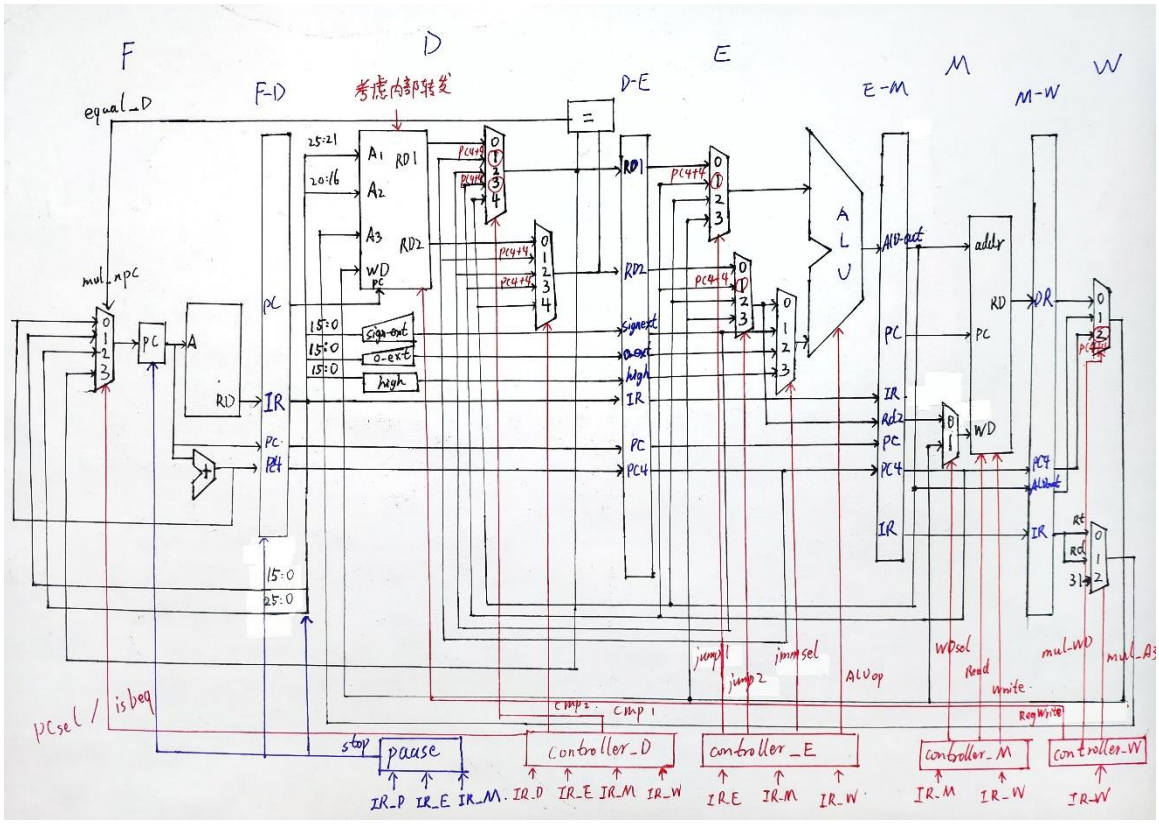
计算机组成原理 P5 CPU 设计实验报告



一、模块规格

1. F 级

F 级端口定义表

信号名	方向	描述
Clk	I	时钟信号
Reset	I	复位信号
Stop	I	暂停信号（D 级产生）
[31:0]IR_E	I	E 级 IR
[31:0]IR_M	I	M 级 IR
[15:0]imm16	I	16 位地址（D 级产生）
[25:0]imm26	I	26 位地址（D 级产生）
[31:0]rs	I	rs 寄存器回写 pc（D 级产生）
[2:0]pcsel	I	Npc 选择信号（D 级产生）

Equal	I	Beq 判断相等信号（D 级产生）
Isbeq	I	Beq 信号判断（D 级产生）
[31:0]Pc_F	O	F 级当前的 pc
[31:0]Pc4_F	O	F 级的 pc+4
[31:0]IR_F	O	F 级当前的 IR

1.1 IM

（1） 端口说明

IM 端口定义表

信号名	方向	描述
[31:0]Address	I	指令地址
[31:0]IR	O	读出指令

（2）功能定义

IM 功能描述表

序号	功能定义	功能描述
1	加载地址	将指令的地址加载至 ROM 中
2	寻址	根据 address 找到对应指令

1.2 Npc

（2） 端口说明

npc 端口定义表

信号名	方向	描述
[31:0]Pc4	I	Pc+4 的值
[31:0]Rs	I	rs 寄存器回写 pc
[2:0]Pcsel	I	Npc 选择信号
[15:0]Imm16	I	16 位地址
[25:0]Imm26	I	26 位地址
Isbeq	I	判断 beq 指令
Stop	I	暂停信号

Clk	I	时钟信号
Reset	I	复位信号
Equal	I	Beq 判断相等指令
[31:0]Nextpc	O	下一周期的 pc 值

（2）功能定义

npc 功能描述表

序号	功能定义	功能描述

1	计算下一周期 pc	Pcsrc==0
		Nextpc <= pc4;
		Pcsrc==1
		if(isbeq && equal)
		Nextpc <= pc4 + ext16;
		else
		Nextpc <= pc4;
		pcsrc === 2
		Nextpc<= ext26;
		pcsrc === 3
		Nextpc <= rs;

2	复位	复位信号有效时，Nextpc <= 32'h3000;
3	暂停	锁死 pc，pc = pc4 - 4

2．FD 级寄存器

FD 级寄存器端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
stop	I	暂停信号（D 级产生）(锁死 FD)
[31:0]IR	I	F 级 IR
[31:0]PC	I	F 级 PC

[31:0]PC4	I	F 级 PC+4
[31:0]IR_D	O	D 级 IR
[31:0]PC_D	O	D 级 PC
[31:0]PC4_D	O	D 级 PC4

3．D 级

D 级端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
[31:0]IR	I	D 级当前的 IR
[31:0]IR_E	I	E 级 IR
[31:0]IR_M	I	M 级 IR
[31:0]IR_W	I	W 级 IR
[31:0]pc	I	当前的 pc
[31:0]pc4	I	当前的 pc+4
[31:0]pc4_E	I	E 级 pc+4
[31:0]high_E	I	E 级加载立即数到高位
[31:0]pc4_M	I	M 级 pc+4
[31:0]ALUout_M	I	M 级的 ALUout
[31:0]mul_WD	I	W 级的回写数据
[4:0]mul_A3	I	W 级的回写地址
RegWrite_W	I	寄存器写入控制信号（W 级产生）
[31:0]pc_W	I	W 级 pc
[31:0]IR_D	O	D 级 IR
[31:0]Data1	O	Rs 读出的数据
[31:0]Data2	O	Rt 读出的数据
[31:0]Sign_ext	O	符号扩展
[31:0]Zero_ext	O	零扩展
[31:0]high	O	加载立即数到高位
[31:0]PC_D	O	D 级 pc
[31:0]PC4_D	O	D 级 pc+4

[15:0]imm16	0	16 位立即数
[25:0]imm26	0	26 位立即数
[2:0]PCsel	0	Npc 选择信号
Equal_D	0	Beq 相等信号
Isbeq_D	0	Beq 信号
Stop	0	暂停信号

3.1 GRF

（1） 端口说明

GRF 端口定义表

信号名	方向	描述
RegWrite	I	数据写入信号
Clk	I	时钟信号
Reset	I	复位信号
WD[31:0]	I	目标寄存器写入数据
pc[31:0]	I	指令地址
A1[4:0]	I	第一个寄存器源操作数
A2[4:0]	I	第二个寄存器源操作数
A3[4:0]	I	寄存器目标操作数
data1[31:0]	O	第一个源寄存器读出数据
data2[31:0]	O	第二个源寄存器读出数据

（2） 功能定义

GRF 功能描述表

序号	功能定义	功能描述
1	从寄存器堆读数据	读出 rs(A1)、rt(A2)地址对应寄存器中的数据到 data1 和 data2
2	向寄存器堆写数据	RegWrite 信号有效时，将 WD 写入 A3 地址对应的寄存器
3	复位	Reset 信号有效时，所有寄存器数据清零。
4	打印输出	`$display("%d@%h: $%d <= %h", $time ,pc, A3,WD)`

3.2 pause

（1） 端口说明

pause 端口定义表

信号名	方向	描述
[31:0]IR	I	D 级 IR
[31:0]IR_E	I	E 级 IR
[31:0]IR_M	I	M 级 IR
stop	O	暂停信号

（2） 功能定义

GRF 功能描述表

序号	功能定义	功能描述
1	判断暂停	根据 IR、IR_E、IR_M 判断是否需要暂停

4．DE 级寄存器

DE 级寄存器端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
[31:0]RD1	I	D 级 RD1
[31:0]RD2	I	D 级 RD2
[31:0]sign_ext	I	D 级符号扩展
[31:0]ext_0	I	D 级零扩展
[31:0]high	I	D 级加载立即数到高位
[31:0]IR	I	D 级 IR
[31:0]PC	I	D 级 PC
[31:0]PC4	I	D 级 PC+4
[31:0]RD1_E	O	E 级 RD1
[31:0]RD2_E	O	E 级 RD2
[31:0]Sign_ext_E	O	E 级符号扩展
[31:0]Ext_0_E	O	E 级零扩展

[31:0]High_E	O	E 级加载立即数到高位
[31:0]IR_E	O	E 级 IR
[31:0]PC_E	O	E 级 PC
[31:0]PC4_E	O	E 级 PC4

5. E 级

E 级端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
[31:0]IR	I	E 级当前的 IR
[31:0]IR_M	I	M 级 IR
[31:0]IR_W	I	W 级 IR
[31:0]pc4_M	I	M 级 pc+4
[31:0]ALUout_M	I	M 级的 ALUout
[31:0]mul_WD	I	W 级的回写数据
[31:0]zeroext	I	零扩展
[31:0]signext	I	符号扩展
[31:0]high	I	加载立即数到高位
[31:0]pc	I	Pc
[31:0]pc4	I	Pc+4
[31:0]Data1	I	Data1
[31:0]Data2	I	Data2
[31:0]ALUout_E	O	E 级 alu 输出
[31:0]data2_E	O	E 级转发后的 data2
[31:0]IR_E	O	E 级 IR
[31:0]PC_E	O	E 级 pc
[31:0]PC4_E	O	E 级 pc+4

5.1 ALU

（1） 端口说明

ALU 端口定义表

信号名	方向	描述
Alu1[31:0]	I	参与 ALU 计算的第一个值
Alu2[31:0]	I	参与 ALU 计算的第二个值
ALUop[3:0]	I	ALU 功能选择信号 0000：A&B 0001：A|B 0010：A+B 0110：A-B
aluout[31:0]	O	ALU 计算的结果

（2） 功能定义

ALU 功能描述表

序号	功能定义	功能描述
1	与运算	reselt = A & B
2	或运算	reselt = A | B
3	加运算	reselt = A + B
4	减运算	reselt = A - B

6．EM 级寄存器

EM 级寄存器端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
[31:0]ALUout	I	E 级 RD1
[31:0]data2	I	E 级 RD2
[31:0]IR	I	E 级 IR
[31:0]PC	I	E 级 PC
[31:0]PC4	I	E 级 PC+4
[31:0]ALUout_M	O	M 级 ALUout
[31:0]Data2_M	O	M 级 data2

[31:0]IR_M	O	M 级 IR
[31:0]PC_M	O	M 级 PC
[31:0]PC4_M	O	M 级 PC4

7. M 级

M 级端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
[31:0]IR	I	M 级当前的 IR
[31:0]IR_W	I	W 级 IR
[31:0]pc	I	Pc
[31:0]pc4	I	Pc+4
[31:0]ALUout	I	Alu 的输出
[31:0]data2	I	Data2
[31:0]mul_WD	I	W 级的回写数据
[31:0]ALUout_M	O	M 级 aluout
[31:0]DMout_M	O	M 级的 DM 输出
[31:0]IR_M	O	M 级 IR
[31:0]PC_M	O	M 级 pc
[31:0]PC4_M	O	M 级 pc+4

7.1 DM

（1） 端口说明

DM 端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
memWrite	I	信号有效时，向内存中写入数据
memRead	I	信号有效时，从内存中读出数据
pc[31:0]	I	指令地址

Address[31:0]	I	32 位内存地址
Writedata[31:0]	I	内存写入数据
readdata[31:0]	O	从内存中读出的数据

（2） 功能定义

DM 功能描述表

序号	功能定义	功能描述
1	读数据	根据地址信号从内存中读取数据
2	写数据	根据地址信号向内存中吸入数据
3	复位	Reset 信号有效时，内存清零
4	打印输出	$display("%d@%h: *%h <= %h",$time,pc, address, writedata);

8. MW 级寄存器

MW 级寄存器端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
[31:0]IR	I	M 级 IR
[31:0]PC	I	M 级 PC
[31:0]PC4	I	M 级 PC+4
[31:0]ALUout	I	M 级 ALUout
[31:0]DMout	I	M 级 DMout
[31:0]IR_W	O	W 级 IR
[31:0]PC_W	O	W 级 PC
[31:0]PC4_W	O	W 级 PC4
[31:0]ALUout_W	O	W 级 ALUout
[31:0]DMout_W	O	W 级 DMout

9. W 级

W 级端口定义表

信号名	方向	描述
[31:0]IR	I	W 级当前的 IR
[31:0]pc	I	Pc
[31:0]pc4	I	Pc+4
[31:0]ALUout	I	Alu 的输出
[31:0]DMout	I	DMout
[31:0]WD_W	O	W 级回写数据
[31:0]addr_W	O	W 级回写地址
RegWrite_W	O	W 级写寄存器控制信号

二、控制器设计

Controller

（1） 端口说明

Cotroller 端口定义表

信号名	方向	描述
IR[31:0]	I	当前流水级的 IR
D_IR[31:0]	I	D 级 IR
E_IR[31:0]	I	E 级 IR
M_IR[31:0]	I	M 级 IR
W_IR[31:0]	I	W 级 IR
RegWrite	O	寄存器写入控制
MemRead	O	内存读取控制
MemWrite	O	内存写入控制
IMMsel[1:0]	O	立即数选择信号
PCsel[2:0]	O	nPC 选择信号
ALUop[3:0]	O	ALU 功能选择信号
Mul_A3 [1:0]	O	写寄存器的目标寄存器来自 rt/rd/31 操作数
Mul_WD [1:0]	O	写入寄存器数据选择信号
Z_D_rs[2:0]	O	D 级 rs 转发选择信号
Z_D_rt[2:0]	O	D 级 rt 转发选择信号
Z_E_rs[2:0]	O	E 级 rs 转发选择信号

z_E_rt[2:0]	0	E 级 rt 转发选择信号
Z_M_rt[1:0]	0	M 级 rt 转发选择信号

（2）控制信号意义

控制信号	功能定义	功能描述
RegWrite	寄存器写入控制	寄存器写入使能有效
MemRead	内存读取控制	内存读取使能有效
MemWrite	内存写入控制	内存写入使能有效
Mul_A3[1:0]	写寄存器的目标寄存器来自 rt/rd/31 操作数	0： rt 1: rd 2: 目标寄存器 31
Mul_WD [1:0]	写入寄存器数据选择信号	00：向寄存器写入 DM 读取结果 01：向寄存器写入从 ALU 计算的值 10：向寄存器写入 pc+8
PCsel[2:0]	nPC 选择信号	00：pc+4 01：pc+4+shift2 or pc+4 10：PC31…28||shift2 11：Rdata1
ALUop[3:0]	ALU 功能选择信号	0000：A&B 0001：A|B 0010：A+B 0110：A-B
Imm_sel [1:0]	Data2-imm 选择信号	00：data2 01：sign-ext 10：0-ext 11：加载立即数到高位
Z_D_rs[2:0]	D 级 rs 转发选择信号	000：Rdata1 001：PC4@E+4 010：high@E 011：PC4@M+4

		100：ALUout@M
Z_D_rt[2:0]	D 级 rt 转发选择信号	000：Rdata2 001：PC4@E+4 010：high@E 011：PC4@M+4 100：ALUout@M
Z_E_rs[2:0]	E 级 rs 转发选择信号	000：data1 001：PC4@M+4 010：ALUout@M 011：M_WD
z_E_rt[2:0]	E 级 rt 转发选择信号	000：data2 001：PC4@M+4 010：ALUout@M 011：M_WD
Z_M_rt[1:0]	M 级 rt 转发选择信号	00：data2 01：M_WD

（3）控制信号真值表

	op	func	PCsel	z_rs_D	z_rt_D	z_rs_E	z_rt_E
addu	000000	100001	000	000	000	000	000
subu	000000	100011	000	000	000	000	000
ori	001101		000	000	000	000	000
lw	100011		000	000	000	000	000
sw	101011		000	000	000	000	000
lui	001111		000	000	000	000	000
beq	000100		001	000	000	000	000
j	000010		010	000	000	000	000
jal	000011		010	000	000	000	000
jr	000000	001000	011	000	000	000	000

	imm-sel	ALUop	MemRead	MemWrite	z_rt_M	M_A3	M_WD	RegWrite
addu	00	0010	0	0	00	01	01	1
subu	00	0110	0	0	00	01	01	1

ori	10	0001	0	0	00	00	01	1
lw	01	0010	1	0	00	00	00	1
sw	01	0010	0	1	00	00	01	0
lui	11	0010	0	0	00	00	01	1
beq	00	0110	0	0	00	00	01	0
j	00	0010	0	0	00	00	01	0
jal	00	0010	0	0	00	10	10	1
jr	00	0010	0	0	00	00	01	0

（4）暂停

IF/ID 当前指令			E 级寄存器				M 级寄存器
指令类型	源寄存器	T_use	lw-rt	addu-rd	subu-rd	ori-rt	lw-rt
lw	rs	1	2	1	1	1	1
sw	rs	1	2	1	1	1	1
sw	rt	2	2	1	1	1	1
addu	rs/rt	1	2	1	1	1	1
subu	rs/rt	1	2	1	1	1	1
ori	rs	1	2	1	1	1	1
beq	rs/rt	0	2	1	1	1	1
jr	rs	0	2	1	1	1	1

（5）转发

fcmp2		E 级	M 级	W 级	条件				
000									
001	beq|jr	jal_E			rt = 31	PC4@E+4			
010	beq|jr	lui_E			rt = rt_E	high@E			
011	beq|jr		jal_M		rt = 31		PC4@M+4		
100	beq|jr		addu|subu_M		rt = rd_M		ALUout@M		
	beq|jr		ori_M|lui_M		rt = rt_M		ALUout@M		
101	beq|jr			jal_W	rt = 31			M_WD	内部转发
	beq|jr			addu|subu_W	rt = rd_W			M_WD	内部转发
	beq|jr			ori|lui|lw_W	rt = rt_W			M_WD	内部转发

fcmp1		E 级	M 级	W 级	条件				
000									
001	beq|jr	jal_E			rs = 31	PC4@E+4			
010	beq|jr	lui_E			rs = rt_E	high@E			
011	beq|jr		jal_M		rs = 31		PC4@M+4		
100	beq|jr		addu|subu_M		rs = rd_M		ALUout@M		
	beq|jr		ori_M|lui_M		rs = rt_M		ALUout@M		
101	beq|jr			jal_W	rs = 31			M_WD	内部转发
	beq|jr			addu|subu_W	rs = rd_W			M_WD	内部转发
	beq|jr			ori|lui|lw_W	rs = rt_W			M_WD	内部转发

falu1		E 级	M 级	W 级	条件			
000								
001	addu/subu/ori/lw/sw		jal_M		rs = 31		PC4@M+4	
010	addu/subu/ori/lw/sw		addu_M|subu_M		rs = rd_M		ALUout@M	
	addu/subu/ori/lw/sw		ori_M|lui_M		rs = rt_M		ALUout@M	
011	addu/subu/ori/lw/sw			jal_W	rs = 31			M_WD
	addu/subu/ori/lw/sw			addu|subu_W	rs = rd_W			M_WD
	addu/subu/ori/lw/sw			ori|lui|lw_W	rs = rt_W			M_WD

falu2		E 级	M 级	W 级	条件			
000								
001	addu/subu/sw		jal_M		rt = 31		PC4@M+4	
010	addu/subu/sw		addu_M|subu_M		rt = rd_M		ALUout@M	
	addu/subu/sw		ori_M|lui_M		rt = rt_M		ALUout@M	
011	addu/subu/sw			jal_W	rt = 31			M_WD

	addu/subu/sw			addu|subu_W	rt = rd_W			M_WD
	addu/subu/sw			ori|lui|lw_W	rt = rt_W			M_WD

z_M_rt		E 级	M 级	W 级	条件			
00								
01	sw			jal_W	rt = 31			M_WD
	sw			addu|subu_W	rt = rd_W			M_WD
	sw			ori|lui|lw_W	rt = rt_W			M_WD

三、测试代码

（1）测试指令

```
# function test
# addu subu lui ori nop lw sw j jal jr beq
    ori     $7, $0, 16
    ori     $2, $1, 5
    ori     $3, $5, 1
    ori     $4, $8, 157
    ori     $0   $10, 9999
    ori     $5   $8, 0
    ori     $0, $11, 2333

    nop
    nop

    lui     $11 0xffff
    lui     $12 0x5678
    lui     $13 0
    lui     $0   57890

    nop
    nop

    addu    $31 $30 $7
    addu    $0   $7   $2
    addu    $30 $0   $3

    nop
    nop

    subu    $30 $27 $3
    subu    $0   $27 $8
    subu    $27 $0   $10

    nop
    nop
```

```
        sw      $2, -4($7)
        sw      $14, 0($0)
        sw      $15, 4($0)
        sw      $16, 8($0)

        nop
        nop

        lw      $17, -4($7)     #
        lw      $18, 4($0)      #
        lw      $19, 8($0)      #
        lw      $20, 12($0)     #

        nop
        nop
        nop
        nop

        j       beq_start

        nop
        nop

lb_back_1:

        jal     leaf
        nop

        j       end
        nop
        nop

beq_start:
        beq$31, $17, lb1
        nop
        addu    $27 $27 $2
        nop
        beq     $5 $5 lb1
        nop
        subu    $27 $27 $2
        nop

lb1:
        addu    $23 $22 $3
        beq     $1 $2 lb_back_1
        nop
        subu    $23 $22 $3
        nop
        beq     $24 $24 lb_back_1
        nop
```

```
        nop
        nop
    end:
        j       end             # jump to end
        nop


    leaf:
        addu    $8 $8 $8
        subu    $17 $17 $7
        jr      $ra             # jump to $ra
    nop
```

（2） 测试暂停

```
addu    $2, $0, 0xabcdef12
addu    $3, $0, 0xfd7894
addu    $4, $0, 0x95e2
addu    $5, $0, 4
addu    $6, $0, 16
addu    $7, $0, -12
addu    $8, $0, $0
addu    $9, $0, -9999

    jal     ED_1            # jump to ED_1 and save position to $ra
    addu    $3, $ra, $5
    j       next1           # jump to next1
    nop
ED_1:
    jr      $ra             # jump to $ra
    nop
next1:
    nop
    nop
    nop
    nop

    lui     $10, 0xfeee
    subu    $11, $10, $4
    nop
    nop
    nop
    nop

    jal     MD_1
    nop
    j       next2           # jump to next2
    nop
```

```
MD_1:
        ori         $12, $ra, 0xf8c5
        jr          $ra
        nop
next2:
        nop
        nop
        nop
        nop

        addu        $13, $14, $3
        subu        $14, $15, $4
        addu        $13, $14, $5
        nop
        nop
        nop
        nop

        addu        $16, $1, $2
        subu        $4, $16, $3
        sw          $4, -4($5)          #
        sw          $3,    4($5)        #
        nop
        nop
        nop
        nop
        nop

        lw          $17, -4($5)         #
        lui         $18, 5678
        beq         $7, $8, Oblivion
        addu        $18, $17, $3
        nop
        nop
        nop
        nop


        lw          $19, 4($5)          #
        lui         $11, 5678
        addu        $12, $19, $19
        nop
        nop
        nop
        nop
```

```
    lw      $20, 4($5)
    sw      $20, 0($5)
    nop
    nop
    nop
    nop

Oblivion:
    j       Oblivion                # jump to Oblivion
    nop
```

（3）转发测试

…….

四、思考题

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。

IF/ID 当前指令			E 级寄存器				M 级寄存器
指令类型	源寄存器	T_use	lw-rt	addu-rd	subu-rd	ori-rt	lw-rt
lw	rs	1	2	1	1	1	1
sw	rs	1	2	1	1	1	1
sw	rt	2	2	1	1	1	1
addu	rs/rt	1	2	1	1	1	1
subu	rs/rt	1	2	1	1	1	1
ori	rs	1	2	1	1	1	1
beq	rs/rt	0	2	1	1	1	1
jr	rs	0	2	1	1	1	1

如图所示，一共 15 种需要暂停的情况。其中 addu 和 subu 的行为完全相同，化简后有 12 种情况。依次测试即可。