

mfhi $4

sb $3, 0($4)

TAG70:

sw $4, 0($4)

mflo $4

beq $4, $4, TAG71

xor $4, $4, $4

TAG71:

nor $3, $4, $4

subu $1, $3, $4

mfhi $4

addu $4, $3, $4

TAG72:

addu $4, $4, $4

beq $4, $4, TAG73

divu $4, $4

or $1, $4, $4

TAG73:

blez $1, TAG74

sw $1, 1($1)

mtlo $1

srav $4, $1, $1

TAG74:

sh $4, 2($4)

mfhi $1

sltu $2, $4, $4

lh $4, 0($2)

（3）测试 因为增加p6乘除指令带来的暂停

见（2）乘除指令部分

（4）综合测试：使用讨论区强测数据（见附件）

六．思考题

* 为什么需要有单独的乘除法部件而不是整合进 ALU？为何需要有独立的 HI、LO 寄存器？

答：实际的CPU实现乘除法也是通过加法实现的，实现乘法相较ALU里的简单计算耗时更久，如果并入ALU会降低整个CPU的运行效率。

HI、LO寄存器可以使有多周期时延的乘除法过程和无多周期时延的ALU计算过程相互独立，再通过stall暂停机制的巧妙运转，可以提高CPU的运算效率。

* 参照你对延迟槽的理解，试解释 “乘除槽”。

答：类似延迟槽，当乘除法进行的时候，会有一个 start 信号，在下一个周期会产生busy 信号，但是这个时间内并不影响其他指令的执行，而且不止一条其他指令，而延迟槽只是一条指令的继续执行。只要md类型指令后边跟的不是乘除法型指令，就不需要暂停了，乘除槽使得其他跟在后面的指令的执行不受影响。

* 举例说明并分析何时按字节访问内存相对于按字访问内存性能上更有优势。（Hint： 考虑 C 语言中字符串的情况）

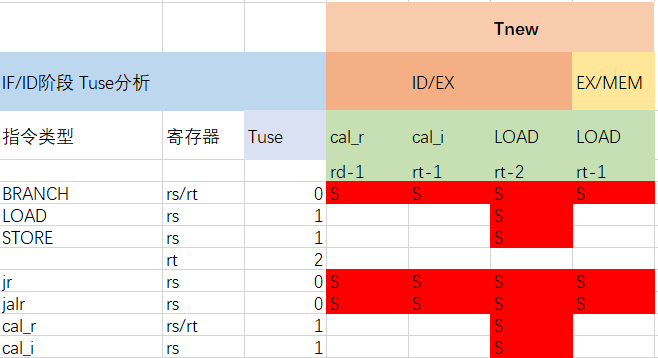
答：“abcd”是字符串常量，连续存储。按照字访问，想取出一个字符，需要先取字，再取字符。而字节访问就比较简单，可以直接取一个字符。此时按字节访问内存相对于按字访问内存性能上更有优势。

* 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？

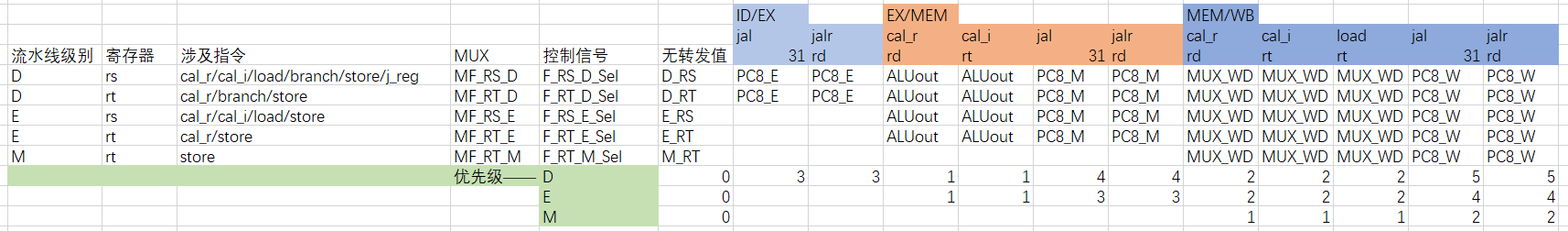
如果你是手动构造的样例，请说明构造策略，说明你的测试程序如何保证**覆盖**了所有需要测试的情况；如果你是**完全随机**生成的测试样例，请思考完全随机的测试程序有何不足之处；如果你在生成测试样例时采用了**特殊的策略**，比如构造连续数据冒险序列，请你描述一下你使用的策略如何**结合了随机性**达到强测的效果。

此思考题请同学们结合自己测试 CPU 使用的具体手段，按照自己的实际情况进行回答

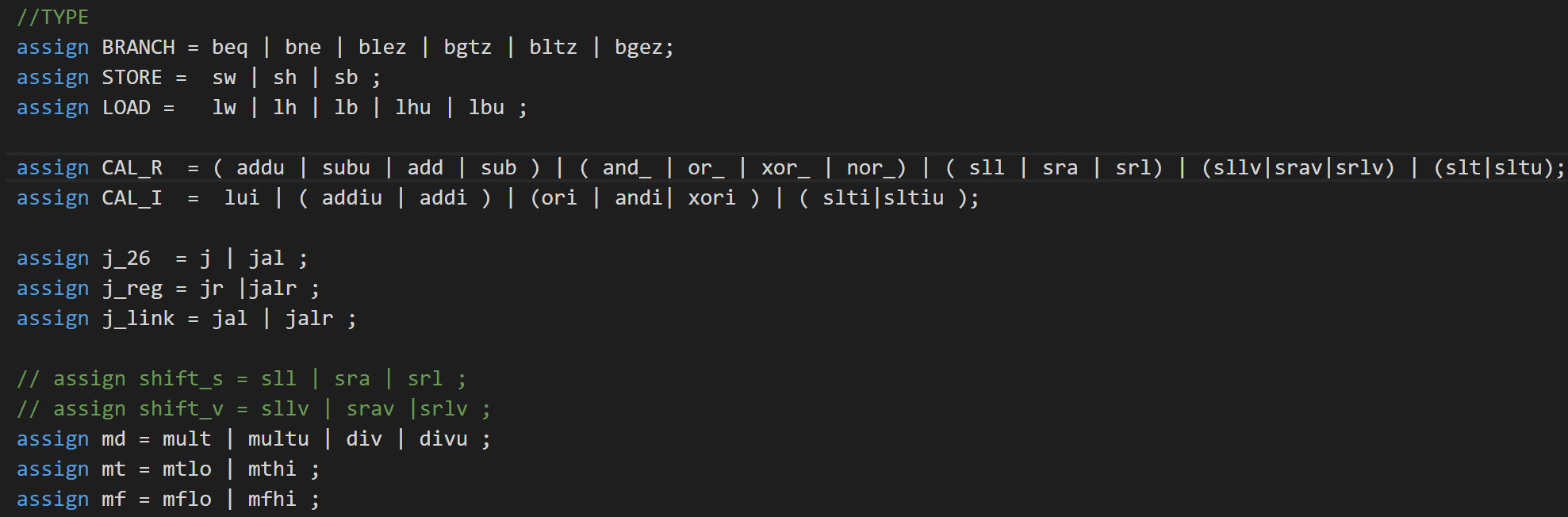
暂停↓



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，再加上乘除法指令带来的新的暂停情况，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）

* 为了对抗复杂性你采取了哪些抽象和规范手段？这些手段在译码和处理数据冲突的时候有什么样的特点与帮助？
* 1. 指令func、opcode段采用parameter常数替代，减少错误
* 2. 控制信号取值用宏表述，方便在多文件中复用
* 3. 对指令进行分类，并将指令分类结果作为CU的输出，在需要使用时可以进行分布式译码

好处：在用AT法分析CPU的暂停和阻塞的时候，可以合并指令同类项，减少错误并且降低了指令的复杂度，增加了指令的可扩展性。

例如：jal，jalr同属于j\_link, 意思是跳转并链接PC+8的指令，这样在译码WDSel信号的时候就可以将两个指令合并。

