2020年北航计组P7实验报告

冯张驰 19373573

一、CPU设计方案综述

（一）总体设计概述

本CPU为verilog实现的流水线MIPS - CPU，支持的MIPS-C4指令集为={LB、LBU、LH、LHU、LW、SB、SH、SW、ADD、ADDU、SUB、 SUBU、 MULT、 MULTU、 DIV、 DIVU、 SLL、 SRL、 SRA、SLLV、SRLV、SRAV、AND、OR、XOR、NOR、ADDI、ADDIU、ANDI、ORI、XORI、LUI、SLT、SLTI、SLTIU、SLTU、BEQ、BNE、BLEZ、BGTZ、BLTZ、BGEZ、J、JAL、JALR、JR、MFHI、MFLO、MTHI、MTLO、ERET、MFC0、MTC0}

处理器为五级流水线设计，支持普通的53条指令的执行和系统中断异常等操作。为了实现这些功能，CPU主要包含了IFU、GRF、ALU、DM、EXT、Controller、以及每级模块之间的流水寄存器模块、确定转发和暂停机制的处理冒险的模块单元等，还有CP0协处理器模块和系统桥，计时器等外设和内部沟通模块。

（二）关键模块定义

1.IFU

本部分分为3个模块，PC,pcCalc,IM

PC模块

端口定义

端口名	方向	描述
clk	I	时钟信号
reset	I	复位信号
next_pc [31:0]	I	下一时刻的PC数值
pc [31:0]	O	PC的当前值
en	I	使能信号

pcCalc模块

端口定义

端口名	方向	描述
zero	I	cmp的比较结果
branch[3:0]	I	当前指令是否为b型指令，以及是哪一类b型指令
imm[31:0]	I	32位地址（beq指令的立即数符号扩展后的数据）
npc	O	next-pc值
jump	I	当前指令是否为j/jal指令
rsData [31:0]	I	GRF模块RD1输出端口的值
imm26 [25:0]	I	当前周期内指令的低26位立即数
pc_D4[31:0]	I	D级的PC+4
pc[31:0]	I	F级的PC值
pcSrc	I	PC的选择信号

IM模块

端口定义

端口名	方向	描述
pc [31:0]	I	PC当前值
instr [31:0]	O	当前PC值对应的32位指令

IFU功能

1. 同步复位 复位信号有效时，PC被设置为**0x00003000。**
2. 取指令 根据**PC当前值**从**IM模块**中取出指令，并输出。
3. 计算下一条指令地址。
    - 当前指令是jal(pcSrc==1)，PC=PC[31:28]||imm26||0^2
    - 如果当前指令是beq指令（branch==1)且ALU输出的zero信号为1，则PC=PC+4+sign_extend(offset||0^2)
    - 当前指令是jr指令，PC=GPR[RS]
    - 以上三种情况之外，PC=PC+4；

2.F2D（流水寄存器1）

端口定义

端口名	方向	描述
PC_D[31:0]	O	D级PC值
PC_D4[31:0]	O	D级PC值+4
PC_D8[31:0]	O	D级PC值+8
instr_D[31:0]	O	D级指令
PC_F[31:0]	I	F级PC值
instr_F[31:0]	I	F级指令
clk	I	时钟信号
reset	I	异步复位信号
en	I	使能信号
BD_F	I	输入的D级BD信号
BD_D	O	输出的下一周期D级BD信号
excCode_F[4:0]	I	F级的异常信号
excCode_D[4:0]	O	D级的异常信号

功能定义

F/D级流水寄存器

3.GRF

端口定义

端口名	方向	描述
RA1[4:0]	I	RD1中数据的寄存器地址
RA2[4:0]	I	RD2中数据的寄存器地址
RD1[31:0]	O	RA1地址对应寄存器数值
RD2[31:0]	O	RA2地址对应寄存器数值
WA	I	写入数据的寄存器地址
WD	I	写入WA对应寄存器的数据
WE	I	写使能信号
clk	I	时钟信号
reset	I	异步复位信号

功能定义

1.复位。复位信号有效时所有寄存器的数值被设置为0x00000000.

2.读寄存器 根据当前输入的地址信号读出32位数据。

3.写寄存器。根据当前输入的地址信号，把输入的数据写入相应寄存器。

4.CMP

CMP端口定义

端口名	方向	描述
d1[31:0]	I	32位数据d1
d2[31:0]	I	32位数据d2
branch[3:0]	I	D级控制单元输出的b类型指令类别
equal	O	是否满足对应指令下的跳转条件，满足则为1

功能

比较两个端口的数值的关系，确定是否满足跳转条件

5.EXT

EXT端口定义

端口名	方向	描述
inm16[15:0]	I	16位立即数
extOp	I	扩展控制信号 0：符号扩展 1：无符号扩展
ext_D[31:0]	O	扩展后输出的数据值

功能

根据控制信号extOp进行相应的符号扩展操作

6.D2E（流水寄存器2）

端口名	方向	描述
instr_D[31:0]	I	D级指令
pc_D[31:0]	I	D级PC值
pc_D4[31:0]	I	D级PC值+4
pc_D8[31:0]	I	D级PC值+8
pc_E[31:0]	O	E级PC值
pc_E4[31:0]	O	E级PC值+4
pc_E8[31:0]	O	E级PC值+8
grf_RD1[31:0]	I	GRF端口RD1读的数值
grf_RD2[31:0]	I	GRF端口RD2读的数值
ext_D[31:0]	I	D级的32位扩展后立即数
instr_E[31:0]	O	E级指令
rs_E[31:0]	O	输出的rs的数值
rt_E[31:0]	O	输出的rt的数值
ext_E[31:0]	O	传给E级的32位扩展后立即数
clk	I	时钟信号
reset	I	异步复位信号
BD_D	I	D级BD值
BD_E	O	E级BD值
excCode_D[4:0]	I	D级异常信号
excCode_E[4:0]	O	E级异常信号

7.ALU

端口定义

端口名	方向	描述
A[31:0]	I	数据1
B[31:0]	I	数据2
aluCtrl[3:0]	I	控制信号0000：与 0001：或 0010：加 0011：减 0100：nor 0101：xor 0110：逻辑左移 0111：逻辑右移 1000：算数右移 1001：符号数相比小于置1 1010：无符号数相比，小于置1 1011：lui
result	O	输出的数据
zero	O	判断运算结果是否为0

功能定义

1.与：A&B

2.或：A|B

3.加：A+B

4.减：A-B

5.其他，如上表所示

8.E2M（流水寄存器3）

端口名	方向	描述
instr_E[31:0]	I	E级指令
pc_E[31:0]	I	E级PC值
pc_E4[31:0]	I	E级PC值+4
pc_E8[31:0]	I	E级PC值+8
rt_E[31:0]	I	E级保存的rt寄存器数值
aluRet_E[31:0]	I	ALU运算的结果
pc_M[31:0]	O	M级PC值
pc_M4[31:0]	O	M级PC值+4
pc_M8[31:0]	O	M级PC值+8
aluRet_M[31:0]	O	M级的ALU运算结果
instr_M[31:0]	O	M级指令
rt_M[31:0]	O	M级rt寄存器对应数值
clk	I	时钟信号
reset	I	同步复位信号

9.DM

端口定义

端口名	方向	描述
A[11:2]	I	数据的地址信号
WD[31:0]	I	当WE为高电平时，写入地址A的数据
RD[31:0]	O	当RE为高电平时，读出地址A的数据
RE	I	高电平时允许读数据
WE	I	高电平时允许写入数据
clk	I	时钟信号
reset	I	异步复位信号

功能定义

1. 复位：复位信号有效时，所有数据被设置为0x00000000。
2. 读：根据当前输入的寄存器地址读出数据
3. 写数据：根据输入地址，写入输入的数据。

store_judge模块

端口名	方向	描述
op[1:0]	I	写内存控制信号
A_1_0	I	地址最低两位
BE[3:0]	O	写使能信号

data_ext模块

端口名	方向	描述
A[1:0]	I	地址最低两位
op[2:0]	I	读内存的控制信号
Din[31:0]	I	内存中load的数据
Dout[31:0]	O	扩展的读取内存的输出数据

10.M2W（流水寄存器3）

端口名	方向	描述
instr_M[31:0]	I	M级指令
pc_M[31:0]	I	M级PC值
pc_M4[31:0]	I	M级PC值+4
pc_M8[31:0]	I	M级PC值+8
rt_M[31:0]	I	M级保存的rt寄存器数值
aluRet_M[31:0]	I	M级ALU运算的结果
pc_W[31:0]	O	W级PC值
pc_W4[31:0]	O	W级PC值+4
pc_W8[31:0]	O	W级PC值+8
aluRet_W[31:0]	O	W级的ALU运算结果
instr_W[31:0]	O	W级指令
rt_W[31:0]	O	W级rt寄存器对应数值
clk	I	时钟信号
reset	I	同步复位信号
RD_M[31:0]	I	M级读出的数据
RD_W[31:0]	O	W保存的M级读出的数据

11.Controller

端口定义

端口名	方向	描述
op[5:0]	I	6位opcode字段
funcode[5:0]	I	6位funcode字段
regDst[1:0]	O	GRF写地址控制信号 **2'b00**：选择rt端 **2'b01**：选择rd端 **2'b10**：选择31号寄存器
aluSrc	O	决定输入ALU模块的第二个输入端口数据是寄存器的输出值还是指令最低16位立即数扩展后的数值，**0，是前者， 1，是后者**
regWrite	O	GRF写控制
memRead	O	DM读控制
memWrite	O	DM写控制
memToReg[1:0]	O	GRF写数据选择： **2'b00**：写入ALU的计算结果 **2'b01**：写入DM的读出数据 **2'b10**：写入lui计算所得的数据结果 **2'b11**：写入当前PC加4后的值
extOp[1:0]	O	扩展控制信号 0：符号扩展 1：无符号扩展 2：将16位立即数加载到高16位，低16位补0
branch	O	是否为beq指令
aluCtrl	O	ALU的控制信号
jump	O	传递给pcCalc模块，决定next_pc的值
pcSrc	O	传递给pcCalc模块，选择是否为j类型的指令
shamt_or_rs	O	决定用于移位的运算符是shamt还是rs
mord	O	判断乘法还是除法
wHiLo	O	判断是写高位寄存器还是低位
weMD	O	判断是否需要写乘除寄存器
mdStart	O	判断当前回合乘除单元是否需要开始计算
extRdOp[2:0]	O	判断读出的数据扩展信号
storeOp[1:0]	O	写入的数据的控制
signmd	O	乘除是否是无符号计算
rHiLo	O	需要读还是写高、低位寄存器

12.CP0

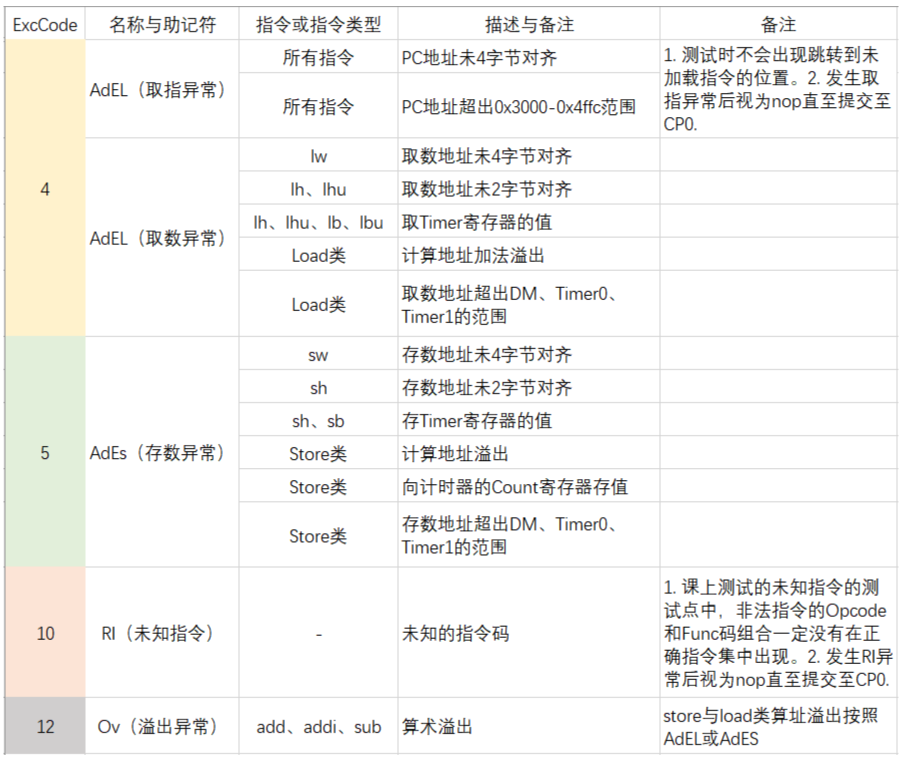
本CPU的设计放在M级进行检测。

其中delayslot信号，也就是BD信号，每周期在D级判断是否为跳转指令，并将结果传至F级进行流水，最终传至M级的CP0存到CAUSE寄存器中

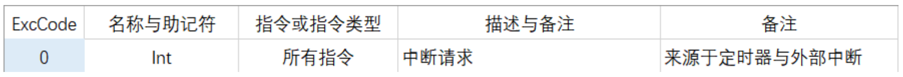
eret指令的检测放于D级，当D级为eret指令时，D级PC置为EPC的值，且清空延迟槽

EXLClr信号与W级的eret判断信号相连。

支持中断和异常的规范



ExcCode	名称与助记符	指令或指令类型	描述与备注	备注
4	AdEL（取指异常）	所有指令	PC地址未4字节对齐	1. 测试时不会出现跳转到未加载指令的位置。2. 发生取指异常后视为nop直至提交至CP0.
		所有指令	PC地址超出0x3000-0x4ffc范围	
	AdEL（取数异常）	lw	取数地址未4字节对齐	
		lh、lhu	取数地址未2字节对齐	
		lh、lhu、lb、lbu	取Timer寄存器的值	
		Load类	计算地址加法溢出	
		Load类	取数地址超出DM、Timer0、Timer1的范围	
5	AdEs（存数异常）	sw	存数地址未4字节对齐	
		sh	存数地址未2字节对齐	
		sh、sb	存Timer寄存器的值	
		Store类	计算地址溢出	
		Store类	向计时器的Count寄存器存值	
		Store类	存数地址超出DM、Timer0、Timer1的范围	
10	RI（未知指令）	-	未知的指令码	1. 课上测试的未知指令的测试点中，非法指令的Opcode和Func码组合一定没有在正确指令集中出现。2. 发生RI异常后视为nop直至提交至CP0.
12	Ov（溢出异常）	add、addi、sub	算术溢出	store与load类算址溢出按照AdEL或AdES



ExcCode	名称与助记符	指令或指令类型	描述与备注	备注
0	Int	所有指令	中断请求	来源于定时器与外部中断

端口定义

```
module CP0(
    input [4:0] A1,
    input [4:0] A2,
    input [31:0] Din,
    input [31:0] PC,
    input [6:2] ExcCode,
    input [5:0] HWInt,
    input We,
    input EXLSet,
    input EXLClr,
    input clk,
    input reset,
    output IntReq,
    output reg [31:0]  EPC,
    output [31:0] DOut,
    input delayslot
    );
```

13.Bridge

Praddr信号设置为M级的alu计算结果

各个信号沟通CPU和两个计时器

端口定义

```
module Bridge(
    input [31:2] PrAddr,
    input PrWE,
    input [31:0] PrWD,
    input [31:0] DEV0_RD,
    input [31:0] DEV1_RD,
    output [31:0] PrRD,
    output [31:2] DEV_Addr,
    output [31:0] DEV_WD,
    output WeDEV0,
    output WeDEV1
    );
```

14.MIPS.v

宏观PC设置为各级PC值的优先值，用一个mux确定选取哪一级的PC。

顶层模块，包含cpu/bridge/Timer1/Timer2等四个模块

```
module mips(
    input clk,
    input reset,
    input interrupt,
    output [31:0] addr
    );

    wire PrWE_cpu_o,PrWE_bri_o,WeDEV0,WeDEV1,IRQ_tc0,IRQ_tc1;
    wire [31:2] PrAddr_cpu_o,PrAddr_bri_o;
    wire [31:0] PrWD_cpu_o,PrWD_bri_o,TCRD0,TCRD1,PrRD;

    Bridge Bridge (.PrAddr(PrAddr_cpu_o), .PrWE(PrWE_cpu_o), .PrWD(PrWD_cpu_o),
.DEV0_RD(TCRD0), .DEV1_RD(TCRD0),
    .PrRD(PrRD), .DEV_Addr(PrAddr_bri_o), .DEV_WD(PrWD_bri_o), .WeDEV0(WeDEV0),
.WeDEV1(WeDEV1));

    TC Timer0(.clk(clk), .reset(reset), .Addr(PrAddr_bri_o), .WE(WeDEV0),
.Din(PrWD_bri_o), .Dout(TCRD0), .IRQ(IRQ_tc0));
    TC Timer1(.clk(clk), .reset(reset), .Addr(PrAddr_bri_o), .WE(WeDEV1),
.Din(PrWD_bri_o), .Dout(TCRD1), .IRQ(IRQ_tc1));

    cpu MyCPU (.reset(reset), .clk(clk), .interrupt(interrupt),
.PrAddr(PrAddr_cpu_o), .PrWD(PrWD_cpu_o),
    .PrRD(PrRD), .PrWE(PrWE_cpu_o),
.HWInt({3'b0,interrupt,IRQ_tc0,IRQ_tc1}),.PCAddr(addr));
endmodule
```

（三）控制器设计思路

设计方法

根据每个指令的opcode,funcode信号和对应的控制信号间的关系列出真值表，在 `always@(*)` 块中使用用 `case` 语句块对译码和输出控制信号过程进行描述。详细代码在上一部分已展出。

主控单元译码电路真值表

cal_r型指令	add	addu	sub	subu	AND	OR	NOR	XOR	sll	sllv	slt	sltu	sra	srav	srl	srlv
op[5:0]	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
func[5:0]	100000	100001	100010	100011	100100	100101	100111	100110	000000	000100	101010	101011	000011	000111	000010	000110
rt[5:0]	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx
regDst[1:0]	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
aluSrc	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
regWrite	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
memRead	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
memWrite	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
memToReg[2:0]	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
extOp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
branch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
aluCtrl[3:0]	0010	0010	0011	0011	0000	0001	0100	0101	0110	0110	1001	1010	1000	1000	0111	0111
jump	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pcSrc	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
extRdOp[2:0]	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000	000
storeOp[1:0]	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
mdStart	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mord	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
shamt_or_rs	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0
wHiLo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
weMD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

cal_i型指令	lui	ori	addi	addiu	andi	xori	slti	sltiu
op[5:0]	001111	001101	001000	001001	001100	001110	001010	001011
func[5:0]	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx
rt[5:0]	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx
regDst[1:0]	00	00	00	00	00	00	00	00
aluSrc	1	1	1	1	1	1	1	1
regWrite	1	1	1	1	1	1	1	1
memRead	0	0	0	0	0	0	0	0
memWrite	0	0	0	0	0	0	0	0
memToReg[2:0]	000	000	000	000	000	000	000	000
extOp	0	1	0	0	1	1	0	0
branch	0	0	0	0	0	0	0	0
aluCtrl[3:0]	1011	0001	0010	0010	0000	0101	1001	1001
jump	0	0	0	0	0	0	0	0
pcSrc	0	0	0	0	0	0	0	0
extRdOp[2:0]	000	000	000	000	000	000	000	000
storeOp[1:0]	00	00	00	00	00	00	00	00
mdStart	0	0	0	0	0	0	0	0
mord	0	0	0	0	0	0	0	0
shamt_or_rs	0	0	0	0	0	0	0	0
wHiLo	0	0	0	0	0	0	0	0
weMD	0	0	0	0	0	0	0	0

md类型指令	mfhi	mflo	mthi	mtlo	mult	multu	div	divu
op[5:0]	000000	000000	000000	000000	000000	000000	000000	000000
func[5:0]	010000	010010	010001	010011	011000	011001	011010	011011
rt[5:0]	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx
regDst[1:0]	01	01	00	00	00	00	00	00
aluSrc	x	x	x	x	0	0	0	0
regWrite	1	1	0	0	0	0	0	0
memRead	0	0	0	0	0	0	0	0
memWrite	0	0	0	0	0	0	0	0
memToReg[2:0]	011	100	000	000	000	000	000	000
extOp	0	0	0	0	0	0	0	0
branch	0	0	0	0	0	0	0	0
aluCtrl[3:0]	0000	0000	0000	0000	0000	0000	0000	0000
jump	0	0	0	0	0	0	0	0
pcSrc	0	0	0	0	0	0	0	0
extRdOp[2:0]	000	000	000	000	000	000	000	000
storeOp[1:0]	00	00	00	00	00	00	00	00
mdStart	0	0	0	0	1	1	1	1
mord	0	0	0	0	0	0	1	1
shamt_or_rs	0	0	0	0	0	0	0	0
wHiLo	0	0	0	1	0	0	0	0
weMD	0	0	1	1	0	0	0	0

store类型	sw	sb	sh
op[5:0]	101011	101000	101001
func[5:0]	xxxxxx	xxxxxx	xxxxxx
rt[5:0]	xxxxx	xxxxx	xxxxx
regDst[1:0]	xx	xx	xx
aluSrc	1	1	1
regWrite	0	0	0
memRead	0	0	0
memWrite	1	1	1
memToReg[2:0]	xxx	xxx	xxx
extOp	0	0	0
branch	0	0	0
aluCtrl[3:0]	0010	0010	0010
jump	0	0	0
pcSrc	0	0	0
extRdOp[2:0]	xxx	xxx	xxx
storeOp[1:0]	00	10	01
mdStart	0	0	0
mord	0	0	0
shamt_or_rs	0	0	0
wHiLo	0	0	0
weMD	0	0	1

load类型	lw	lb	lbu	lh	lhu
op[5:0]	100011	100000	100100	100001	100101
func[5:0]	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx
rt[5:0]	xxxxx	xxxxx	xxxxx	xxxxx	xxxxx
regDst[1:0]	00	00	00	00	00
aluSrc	1	1	1	1	1
regWrite	1	1	1	1	1
memRead	1	1	1	1	1
memWrite	0	0	0	0	0
memToReg[2:0]	001	001	001	001	001
extOp	0	0	0	0	0
branch	0	0	0	0	0
aluCtrl[3:0]	0010	0010	0010	0010	0010
jump	0	0	0	0	0
pcSrc	0	0	0	0	0
extRdOp[2:0]	000	010	001	100	011
storeOp[1:0]	xx	xx	xx	xx	xx
mdStart	0	0	0	0	0
mord	0	0	0	0	0
shamt_or_rs	0	0	0	0	0
wHiLo	0	0	0	1	0
weMD	0	0	1	1	0

b类型	**beq**	**bne**	**bgtz**	**blez**	**bgez**	**bltz**
op[5:0]	000100	000101	000111	000110	000001	000001
func[5:0]	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx
rt[4:0]	xxxxx	xxxxx	xxxxx	xxxxx	00001	00000
regDst[1:0]	xx	xx	xx	xx	xx	xx
aluSrc	x	x	x	x	x	x
regWrite	0	0	0	0	0	0
memRead	0	0	0	0	0	0
memWrite	0	0	0	0	0	0
memToReg[2:0]	xxx	xxx	xxx	xxx	xxx	xxx
extOp	x	x	x	x	x	x
branch[3:0]	0001	0010	0011	0100	0101	0110
aluCtrl[3:0]	xxxx	xxxx	xxxx	xxxx	xxxx	xxxx
jump	0	0	0	0	0	0
pcSrc	0	0	0	0	0	0
extRdOp[2:0]	xxx	xxx	xxx	xxx	xxx	xxx
storeOp[1:0]	xx	xx	xx	xx	xx	xx
mdStart	0	0	0	0	0	0
mord	0	0	0	0	0	0
shamt_or_rs	0	0	0	0	0	0
wHiLo	0	0	0	0	0	0
weMD	0	0	0	0	0	0

j类型	jal	j	jr	jalr
op[5:0]	000011	000010	000000	000000
func[5:0]	xxxxxx	xxxxxx	001000	001001
rt[4:0]	xxxxx	xxxxx	xxxxx	xxxxx
regDst[1:0]	10	xx	xx	01
aluSrc	x	x	x	x
regWrite	1	0	0	1
memRead	0	0	0	0
memWrite	0	0	0	0
memToReg[2:0]	010	xxx	xxx	010
extOp	x	x	x	x
branch[3:0]	0000	0000	0000	0000
aluCtrl[3:0]	xxxx	xxxx	xxxx	xxxx
jump	1	1	0	0
pcSrc	1	1	1	1
extRdOp[2:0]	xxx	xxx	xxx	xxx
storeOp[1:0]	xx	xx	xx	xx
mdStart	0	0	0	0
mord	0	0	0	0
shamt_or_rs	0	0	0	0
wHiLo	0	0	0	0
weMD	0	0	0	0

对于异常中断类型指令，此处只列出部分有用的需要设置为高电平的信号

CP0类	mtc0	mfc0	eret
op[5:0]	010000	010000	010000
fun[5:0]	xxxxxx	xxxxxx	011000
rs[4:0]	00100	00000	xxxxx
wecp0	1	0	0
memToReg[2:0]	0	4	0
regWrite	0	1	0

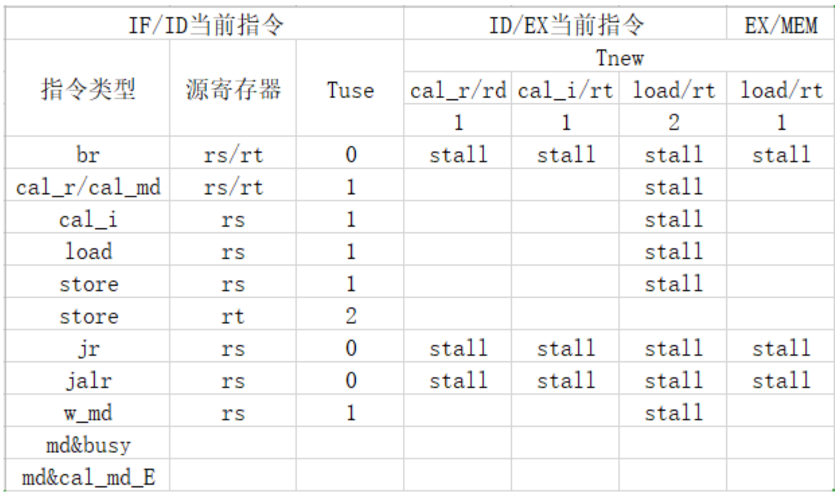
eret从controller里传出信号，根据D级是否为eret指令判断F级PC是否返回epc

（四）冒险单元设计

端口定义

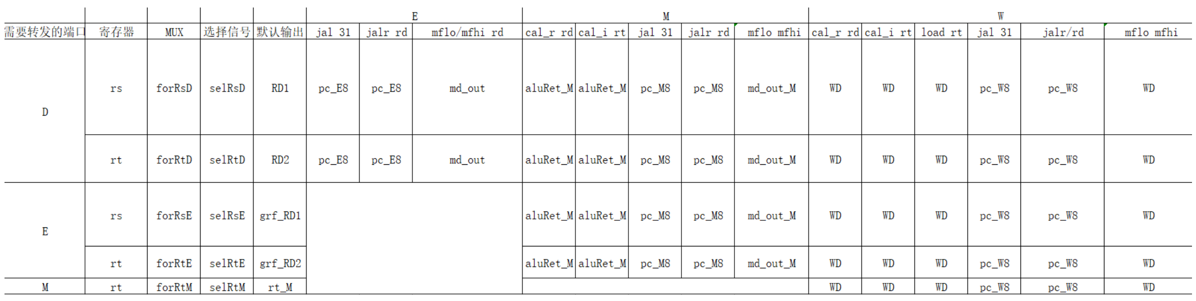
端口名	方向	描述
instr_D[31:0]	I	D级的31位指令
instr_E[31:0]	I	E级的31位指令
instr_M[31:0]	I	M级的31位指令
instr_W[31:0]	I	W级的31位指令
stall_F	O	F级暂停信号
stall_D	O	D级暂停信号
flush_E	O	E级暂停信号
selRsD[2:0]	O	forRsD选择信号
selRtD[2:0]	O	forRtD选择信号
selRsE[2:0]	O	forRsE选择信号
selRtE[2:0]	O	forRtE选择信号
selRtM[2:0]	O	forRtM选择信号
busy	I	乘除单元是否正在工作

冒险信号控制表格



IF/ID当前指令			ID/EX当前指令			EX/MEM
			Tnew			
指令类型	源寄存器	Tuse	cal_r/rd	cal_i/rt	load/rt	load/rt
			1	1	2	1
br	rs/rt	0	stall	stall	stall	stall
cal_r/cal_md	rs/rt	1			stall	
cal_i	rs	1			stall	
load	rs	1			stall	
store	rs	1			stall	
store	rt	2				
jr	rs	0	stall	stall	stall	stall
jalr	rs	0	stall	stall	stall	stall
w_md	rs	1			stall	
md&busy						
md&cal_md_E						

暂停控制信号表



					E			M					W					
需要转发的端口	寄存器	MUX	选择信号	默认输出	jal 31	jalr rd	mflo/mfhi rd	cal_r rd	cal_i rt	jal 31	jalr rd	mflo mfhi	cal_r rd	cal_i rt	load rt	jal 31	jalr/rd	mflo mfhi
D	rs	forRsD	selRsD	RD1	pc_E8	pc_E8	md_out	aluRet_M	aluRet_M	pc_M8	pc_M8	md_out_M	WD	WD	WD	pc_W8	pc_W8	WD
	rt	forRtD	selRtD	RD2	pc_E8	pc_E8	md_out	aluRet_M	aluRet_M	pc_M8	pc_M8	md_out_M	WD	WD	WD	pc_W8	pc_W8	WD
E	rs	forRsE	selRsE	grf_RD1				aluRet_M	aluRet_M	pc_M8	pc_M8	md_out_M	WD	WD	WD	pc_W8	pc_W8	WD
	rt	forRtE	selRtE	grf_RD2				aluRet_M	aluRet_M	pc_M8	pc_M8	md_out_M	WD	WD	WD	pc_W8	pc_W8	WD
M	rt	forRtM	selRtM	rt_M									WD	WD	WD	pc_W8	pc_W8	WD

转发控制信号表

冒险单元代码

```
`timescale 1ns / 1ps
`include "constants.v"
module hazard_Unit(
    input [31:0] instr_D,
    input [31:0] instr_E,
    input [31:0] instr_M,
    input [31:0] instr_W,
    output stall_F,
    output stall_D,
    output flush_E,
    output [3:0] selRsD,
    output [3:0] selRtD,
    output [3:0] selRsE,
    output [3:0] selRtE,
    output [3:0] selRtM,
    input busy
    );

    wire
D_b,D_cal_r,D_cal_i,D_load,D_store,D_jr,D_jal,D_jalr,D_cal_md,D_md,D_r_md,D_w_md
,D_mtc0,D_mfc0;
    instr_class D_class (.instr(instr_D), .jal(D_jal), .cal_i(D_cal_i),
.cal_r(D_cal_r), .load(D_load), .b(D_b), .jr(D_jr),
.store(D_store),.jalr(D_jalr),.md(D_md),.cal_md(D_cal_md),.r_md(D_r_md),.w_md(D_
w_md),.mfc0(D_mfc0),.mtc0(D_mtc0));
    wire
E_b,E_cal_r,E_cal_i,E_loaE,E_store,E_jr,E_jal,E_jalr,E_cal_md,E_md,E_r_md,E_mtc0
,E_mfc0;
    instr_class E_class (.instr(instr_E), .jal(E_jal), .cal_i(E_cal_i),
.cal_r(E_cal_r), .load(E_load), .b(E_b), .jr(E_jr),
.store(E_store),.jalr(E_jalr),.md(E_md),.cal_md(E_cal_md),.r_md(E_r_md),.mfc0(E_
mfc0),.mtc0(E_mtc0));
    wire
M_b,M_cal_r,M_cal_i,M_load,M_store,M_jr,M_jal,M_jalr,M_cal_md,M_md,M_r_md,M_mtc0
,M_mfc0;
    instr_class M_class (.instr(instr_M), .jal(M_jal), .cal_i(M_cal_i),
.cal_r(M_cal_r), .load(M_load), .b(M_b), .jr(M_jr),
.store(M_store),.jalr(M_jalr),.md(M_md),.cal_md(M_cal_md),.r_md(M_r_md),.mfc0(M_
mfc0),.mtc0(M_mtc0));
    wire
W_b,W_cal_r,W_cal_i,W_load,W_store,W_jr,W_jal,W_jalr,W_cal_md,W_md,W_r_md,W_w_md
,W_mtc0,W_mfc0;
    instr_class W_class (.instr(instr_W), .jal(W_jal), .cal_i(W_cal_i),
.cal_r(W_cal_r), .load(W_load), .b(W_b), .jr(W_jr),
.store(W_store),.jalr(W_jalr),.md(W_md),.cal_md(W_cal_md),.r_md(W_r_md),.w_md(W_
w_md),.mfc0(W_mfc0),.mtc0(W_mtc0));

    wire stallRt,stallRs,stall,stallMd;

    wire [4:0] rs_D,rs_E,rt_D,rt_E,rs_M,rt_M,rd_M,rs_W,rt_W,rd_W,rd_E;
    assign rs_D = instr_D[`rs];
    assign rs_E = instr_E[`rs];
    assign rd_E = instr_E[`rd];
    assign rt_D = instr_D[`rt];
    assign rt_E = instr_E[`rt];
    assign rs_M = instr_M[`rs];
    assign rt_M = instr_M[`rt];
    assign rd_M = instr_M[`rd];
    assign rs_W = instr_W[`rs];
    assign rt_W = instr_W[`rt];
    assign rd_W = instr_W[`rd];

    //stall logic
    assign stallRt = (D_b&&(E_load||E_mfc0)&&rt_E==rt_D&&rt_D)||
(D_b&&E_cal_i&&rt_E==rt_D&&rt_D)||(D_b&&E_cal_r&&rd_E==rt_D&&rt_D)||(D_b&&
(M_load||M_mfc0)&&rt_M==rt_D&&rt_D)||
                     (D_cal_r&&(E_load||E_mfc0)&&rt_D==rt_E&&rt_D||D_cal_md&&
(E_load||E_mfc0)&&rt_D==rt_E&&rt_D);
    assign stallRs = (D_b&&(E_load||E_mfc0)&&rt_E==rs_D&&rs_D)||
(D_b&&E_cal_i&&rt_E==rs_D&&rs_D)||(D_b&&E_cal_r&&rd_E==rs_D&&rs_D)||(D_b&&
(M_load||M_mfc0)&&rt_M==rs_D&&rs_D)||
                     (D_cal_r&&(E_load||E_mfc0)&&rs_D==rt_E&&rs_D)||(D_cal_i&&
(E_load||E_mfc0)&&rs_D==rt_E&&rs_D)||(D_load&&
(E_load||E_mfc0)&&rs_D==rt_E&&rs_D)||(D_store&&
(E_load||E_mfc0)&&rs_D==rt_E&&rs_D)||(D_cal_md&&
(E_load||E_mfc0)&&rs_D==rt_E&&rs_D)||
                     (D_jr&&(E_load||E_mfc0)&&rt_E==rs_D&&rs_D)||
(D_jr&&E_cal_i&&rt_E==rs_D&&rs_D)||(D_jr&&E_cal_r&&rd_E==rs_D&&rs_D)||(D_jr&&
(M_load||M_mfc0)&&rt_M==rs_D&&rs_D)||
                     ((D_jalr||D_w_md)&&(E_load||E_mfc0)&&rt_E==rs_D&&rs_D)||
(D_jalr&&E_cal_i&&rt_E==rs_D&&rs_D)||(D_jalr&&E_cal_r&&rd_E==rs_D&&rs_D)||
(D_jalr&&(M_load||M_mfc0)&&rt_M==rs_D&&rs_D);
    assign stallMd=(D_md&&busy)||(D_md&&E_cal_md);
    //assign stallcp0=1;
    assign stall = stallRs||stallRt||stallMd;
    assign stall_F = ~stall;
    assign stall_D = ~stall;
    assign flush_E = stall;

    //forward logic
    assign selRsD = (rs_D==31&&E_jal&&rs_D||rs_D==rd_E&&E_jalr&&rs_D)?1:
                    (rs_D==rd_E&&E_r_md&&rs_D)?6:
                    (rs_D==31&&M_jal&&rs_D||rs_D==rd_M&&M_jalr&&rs_D)?2:
(rs_D==rt_M&&(M_cal_i)&&rs_D)||(rs_D==rd_M&&M_cal_r&&rs_D)?3:
                    (rs_D==rd_M&&M_r_md&&rs_D)?7:
                    (rs_D==31&&W_jal&&rs_D||rs_D==rd_W&&W_jalr&&rs_D)?4:
(rs_D==rt_W&&(W_cal_i||W_load||W_mfc0)&&rs_D)||(rs_D==rd_W&&W_cal_r&&rs_D)||
(rs_D==rd_W&&W_r_md&&rs_D)?5:0;

    assign selRtD = (rt_D==31&&E_jal&&rt_D||rt_D==rd_E&&E_jalr&&rt_D)?1:
                    (rt_D==rd_E&&E_r_md&&rt_D)?6:
                    (rt_D==31&&M_jal&&rt_D||rt_D==rd_M&&M_jalr&&rt_D)?2:
(rt_D==rt_M&&(M_cal_i)&&rt_D)||(rt_D==rd_M&&M_cal_r&&rt_D)?3:
                    (rt_D==rd_M&&M_r_md&&rt_D)?7:
                    (rt_D==31&&W_jal&&rt_D||rt_D==rd_W&&W_jalr&&rt_D)?4:
(rt_D==rt_W&&(W_cal_i||W_load||W_mfc0)&&rt_D)||(rt_D==rd_W&&W_cal_r&&rt_D)||
(rt_D==rd_W&&W_r_md&&rt_D)?5:0;

    assign selRsE = (rs_E==31&&M_jal&&rs_E||rs_E==rd_M&&M_jalr&&rs_E)?1:
(rs_E==rt_M&&(M_cal_i)&&rs_E)||(rs_E==rd_M&&M_cal_r&&rs_E)?2:
```

```
                    (rs_E==rd_M&&M_r_md&&rs_E)?5:
                    (rs_E==31&&W_jal&&rs_E||rs_E==rd_W&&W_jalr&&rs_E)?3:
(rs_E==rt_W&&(W_cal_i||W_load||W_mfc0)&&rs_E)||(rs_E==rd_W&&W_cal_r&&rs_E)||
(rs_E==rd_W&&W_r_md&&rs_E)?4:0;

    assign selRtE = (rt_E==31&&M_jal&&rt_E||rt_E==rd_M&&M_jalr&&rt_E)?1:
(rt_E==rt_M&&(M_cal_i)&&rt_E)||(rt_E==rd_M&&M_cal_r&&rt_E)?2:
                    (rt_E==rd_M&&M_r_md&&rt_E)?5:
                    (rt_E==31&&W_jal&&rt_E||rt_E==rd_W&&W_jalr&&rt_E)?3:
(rt_E==rt_W&&(W_cal_i||W_load||W_mfc0)&&rt_E)||(rt_E==rd_W&&W_cal_r&&rt_E)||
(rt_E==rd_W&&W_r_md&&rt_E)?4:0;

    assign selRtM = (rt_M==31&&W_jal&&rt_M||rt_M==rd_W&&W_jalr&&rt_M)?1:
(rt_M==rt_W&&(W_cal_i||W_load||W_mfc0)&&rt_M)||(rt_M==rd_W&&W_cal_r&&rt_M)||
(rt_M==rd_W&&W_r_md&&rt_M)?2:0;
/*  wire test_ans;
    assign test_ans=(rs_E==rt_W&&W_cal_i&&rs_E)||(rs_E==rd_W&&W_cal_r&&rs_E);*/
endmodule
```

二、测试方案

典型测试样例

暂停机制测试

```
#D instr and E instr
#beq cal_r
ori $1 $0 12
addu $1 $1 $0
beq $1 $0 next
ori $4 $0 1234
next:addu $1 $1 $1
#beq cal_i
ori $1 $0 12
beq $1 $0 next
ori $4 $0 1234
next:addu $1 $1 $1
#beq load
ori $1 $0 12
sw $1 0($0)
lw $2 0($0)
beq $2 $0 next
ori $4 $0 1234
next:addu $1 $1 $1
#cal_r load
ori $1 $0 12
sw $1 0($0)
lw $2 0($0)
subu $3 $2 $1
#cal_i load
ori $1 $0 12
sw $1 0($0)
lw $2 0($0)
ori $3 $2 0
#load load
```

```
ori $1 $0 12
sw $1 0($0)
lw $2 0($0)
lw $3 0($2)
#store load
ori $1 $0 12
sw $1 0($0)
lw $2 0($0)
sw $2 0($2)
#jr cal_r
ori $1 $0 0x300c
addu $1 $1 $0
jr $1
ori $4 $0 1234
next:addu $1 $1 $1
#jr cal_i
ori $1 $0 0x300c4
jr $1
ori $4 $0 1234
next:addu $1 $1 $1
#jr load
ori $1 $0 0x3014
sw $1 0($0)
lw $2 0($0)
jr $2
ori $4 $0 1234
next:addu $1 $1 $1

#D instr and M instr
#beq load
ori $1 $0 0x3018
sw $1 0($0)
lw $2 0($0)
ori $5 1234
beq $2 $0 next
ori $4 $0 1234
next:addu $1 $1 $1
#beq load
ori $1 $0 0x3018
sw $1 0($0)
lw $2 0($0)
ori $5 1234
jr $2
ori $4 $0 1234
next:addu $1 $1 $1
```

转发机制测试

```
#whole forward test
ori $1 $0 1244
ori $3 $0 1244
addu $4 $1 $3
addu $5 $3 $3
sw $5 0($3)
lw $7 0($3)
jal 0x3024
addu $2 $31 $0
```

```
j 0x302c
ori $3 $0 12222 #0x3024
jr $31
jal 0x3038
ori $9 $31 0
j 0x3040
ori $3 $0 12222 #0x3038
jr $31
```

综合测试

自动测试脚本

指令生成代码

建立了一个指令类，mipscode

支持产生纯随机指令，以及指定的较强冒险组合指令的mips asm代码

```
import random
import os
testnum=3
mipsDict={}
mipsDict['nop']=['nop']
mipsDict['cal_r']=['add', 'addu', 'sub', 'subu', 'and', 'or', 'nor', 'xor',
 'sllv', 'slt', 'sltu',  'srav',  'srlv']
mipsDict['cal_i']=['ori', 'addi', 'addiu', 'andi', 'xori']
mipsDict['lui']=["lui"]
mipsDict['jump']=["j","jal"]
mipsDict['jr']=["jr"]
mipsDict['jalr']=["jalr"]
mipsDict['b']=['beq', 'bne']
mipsDict['b1']=['bgtz', 'blez', 'bgez', 'bltz']
mipsDict['load']=['lw', 'lb', 'lbu', 'lh', 'lhu']
mipsDict['store']=["sw","sb","sh"]
mipsDict['cal_md']=['mult', 'multu', 'div', 'divu']
mipsDict['r_md']=['mfhi', 'mflo']
mipsDict['w_md']=['mthi','mtlo']
mipsDict['eret']=['eret']
mipsDict['cp0']=['mfc0','mtc0']
mipsDict['s']=['sll','sra','srl']
mipsDict['s1']=['slti', 'sltiu']
n2lei=
['nop','cal_i','cal_r','load','store','lui','jr','jalr','jump','cal_md','w_md','
r_md','b1','cp0','b','s','s1','eret']
lei2n={ "nop":0, "cal_i":1, "cal_r":2, "load":3, "store":4, "lui":5, "jr":6,
"jalr":7, "jump":8,
        "cal_md":9, "w_md":10, "r_md":11, "b1":12, "cp0":13, "b":14, "s":15,
"s1":16, "eret":17}
class mipscode:
    instr=None
    rs=1
    rt=1
    rd=1
    lei=0
    imm16=0
    cp0=0
    label=""

    mipsDict={}
    mipsDict['nop']=['nop']
    mipsDict['cal_r']=['add', 'addu', 'sub', 'subu', 'and', 'or', 'nor', 'xor',
 'sllv', 'slt', 'sltu',  'srav',  'srlv']
    mipsDict['cal_i']=['ori', 'addi', 'addiu', 'andi', 'xori']
    mipsDict['lui']=["lui"]
    mipsDict['jump']=["j","jal"]
    mipsDict['jr']=["jr"]
    mipsDict['jalr']=["jalr"]
    mipsDict['b']=['beq', 'bne']
    mipsDict['b1']=['bgtz', 'blez', 'bgez', 'bltz']
    mipsDict['load']=['lw', 'lb', 'lbu', 'lh', 'lhu']
    mipsDict['store']=["sw","sb","sh"]
    mipsDict['cal_md']=['mult', 'multu', 'div', 'divu']
    mipsDict['r_md']=['mfhi', 'mflo']
    mipsDict['w_md']=['mthi','mtlo']
    mipsDict['eret']=['eret']
    mipsDict['cp0']=['mfc0','mtc0']
    mipsDict['s']=['sll','sra','srl']
    mipsDict['s1']=['slti', 'sltiu']
    n2lei=
['nop','cal_i','cal_r','load','store','lui','jr','jalr','jump','cal_md','w_md','
r_md','b1','cp0','b','s','s1','eret']
    lei2n={ "nop":0, "cal_i":1, "cal_r":2, "load":3, "store":4, "lui":5, "jr":6,
"jalr":7, "jump":8,
            "cal_md":9, "w_md":10, "r_md":11, "b1":12, "cp0":13, "b":14, "s":15,
"s1":16, "eret":17}
    def __init__(self,lei,rs=1,rt=1,rd=1,imm16=0,instr=None,cp0=0,label=None):
        self.lei=lei
        self.rs=rs
        self.rt=rt
        self.rd=rd
        self.imm16=imm16
        self.instr=instr
        self.cp0=cp0
        self.label=label
    def getMipsCode(self):
        if self.lei==0: #nop
            return "nop"
        elif(self.lei==1): #cal_i
            return "%s $%d $%d %d" % (self.instr,self.rt,self.rs,self.imm16)
        elif self.lei==2: #cal_r
            return "%s $%d $%d $%d" % (self.instr,self.rd,self.rt,self.rs)
        elif self.lei==3: #load
            return "%s $%d %d($%d)"%(self.instr,self.rt,self.imm16,self.rs)
        elif self.lei==4: #store
            return "%s $%d %d($%d)"%(self.instr,self.rt,self.imm16,self.rs)
        elif self.lei==5: #lui
            return "lui $%d %d"%(self.rt,self.imm16)
        elif self.lei==6: #jr
            return "%s $%d"%("jr",self.rs)
        elif self.lei==7: #jalr
            return "%s $%d $%d"%("jalr",self.rd,self.rs)
        elif self.lei==8: #jump
            return "%s %s" % (self.instr,self.label)
        elif self.lei==9: #cal_md
```

```
            return "%s $%d $%d"%(self.instr,self.rs,self.rt)
        elif self.lei==10: #w_md
            return "%s $%d"%(self.instr,self.rd)
        elif self.lei==11: #r_md
            return "%s $%d"%(self.instr,self.rs)
        elif self.lei==12: #b1
            return "%s $%d %s"%(self.instr,self.rs,self.label)
        elif self.lei==13: #mf/mt c0
            return "%s $%d $%d"%(self.instr,self.rt,self.cp0)
        elif self.lei==14: #b
            return "%s $%d $%d %s"%(self.instr,self.rs,self.rt,self.label)
        elif self.lei==15: #s
            return "%s $%d $%d %d" %
(self.instr,self.rt,self.rs,self.imm16/(2**11))
        elif self.lei==16: #s1
            return "%s $%d $%d %d" % (self.instr,self.rt,self.rs,self.imm16/4)
        elif self.lei==17: #eret
            return "%s"%("eret")
        else:
            return  "nop"
    def getInstr(self):
        dict1=self.mipsDict
        if(self.instr):
            for x in dict1:
                if(self.instr in dict1[x]):
                    self.lei=self.n2lei.index(self.instr)
                    break
                else:
                    self.lei=0
        else:
            name=self.n2lei[self.lei]
            self.instr=dict1[name][random.randint(0,len(dict1[name])-1)]
    def getMipsStr(self):
        self.getInstr()
        return self.getMipsCode()
def exhand():
    return '''
.ktext 0x4180
mfc0 $28 $14
ori $29 $0 0x4ffc
bgt $28 $29 next
ori $29 $0 0x3000
blt $28 $29 next
normal:nop
eret
nop
next:
la $29 label1
mtc0 $29 $14
eret
'''
def hazardTestMake(lei1,src):
    l=[random.randint(0,31) for i in range(3)]
    for i in l:
        if(l==28):
            l=27
        elif l==29:
            l=30
    m=len(leil)
    s=""
    for x in leil:
        s=s+"+"+x
    #print(s)
    fname=src+"\\Test"+s+".asm"
    leil=[lei2n[leil[i]] if isinstance(leil[i],str) else leil[i] for i in
range(m)]
    n=20
    path=os.path.dirname(os.path.realpath(__file__))
    os.chdir(path)
    with open(fname,"w") as f:
        for j in range(m):
            f.write("label%d:nop\n"%j)
            for i in range(n):
                a=random.randint(0,m-1)
                num=[random.randint(0,2) for i in range(3)]
                imm16=random.randint(0,0xfff)

 f.write(mipscode(leil[a],l[num[0]],l[num[1]],l[num[2]],imm16,cp0=random.randint
(12,15),label="label"+str(random.randint(0,m-1))).getMipsStr()+"\n")
        f.write(exhand())


def get_one_testpoint(file_name):
    path=os.path.dirname(os.path.realpath(__file__))
    os.chdir(path)
    with open(file_name,"w") as f:
        f.write("ori $gp $0 0"+"\n")
        f.write("ori $sp $0 0"+"\n")
        f.write("mtc0 $0 $15\n")
        n=10
        for i in range(0,n):
            f.write("label%d:nop\n"%(i))
            l=[]
            for i in range(5):
                l.append(random.randint(0,31))
            for i in l:
                if(i==28):
                    i=27
                elif i==29:
                    i=30
            imm16=random.randint(0,0xffff)
            lei1=0
            for i in range(50):
                lei=random.randint(1,16)
                if((lei1==6 or lei1==7 or lei1==8 or lei1==12 or lei1==14) and
(lei==6 or lei==7 or lei==8 or lei==12 or lei==14)):
                    lei=1

 a=mipscode(lei,l[random.randint(0,4)],l[random.randint(0,4)],l[random.randint(0
,4)],imm16,cp0=random.randint(12,15),label="label"+str(random.randint(0,n-1)))
                f.write(a.getMipsStr()+"\n")
                lei1=lei
        f.write(exhand())
def getrandompoint(num,src):
    for i in range(num):
        pointname="%s//testpoint%d.asm"%(src,i+1)
        get_one_testpoint(pointname)
#main()
def gethazardpoint(l,src):
    if(l==None):
        for i in mipsDict:
            for j in mipsDict:
            # print(i,j)
                hazardTestMake([i,j],src)
    else:
        hazardTestMake(l,src)
getrandompoint(10,"test_fzc")
gethazardpoint(["load","store"],"test1")
```

生成的随机mips代码举例

```
ori $gp $0 0
ori $sp $0 0
mtc0 $0 $15
label0:nop
sll $19 $4 27
beq $16 $4 label8
sra $19 $4 27
sh $4 57173($9)
beq $4 $4 label9
mfc0 $9 $15
bne $16 $16 label3
andi $19 $4 57173
jalr $4 $19
mflo $19
sra $4 $9 27
jalr $9 $9
lbu $4 57173($9)
lui $4 57173
nor $4 $9 $16
jr $9
sll $4 $4 27
mtlo $4
beq $16 $19 label1
srl $9 $16 27
bltz $4 label1
sll $19 $19 27
jr $4
xori $16 $9 57173
lh $4 57173($16)
jalr $4 $4
mtc0 $4 $13
sll $16 $16 27
lb $19 57173($9)
mult $4 $4
sll $4 $4 27
andi $4 $4 57173
mthi $4
sb $19 57173($4)
jal label5
xori $4 $9 57173
sw $4 57173($4)
slti $9 $4 14293
```

```
jalr $4 $9
lw $9 57173($19)
j label6
sw $9 57173($19)
sh $9 57173($19)
bne $4 $16 label3
addiu $16 $4 57173
sll $19 $16 27
mult $4 $9
srl $16 $4 27
jalr $16 $4
mfc0 $9 $15
label1:nop
sltu $14 $29 $14
jalr $14 $18
srlv $18 $29 $18
mfhi $14
mfc0 $14 $12
mthi $14
andi $18 $14 63837
bne $1 $1 label4
sw $14 63837($18)
blez $18 label4
mfhi $18
lui $0 63837
ori $18 $14 63837
andi $18 $1 63837
beq $18 $29 label2
slti $18 $0 15959
lh $29 63837($29)
srl $18 $18 31
jr $1
mfhi $18
bgtz $1 label9
addi $1 $0 63837
mult $0 $0
sw $14 63837($0)
bne $14 $0 label2
sltiu $14 $14 15959
addi $0 $1 63837
jalr $0 $0
slti $1 $0 15959
sw $1 63837($1)
slti $18 $0 15959
mtlo $29
div $29 $0
mtc0 $18 $13
mtc0 $29 $14
mtlo $0
jalr $18 $1
mthi $0
jr $29
lui $1 63837
j label2
mfhi $14
mult $14 $29
div $18 $14
beq $29 $14 label4
```

```
addi $14 $0 63837
lbu $18 63837($14)
andi $0 $18 63837
nor $18 $14 $14
srl $1 $29 31
label2:nop
jr $6
lui $6 53627
sll $6 $6 26
jal label0
addi $6 $6 53627
sltiu $4 $8 13406
mfc0 $4 $12
multu $26 $6
mfhi $6
add $26 $8 $26
lui $6 53627
jr $8
div $8 $26
or $6 $6 $8
slti $26 $4 13406
addi $8 $26 53627
bne $26 $26 label8
lbu $6 53627($26)
divu $26 $8
mfhi $8
mfhi $8
blez $26 label5
lbu $6 53627($26)
jr $8
slti $6 $8 13406
blez $26 label8
sltiu $6 $6 13406
blez $6 label5
addiu $8 $6 53627
lbu $6 53627($26)
slti $26 $26 13406
sltiu $8 $6 13406
mtc0 $4 $15
jalr $26 $26
sh $8 53627($26)
jal label9
addu $26 $26 $8
beq $4 $8 label5
sh $6 53627($8)
sb $26 53627($4)
sll $4 $26 26
sb $6 53627($4)
lb $4 53627($8)
mtlo $26
ori $8 $6 53627
lui $6 53627
slti $6 $6 13406
mflo $4
mfc0 $26 $13
jal label7
label3:nop
beq $6 $18 label3
```

```
mthi $11
lui $6 29650
bne $11 $6 label0
sll $18 $6 14
blez $14 label8
xori $18 $6 29650
addiu $6 $18 29650
sll $11 $6 14
mfc0 $14 $15
bgtz $0 label6
sltiu $18 $11 7412
beq $0 $14 label0
srl $0 $14 14
jalr $0 $14
lui $14 29650
sw $6 29650($18)
multu $14 $11
lui $0 29650
j label1
andi $11 $18 29650
jr $18
ori $11 $0 29650
jr $6
mtlo $6
blez $0 label1
mthi $18
beq $6 $14 label1
xori $18 $14 29650
sltiu $0 $6 7412
lui $11 29650
jal label5
sltiu $6 $11 7412
nor $18 $6 $6
lhu $0 29650($18)
jalr $14 $18
slt $0 $0 $11
sltiu $0 $14 7412
mfc0 $11 $14
beq $0 $14 label3
andi $18 $18 29650
beq $18 $18 label3
ori $6 $18 29650
mfhi $0
lui $18 29650
divu $18 $6
sltiu $14 $14 7412
subu $6 $14 $11
bgez $14 label8
lbu $0 29650($18)
label4:nop
mthi $11
j label7
sllv $11 $16 $5
slti $18 $11 14629
srlv $18 $1 $11
jalr $18 $5
lhu $16 58518($18)
mthi $18
```

```
mfhi $1
mtlo $5
jalr $11 $11
add $5 $11 $1
multu $1 $5
jalr $11 $16
andi $11 $11 58518
jal label6
mfc0 $16 $15
sh $1 58518($11)
mtlo $1
mflo $18
lui $5 58518
bgtz $16 label1
sltiu $16 $5 14629
bne $18 $18 label9
addi $1 $11 58518
mfhi $11
bgtz $18 label3
ori $11 $16 58518
sb $1 58518($1)
mflo $5
ori $16 $5 58518
bltz $18 label4
mtc0 $1 $15
sra $11 $5 28
bne $16 $1 label3
add $1 $1 $1
jalr $1 $1
ori $16 $18 58518
addiu $18 $16 58518
sltiu $5 $1 14629
or $18 $18 $5
multu $1 $18
jalr $11 $5
xori $1 $18 58518
ori $5 $11 58518
jal label7
ori $16 $11 58518
mtc0 $16 $12
mflo $18
lb $1 58518($18)
label5:nop
mtlo $23
slti $23 $23 1693
ori $12 $23 6774
mtlo $18
mfhi $3
mthi $3
lui $23 6774
mflo $12
srav $23 $12 $3
bgtz $18 label5
lui $12 6774
jr $18
lhu $8 6774($18)
sh $23 6774($23)
beq $18 $18 label0
```

```
mtlo $8
mflo $23
sltiu $3 $12 1693
jal label6
mult $8 $18
jr $12
ori $3 $12 6774
xori $12 $18 6774
j label1
xori $12 $23 6774
mtlo $12
j label1
lhu $3 6774($3)
sw $8 6774($12)
ori $12 $18 6774
mtlo $3
divu $18 $12
lui $12 6774
mflo $18
sw $3 6774($18)
bne $3 $3 label7
lui $8 6774
mthi $8
lw $12 6774($12)
slti $12 $23 1693
mfc0 $8 $14
sh $12 6774($12)
sltiu $23 $12 1693
jal label7
mflo $23
beq $23 $18 label9
srl $23 $18 3
jalr $18 $3
xori $23 $23 6774
lui $8 6774
label6:nop
sltiu $5 $10 2032
sw $24 8128($5)
slti $5 $10 2032
mthi $10
mtlo $2
lui $24 8128
bgtz $24 label9
mtlo $28
mfhi $10
xori $24 $28 8128
lui $2 8128
mthi $5
slti $28 $24 2032
jalr $28 $28
lui $5 8128
ori $10 $24 8128
lbu $10 8128($10)
jalr $24 $10
addiu $10 $10 8128
lui $24 8128
lui $5 8128
mtc0 $28 $13
```

```
mflo $28
srl $24 $10 3
lhu $2 8128($10)
divu $10 $5
j label1
addiu $2 $5 8128
beq $10 $2 label5
mtc0 $10 $12
subu $5 $28 $28
andi $10 $24 8128
multu $24 $5
subu $24 $24 $28
mflo $10
mtlo $2
sll $10 $5 3
blez $5 label3
mfhi $10
mtc0 $2 $12
slti $5 $2 2032
slt $10 $5 $28
mtc0 $28 $15
sll $28 $28 3
sw $10 8128($24)
srlv $5 $5 $5
srl $5 $5 3
mfc0 $5 $14
mtc0 $2 $14
srl $24 $2 3
label7:nop
mflo $24
lui $1 43964
jalr $24 $20
mfc0 $1 $12
jalr $20 $20
sra $24 $20 21
mtc0 $20 $12
mfhi $24
lui $1 43964
ori $3 $1 43964
mfc0 $24 $12
jalr $3 $24
addiu $1 $3 43964
jalr $24 $0
lui $20 43964
mfc0 $3 $14
bltz $3 label4
andi $20 $20 43964
lh $1 43964($3)
mfhi $24
lui $0 43964
jal label8
slti $20 $24 10991
lhu $20 43964($20)
lui $20 43964
and $0 $3 $3
xori $1 $1 43964
bne $20 $0 label3
sh $20 43964($0)
```

```
addiu $24 $24 43964
mtlo $0
mtlo $20
sw $3 43964($3)
mtc0 $24 $15
bltz $24 label9
andi $3 $0 43964
mflo $3
mflo $3
bgez $0 label9
mtc0 $3 $14
add $3 $0 $0
divu $20 $3
lui $3 43964
lui $1 43964
ori $0 $3 43964
blez $3 label6
mfc0 $1 $12
lb $0 43964($3)
lh $3 43964($24)
jr $1
label8:nop
mtc0 $24 $12
mfc0 $19 $13
mthi $31
sw $19 22587($29)
mfc0 $8 $14
and $31 $24 $24
sw $29 22587($19)
mfhi $8
lui $19 22587
jal label7
ori $29 $24 22587
jalr $19 $8
lui $24 22587
bltz $31 label0
sw $19 22587($31)
jr $19
addi $8 $19 22587
lui $24 22587
bltz $29 label7
addi $31 $31 22587
lui $31 22587
lui $19 22587
j label3
andi $31 $19 22587
sll $29 $24 11
srav $29 $24 $8
bne $8 $8 label5
addiu $8 $19 22587
j label4
addiu $19 $19 22587
mfc0 $19 $15
mthi $19
lw $31 22587($31)
bne $19 $19 label4
addiu $8 $29 22587
xori $19 $31 22587
```

```
addiu $24 $24 22587
sw $8 22587($29)
jr $19
addi $24 $29 22587
lw $19 22587($8)
bne $31 $19 label5
mfhi $31
sub $24 $29 $24
mflo $24
j label5
slti $19 $29 5646
divu $29 $29
sb $29 22587($19)
j label9
label9:nop
mflo $7
bne $5 $7 label0
lui $3 30719
jal label5
lui $4 30719
xori $5 $5 30719
sra $7 $26 14
jalr $5 $26
lbu $3 30719($5)
lui $3 30719
sb $3 30719($3)
jalr $3 $5
add $3 $7 $7
lhu $7 30719($5)
mfhi $3
slti $4 $26 7679
add $5 $5 $26
sw $4 30719($4)
jr $3
andi $4 $4 30719
mfhi $26
bne $3 $4 label2
lui $4 30719
addi $4 $7 30719
mfhi $5
sltiu $26 $5 7679
bgtz $4 label5
andi $26 $3 30719
jr $3
lui $4 30719
mfc0 $26 $13
srlv $5 $7 $3
addu $3 $26 $4
sll $4 $7 14
bltz $3 label6
addi $5 $4 30719
lh $7 30719($26)
sb $5 30719($4)
blez $4 label7
mflo $4
lui $7 30719
sll $26 $26 14
multu $3 $4
```

```
sh $26 30719($5)
sh $4 30719($7)
mthi $26
bgtz $26 label3
andi $5 $3 30719
jal label0
lui $26 30719

.ktext 0x4180
mfc0 $28 $14
ori $29 $0 0x4ffc
bgt $28 $29 next
ori $29 $0 0x3000
blt $28 $29 next
normal:nop
eret
nop
next:
la $29 label1
mtc0 $29 $14
eret
```

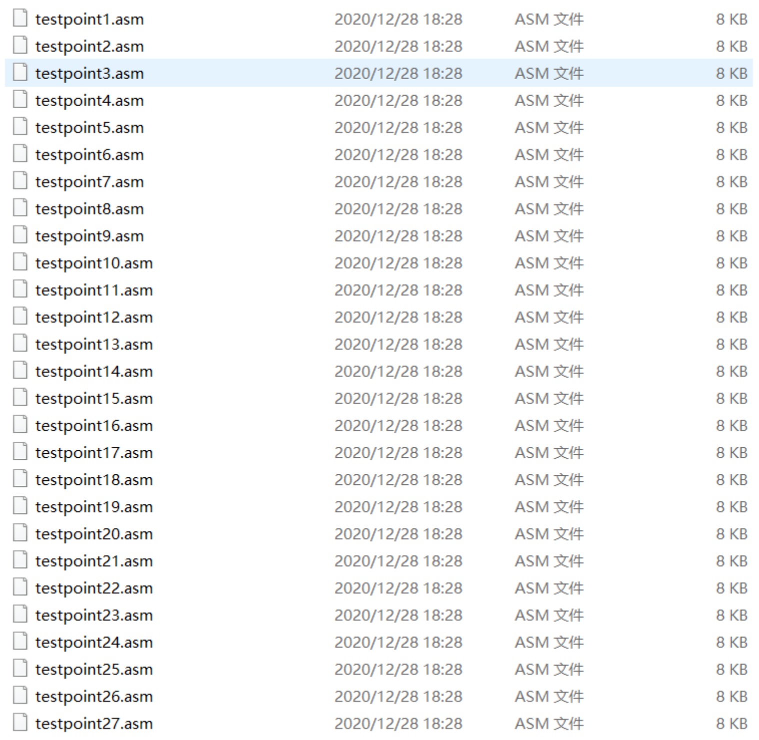
生成的冒险测试代码举例

```
label0:nop
lhu $22 927($11)
sw $11 539($11)
lhu $11 96($22)
lb $19 2352($19)
sw $22 2131($22)
sw $19 3702($22)
lbu $11 3681($11)
lw $22 2051($11)
lb $11 245($11)
sh $19 2451($11)
lhu $22 266($22)
lb $11 3202($22)
sb $19 1560($22)
lh $19 291($22)
lb $11 186($19)
lbu $22 3553($11)
lw $19 2859($19)
lh $11 2035($19)
sb $11 3040($11)
sw $22 2878($22)
label1:nop
lb $22 2412($11)
lw $22 2037($22)
lb $19 2921($22)
sw $11 3854($22)
lh $22 3639($22)
sw $11 3429($19)
sh $19 2($19)
lh $19 462($22)
lb $22 3650($11)
sw $19 1429($22)
```

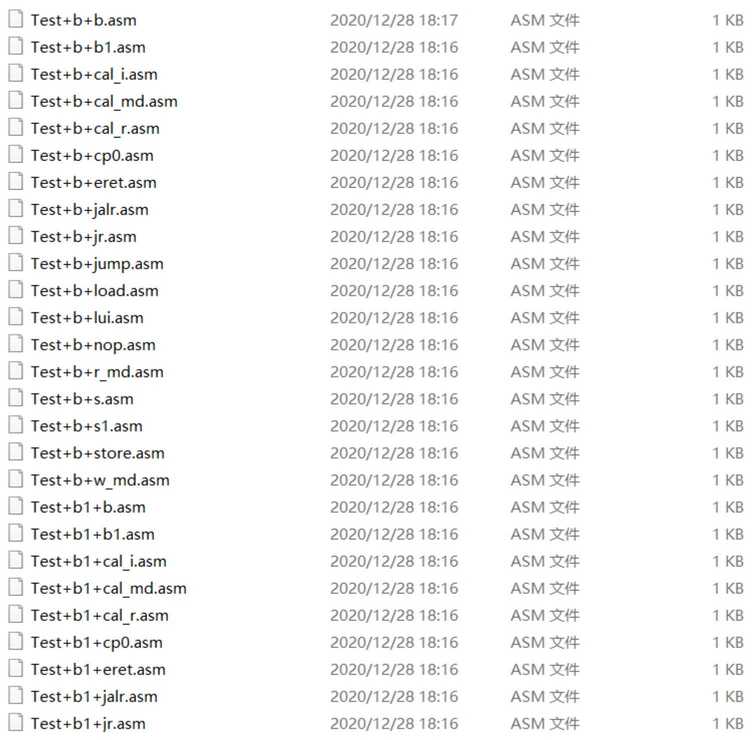
```
lw $19 4059($11)
sw $22 1959($11)
lw $11 839($11)
sh $11 3906($19)
lw $22 880($22)
sb $22 3008($19)
lw $19 2602($19)
lh $11 72($19)
lbu $22 903($11)
sb $11 1831($22)

.ktext 0x4180
mfc0 $28 $14
ori $29 $0 0x4ffc
bgt $28 $29 next
ori $29 $0 0x3000
blt $28 $29 next
normal:nop
eret
nop
next:
la $29 label1
mtc0 $29 $14
eret
```

批量生成截图



testpoint1.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint2.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint3.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint4.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint5.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint6.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint7.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint8.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint9.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint10.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint11.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint12.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint13.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint14.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint15.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint16.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint17.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint18.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint19.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint20.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint21.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint22.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint23.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint24.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint25.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint26.asm	2020/12/28 18:28	ASM 文件	8 KB
testpoint27.asm	2020/12/28 18:28	ASM 文件	8 KB



Test+b+b.asm	2020/12/28 18:17	ASM 文件	1 KB
Test+b+b1.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+cal_i.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+cal_md.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+cal_r.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+cp0.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+eret.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+jalr.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+jr.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+jump.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+load.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+lui.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+nop.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+r_md.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+s.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+s1.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+store.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b+w_md.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b1+b.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b1+b1.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b1+cal_i.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b1+cal_md.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b1+cal_r.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b1+cp0.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b1+eret.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b1+jalr.asm	2020/12/28 18:16	ASM 文件	1 KB
Test+b1+jr.asm	2020/12/28 18:16	ASM 文件	1 KB

自动测试

需要环境：python3环境，且装好os,re,random库

```python
import os
import re
import random
import subprocess
data_source="class_wzk"#input("describe the test data:")
xianshi=1#input("dou you want to see the line of asm code?(y/n):")=="y"
#data_source=""
case=input("Your test files num >1?(y/n):")=="y"
xlinx="D:\\Xilinx\\14.7\\ISE_DS\\ISE"
#case=
#xlinx=input("your xilinx src is:")
if(case):
    file_src=input("input asm test files folder src:(must like this
D:\\P5\\test\\)")
    #file_src=""
    file_list=os.listdir(file_src)

    #k=1
    # for name in file_list:
    #     namek="testpoint"+str(k)+".asm"
    #     try:
    #         os.rename(file_src+name,file_src+namek)
    #     except:
    #         pass
    #     k=k+1
    # file_num=k-1
    file_num=len(file_list)
    print(file_num)
    error_list=[]
    wrong_cnt=0
    path=os.path.dirname(os.path.realpath(__file__))
    os.chdir(path)
    info_name="test_info_"+data_source+".txt"
    f=open(info_name,"w")
    for j in range(1,file_num+1):
        path=os.path.dirname(os.path.realpath(__file__))
        os.chdir(path)
        asmfilename=file_src+file_list[j-1]
        time="2000us"
        filelist=os.walk(path)
        os.environ['XILINX']=xlinx
        with open("mips.prj","w") as prj:
            for folder in filelist:
                for file in folder[2]:
                    if(len(file.split("."))>1 and file.split(".")[1]=="v"):
                        prj.write("verilog work \""+folder[0]+"\\"+file+"\"\n")
        with open("mips.tcl","w") as tcl:
            tcl.write("run "+time+";\nexit;\n")
        os.system("java -jar Mars.jar db a nc mc CompactDataAtZero dump .text 
HexText code.txt 1000000 "+asmfilename)
        os.system("java -jar Mars.jar db a nc mc CompactDataAtZero dump 
0x00004180-0x00005180 HexText code_handler.txt "+asmfilename)
        os.system("java -jar Mars.jar  db nc mc CompactDataAtZero dump .text 
HexText code.txt >ans.txt 1000000 "+asmfilename)
        os.system(xlinx+"\\bin\\nt64\\fuse "+"--nodebug  --prj mips.prj -o 
mips.exe mips_tb >log.txt")
        os.system("mips.exe -nolog -tclbatch mips.tcl >my.txt")

        process=0
        with open("my.txt","r") as my:
            lines=my.readlines()
            if(len(lines)==0):
                print("fail to simulate")
                os._exit(1)
            if(lines[0][0]=='I'):
                process=1
        my.close()
        n=0
        while(1):
            if "@" in lines[n]:
                break
            else:
                n=n+1
        if(process):
            with open("my.txt","w") as my:
                my.writelines(lines[n:])
        i=0
        biao=0
        with open("my.txt","r") as my:
            with open("ans.txt","r") as ans:
                while(1):
                    i+=1
                    l1=my.readline()
                    if(l1!="" and l1!=None):
                        l1="@"+l1.strip().split("@")[-1]
                    l2=ans.readline().strip()
                    if((l1== "" or l1==None)and(l2=="" or l2==None)):
                        break
                    elif l1!=l2 and not "$ 0"in l2 and not "$ 0" in  l1:
                        biao=1
                        error_list.append(asmfilename)
                        try:
                            code_line=int(l1[6:9],16)/4+1
                        except:
                            code_line=0
                            xianshi=0
                        #pattern = re.compile(r"@[0-9a-z]{8}:\s[*][0-9]
{8}\s<=\s[0-9]{8}")
                        # print(l1)
                        # print(l2)
                        # if(l1[:12]==l2[:12] and l1[17:20]==l2[17:20] and
l1[24:31]==l2[24:31]):
                        #     biao=0
                        #     error_list.pop()
                        #     continue
                        # try:
                        #     s1=pattern.match(l1).group()
                        #     s2=pattern.match(l2).group()
                        #     print(s1)
                        #     print(s2)
                        #     if(s1[:12]==s2[:12] and s1[17:20]==s2[17:20] and
s1[24:31]==s2[24:31]):
                        #         print("success")
                        #         biao=0
                        # except:
                        #     pass
                        # if(biao==0):
                        #     continue
                        if(xianshi==1):
                            if l2=="" or l2 == None:
                                print("Wrong answer occur in line %d of your ans
and in line %d of test code: "%(i,code_line)+"we got "+l1+" when we expected
Nothing")
                                f.write("wrong answer on point %d\nWrong answer
occur in line %d of your ans: "%(j,i)+"we got "+l1+" when we expected
Nothing\n")
                            else:
                                print("Wrong answer occur in line %d of your ans
and in line %d of test code: "%(i,code_line)+"we got "+l1+" when we expected
"+l2)
                                f.write("wrong answer on point %d\nWrong answer
occur in line %d of your ans: "%(j,i)+"we got "+l1+" when we expected "+l2+"\n")
                        else:
                            if l2=="" or l2 == None:
                                print("Wrong answer occur in line %d of your
ans: "%(i)+"we got "+l1+" when we expected Nothing")
                                f.write("wrong answer on point %d\nWrong answer
occur in line %d of your ans: "%(j,i)+"we got "+l1+" when we expected
Nothing\n")
                            else:
                                print("Wrong answer occur in line %d of your
ans: "%(i)+"we got "+l1+" when we expected "+l2)
                                f.write("wrong answer on point %d\nWrong answer
occur in line %d of your ans: "%(j,i)+"we got "+l1+" when we expected "+l2+"\n")
                        break

                if biao==0:
                    print("Accepted on the point %d"%j)
                    f.write("Accepted on the point %d\n"%j)
                else:
                    print("Wrong on the point %d"%j)
                    wrong_cnt=wrong_cnt+1
        my.close()
        ans.close()
    if(error_list):
        print("------------------------------------\nYou have %d wrong answer\n
test data description:%s\nthey are:"%(wrong_cnt,data_source))
        f.write("------------------------------------\nYou have %d wrong
answer\n test data description:%s\nthey are:\n"%(wrong_cnt,data_source))
        for x in error_list:
            print(x)
            f.write(x+"\n")
    else:
        print("------------------------------------\ncongratulations!You
accepted %d point\ntest data description:%s"%(file_num,data_source))
        f.write("------------------------------------\ncongratulations!You
accepted %d point\ntest data description:%s\n"%(file_num,data_source))
    f.close()
    os._exit(1)
else:

 file_name="E:\computer\\verilog_ISE\CO\p7_test1\\test\\yichang\\mips7.asm"#inpu
t("input asm test file name(can have src):")
    path=os.path.dirname(os.path.realpath(__file__))
    os.chdir(path)
    asmfilename=file_name
    time="2000us"
    filelist=os.walk(path)
    os.environ['XILINX']=xlinx
    with open("mips.prj","w") as prj:
        for folder in filelist:
            for file in folder[2]:
                if(len(file.split("."))>1 and file.split(".")[1]=="v"):
                    prj.write("verilog work \""+folder[0]+"\\"+file+"\"\n")
    with open("mips.tcl","w") as tcl:
        tcl.write("run "+time+";\nexit;\n")
    os.system("java -jar Mars.jar db a nc mc CompactDataAtZero dump .text
HexText code.txt 1000000 "+asmfilename)
    os.system("java -jar Mars.jar db a nc mc CompactDataAtZero dump 0x00004180-
0x00005180 HexText code_handler.txt "+asmfilename)
    os.system("java -jar Mars.jar  db nc mc CompactDataAtZero dump .text HexText
code.txt >ans.txt 1000000 "+asmfilename)
    os.system(xlinx+"\\bin\\nt64\\fuse "+"--nodebug  --prj mips.prj -o mips.exe
mips_tb >log.txt")
    os.system("mips.exe -nolog -tclbatch mips.tcl >my.txt")

    process=0
    with open("my.txt","r") as my:
        lines=my.readlines()
        if(len(lines)==0):
            print("fail to simulate")
            os._exit(1)
        if(lines[0][0]=='I'):
            process=1
    my.close()
    n=0
    while(1):
            if "@" in lines[n]:
                break
            else:
                n=n+1
    if(process):
        with open("my.txt","w") as my:
            my.writelines(lines[n:])
    i=0
    biao=0
    wrong_cnt=0
    with open("my.txt","r") as my:
        with open("ans.txt","r") as ans:
            while(1):
                i+=1
                l1=my.readline()
                if(l1!="" and l1!=None):
                    l1="@"+l1.strip().split("@")[-1]
                l2=ans.readline().strip()
                if((l1== "" or l1==None)and(l2=="" or l2==None)):
                    break
                elif l1!=l2 and not "$ 0"in l2 and not "$ 0" in  l1:
                    biao=1
                    if(l1[:12]==l2[:12] and l1[17:20]==l2[17:20] and
l1[24:31]==l2[24:31]):
                        biao=0
                        continue
                    try:
                        code_line=int("0x"+l1[6:9],16)/4+1
                    except:
                        xianshi=0
                    if(xianshi==1):
                        if l2=="" or l2 == None:
                            print("Wrong answer occur in line %d of your ans and
in line %d of test code: "%(i,code_line)+"we got "+l1+" when we expected
Nothing")
                        else:
                            print("Wrong answer occur in line %d of your ans and
in line %d of test code: "%(i,code_line)+"we got "+l1+" when we expected "+l2)
                    else:
                        if l2=="" or l2 == None:
                            print("Wrong answer occur in line %d of your ans: "%
(i)+"we got "+l1+" when we expected Nothing")
                        else:
                            print("Wrong answer occur in line %d of your ans: "%
(i)+"we got "+l1+" when we expected "+l2)
                    break

            if biao==0:
                print("Accepted on this point")
            else:
                print("Wrong on the point")
                wrong_cnt=wrong_cnt+1
    my.close()
    ans.close()
    os._exit(1)

#E:\computer\verilog_ISE\CO\p6\test\ch\testpoint2.asm
```

本测试程序支持测试多个asm测试点或单个测试点，利用魔改Mars进行对拍

三、思考题

1.我们计组课程一本参考书目标题中有“硬件/软件接口”接口字样，那么到底什么是“硬件/软件接口”？（Tips：什么是接口？和我们到现在为止所学的有什么联系？）

硬件接口是一个硬件和另一个硬件的连接装置，真实存在

软件接口指不同软件间相互联系的入口，往往是一段有一定具体意义的代码。

在P7中，沟通软硬件的接口为系统桥，硬件之间的接口为计时器的各个输入输出端口。CPU的几个输出输入端为软件的接口。

2.在我们设计的流水线中，DM 处于 CPU 内部，请你考虑现代计算机中它的位置应该在何处。

分布在多个部件和CPU的内部的多个部分

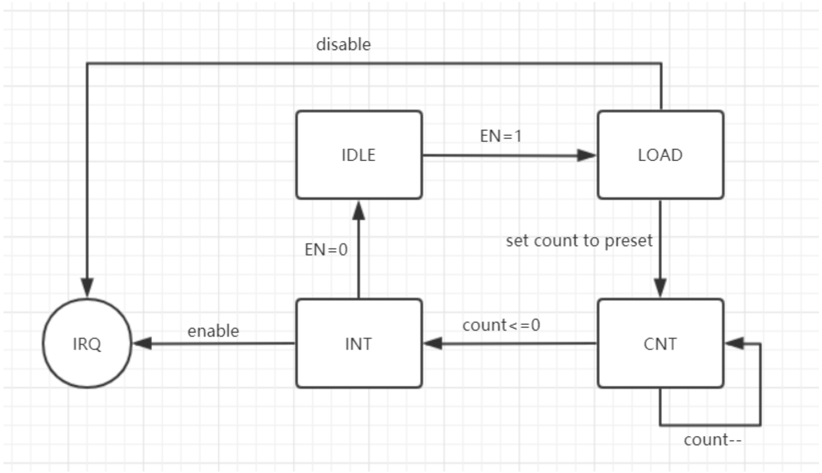
3.BE 部件对所有的外设都是必要的吗?

不是，只是对于能够写入半字或者1字节的设备必需

4.请阅读官方提供的定时器源代码，阐述两种中断模式的异同，并分别针对每一种模式绘制状态转移图

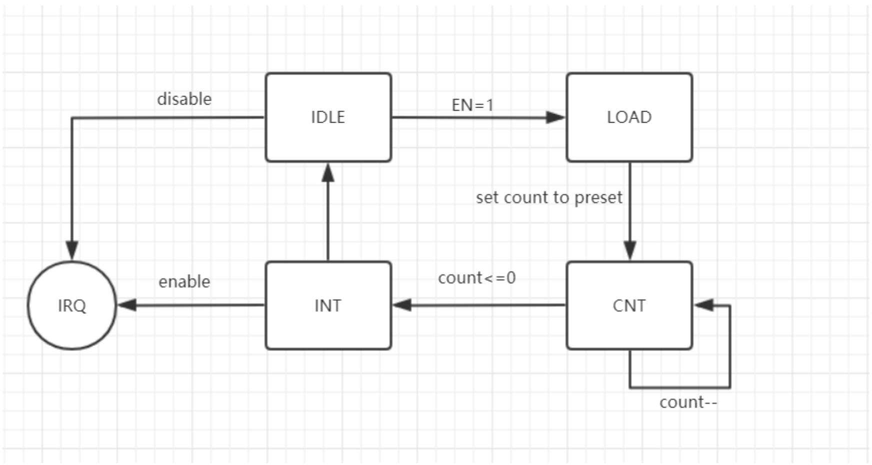
工作模式 1

计时器从设定值到0后，持续中断,直到内存中的ctrl寄存器的最低位变为1时，才停止产生中断信号



工作模式 2

计时器从设定值到0后，中断产生一个周期，在之后转到 IDLE状态且中断停止



相同：都能产生中断
不同：第一种持续若干个周期，第二种只产生一个周期

5.请开发一个主程序以及定时器的exception handler。整个系统完成如下功能：

1. **定时器在主程序中被初始化为模式0；**
2. **定时器倒计数至0产生中断；**
3. **handler设置使能Enable为1从而再次启动定时器的计数器。2及3被无限重复。**
4. **主程序在初始化时将定时器初始化为模式0，设定初值寄存器的初值为某个值，如100或1000。（注意，主程序可能需要涉及对CP0.SR的编程，推荐阅读过后文后再进行。）**

```
.ktext 0x4180
ori $1 $0 9
sw $1 0x7f00($0)
eret

.text
ori $1 $0 0x0c01
mtc0 $1 $12
ori $1 $0 9
sw $1 0x7f00($0)
ori $1 $0 20
sw $1 0x7f04($0)
label:
beq $4 $4 label
nop
```

6.请查阅相关资料，说明鼠标和键盘的输入信号是如何被CPU知晓的?

通过外部中断信号让CPU的PC值进入特殊的地址，处理特定的程序，从而接受和处理输入信号。