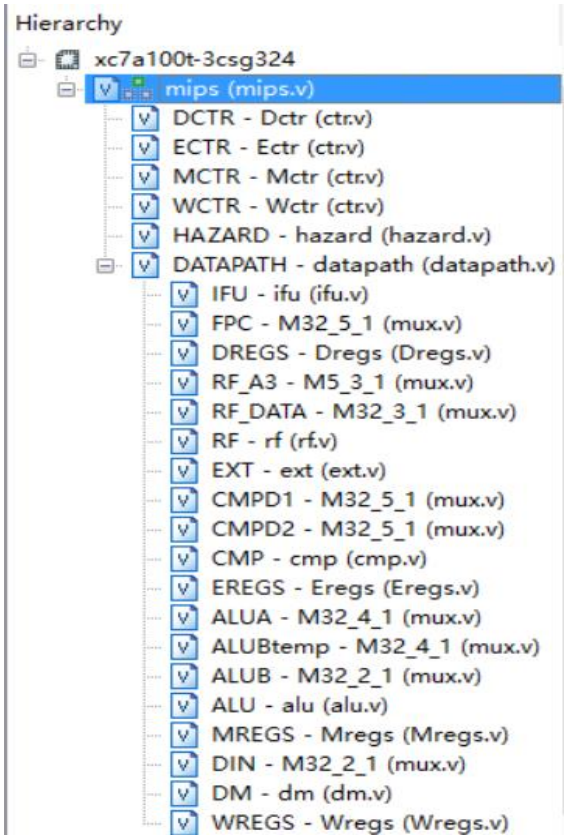
计算机组成原理实验报告

一、 CPU 设计文档

（一） 总体设计



图表 1 模块设计

PC		M_nPC	PCSel	ADD4	NPC	MFPCF		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4/NPC	NPC	NPC	RF.V1	RF.V1
ADD4				PC				PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC
IM				PC				PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC
D级寄存器	IR_D			IM				IM	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM
	PC8_D			ADD4+4												ADD4+4		ADD4+4
	PC4_D			ADD4										ADD4		ADD4		ADD4
RF	A1			IR_D[rs]				IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D[rs]			IR_D[rs]	IR_D[rs]
	A2			IR_D[rt]					IR_D[rt]	IR_D[rt]	IR_D[rt]			IR_D[rt]				
EXT				IR_D[I16]				IR_D[I16]	IR_D[I16]			IR_D[I16]	IR_D[I16]					
CMP	D1			MFcmp1D										RF.V1				
	D2			MFcmp2D										RF.V2				
NPC	I26			IR_D[I26]										IR_D[I16]	IR_D[I26]	IR_D[I26]		
E级寄存器	IR_E			IR_D				IR_D	IR_D	IR_D	IR_D	IR_D	IR_D			IR_D		IR_D
	PC8_E			PC8_D												PC8_D		PC8_D
	PC4_E			PC4_D												PC4_D		PC4_D
	V1_E			D1				RF.V1	RF.V1	RF.V1	RF.V1	RF.V1						
	V2_E			D2					RF.V2	RF.V2	RF.V2							
	E32_E			EXT				EXT	EXT			EXT	EXT					
ALU	A			MFaluaE				V1_E	V1_E	V1_E	V1_E	V1_E						
	B	M_ALUB	BSel	MFalubE	E32_E			E32_E	E32_E	V2_E	V2_E	E32_E	E32_E					
M级寄存器	IR_M			IR_E				IR_E	IR_E	IR_E	IR_E	IR_E	IR_E			IR_E		IR_E
	PC8_M			PC8_E												PC8_E		PC8_E
	PC4_M			PC4_E												PC4_E		PC4_E
	AO_M			ALU				ALU	ALU	ALU	ALU	ALU	ALU					
	V2_M			MFalubE					V2_E									
DM	A			AO_M				AO_M	AO_M									
	WD			MFdmdataM					V2_M									
W级寄存器	IR_W			IR_M				IR_M		IR_M	IR_M	IR_M	IR_M			IR_M		IR_M
	PC8_W			PC8_M												PC8_M		PC8_M
	PC4_W			PC4_M												PC4_M		PC4_M
	AO_W			AO_M						AO_M	AO_M	AO_M	AO_M					
	DR_W			DM				DM										
RF	A3	M_WReg	WRSel	IR_W[rt]	IR_W[rd]	0x1f		IR_W[rt]		IR_W[rd]	IR_W[rd]	IR_W[rt]	IR_W[rt]			0x1f		IR_W[rd]
	WD	M_Wdata	WDSel	AO_W	DR_W	PC8_W		DR_W		AO_W	AO_W	AO_W	AO_W			PC8_W		PC8_W

图表 2 数据通路设计

Tuse		
指令	rs	rt
addu	1	1
subu	1	1
ori	1	
lui		
lw	1	
sw	1	2
beq	0	0
jr	0	
jalr	0	
j		
jal		

Tnew				
指令	功能部件	E	M	W
addu	ALU	1	0	0
subu	ALU	1	0	0
ori	ALU	1	0	0
lui	ALU	1	0	0
lw	DM	2	1	0
sw				
beq				
jr				
jalr	PC	0	0	0
j				
jal	PC	0	0	0

转发MUX	控制信号	输入0	输入1	输入2	输入3	输入4
MFcmp1D	Fcmp1D	RF.V1	M_Wdata	AO_M	PC8_M	PC8_E
MFcmp2D	Fcmp2D	RF.V2	M_Wdata	AO_M	PC8_M	PC8_E
MFaluaE	FaluaE	V1_E	M_Wdata	AO_M	PC8_M	
MFalubE	FalubE	V2_E	M_Wdata	AO_M	PC8_M	
MFdmdataM	FdmdataM	V2_M	M_Wdata			
MFPCF	FPCF	RF.V1	M_Wdata	AO_M	PC8_M	PC8_E

rs		Tnew								
		E			M			W		
		ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
		1	2	0	0	1	0	0	0	0
Tuse	0	S	S	F	F	S	F	F	F	F
	1	F	S	F	F	F	F	F	F	F

rt		Tnew								
		E			M			W		
		ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
		1	2	0	0	1	0	0	0	0
Tuse	0	S	S	F	F	S	F	F	F	F
	1	F	S	F	F	F	F	F	F	F
	2	F	F	F	F	F	F	F	F	F

图表 3 转发暂停设计

func	100001	100011	001000	001001									
op	000000	000000	000000	000000	001101	100011	101011	000100	001111	001110	000011	000010	
	addu	subu	jr	jalr	ori	lw	sw	beq	lui	xori	jal	j	
RegDst	1	1	0	1	0	0	0	0	0	0	0	0	0rt 1rd 进rf2
ALUSrc	0	0	0	0	1	1	1	0	1	1	0	0	0grfdata2 1i32 进aluB
MemtoReg	0	0	0	0	0	1	0	0	0	0	0	0	
RegWrite	1	1	0	1	1	1	0	0	1	1	1	0	
MemWrite	0	0	0	0	0	0	1	0	0	0	0	0	
EXTop	00	00	00	00	00	10	10	11	01	00	00	00	0无符号 1低16位补0 2有符号 3有符号后左移两位
ALUctr	000	001	000	000	010	000	000	000	011	100	000	000	0加 1减 2或 3B 4异或

图表 4 控制器设计

（二） 数据通路设计

1. datapath（数据通路）

1）端口说明

表格 1datapath 端口说明

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 1：复位 0：无效
PCsel[1:0]	I	D 控制器发来选择 nPC 信号 10：转发器 MFPCF 结果 01：NPC 00：ADD4
i16[15:0]	I	D 控制器发来 16 位立即数
i26[25:0]	I	D 控制器发来 26 位立即数
RegWrite	I	W 控制器发来写寄存器信号 1：写寄存器 0：无效
MemWrite	I	M 控制器发来写内存信号 1：写内存 0：无效
EXTop[2:0]	I	D 控制器发来扩展器信号 000：无符号扩展 001：低 16 位补 0 010：有符号扩展 011：有符号扩展后逻辑左移两位
ALUctr[2:0]	I	E 控制器发来 ALU 控制信号 000：加运算 001：减运算 010：或运算 011：输出写入数据 2 100:异或运算
WRsel[1:0]	I	D 控制器发来选择寄存器 A3 端口信号 10:31 号寄存器 01：IR_W[rd] 00：IR_W[rt]
WDsel[1:0]	I	W 控制器发来选择寄存器输入信号 00：AO_W01：DR_W10：PC8_W
Fcmp1D[2:0]	I	冒险单元发来选择 MFcmp1D 信号 000：RF.V1 001：M_Wdata 010：AO_M 011：PC8_M 100: PC8_E
Fcmp2D[2:0]	I	冒险单元发来选择 MFcmp2D 信号 000：RF.V2 001：M_Wdata 010：AO_M 011：PC8_M 100: PC8_E
FdmdataM	I	冒险单元发来选择 MFdmdataM 信号 0：V2_M 1：M_Wdata
FaluaE[1:0]	I	冒险单元发来选择 MFaluaE 信号 00：V1_E 01：M_Wdata 10：AO_M 11：PC8_M
FalubE[1:0]	I	冒险单元发来选择 MFalubE 信号 00：V2_E 01：M_Wdata 10：AO_M 11：PC8_M
FPCF[2:0]	I	冒险单元发来选择 MFPCF 信号 000：RF.V1 001：M_Wdata 010：AO_M 011：

		PC8_M 100: PC8_E
Bsel	I	E 控制器发来 ALUb 选择信号 0：MFalubE 选择结果 1：E32_E
stall	I	冒险单元发来选择暂停信号 1：暂停 0：无效
beq	I	D 控制器发来 beq 识别信号 1：beq 0：无效
IRF[31:0]	O	输出到 D 控制器的 F 级指令
IRD[31:0]	O	输出到 E 控制器的 D 级指令
IRE[31:0]	O	输出到 M 控制器的 E 级指令
IRM[31:0]	O	输出到 W 控制器的 M 级指令
RESE[1:0]	O	输出到冒险单元的 E 级 Tnew 状态 00：NW 不写 01：写 ALU 10：写 DM 11：写 PC
RESM[1:0]	O	输出到冒险单元的 M 级 Tnew 状态 00：NW 不写 01：写 ALU 10：写 DM 11：写 PC
RESW[1:0]	O	输出到冒险单元的 W 级 Tnew 状态 00：NW 不写 01：写 ALU 10：写 DM 11：写 PC
RFA3E[4:0]	O	输出到冒险单元的 E 级指令 A3 寄存器
RFA3M[4:0]	O	输出到冒险单元的 M 级指令 A3 寄存器
RFA3W[4:0]	O	输出到冒险单元的 W 级指令 A3 寄存器

2）功能定义

表格 2datapath 功能定义

序号	功能名称	功能描述
1	连接基本模块	通过 datapath，以声明中间变量和实例化引用的方式连接各基础模块

2. ifu（取指令单元）

1）端口说明

表格 3ifu 端口说明

信号名	方向	描述
reset	I	复位信号 1：复位 0：无效

clk	I	时钟信号
PCsel[1:0]	I	D 控制器发来选择 nPC 信号 10：转发器 MFPCF 结果 01：NPC 00：ADD4
CO	I	cmp 发来比较信号 1：alu 两输入相等 0：alu 两输入不等
i16 [15:0]	I	D 控制器发来 16 位立即数
i26[25:0]	I	D 控制器发来 26 位立即数
PCtempD[31:0]	I	MFPCF 转发多选器的结果
beq	I	D 控制器发来 beq 识别信号 1：beq 0：无效
stall	I	冒险单元发来选择暂停信号 1：暂停 0：无效
instruction[31:0]	O	输出的指令
PC8[31:0]	O	输出的当前 PC+8

2）功能定义

表格 4 ifu 功能定义

序号	功能名称	功能描述
1	复位	复位信号有效时，PC 被置为 0x00000000
2	取指令	根据 PC 从 IM 中取出指令
3	计算下条指令地址	PC←PC+4 || PC←reg1data || PC←PC + 4 + immed32 || PC←{PC[31:28], immed26, 2'b0}
4	暂停	Stall 信号有效时，冻结 PC 寄存器

3. rf（寄存器堆）

1）端口说明

表格 5grf 端口说明

信号名	方向	描述
reset	I	复位信号 1：复位 0：无效
clk	I	时钟信号
reg1 [4:0]	I	读寄存器号 1 编号
Reg2 [4:0]	I	读寄存器号 2 编号

writereg[4:0]	I	写寄存器编号
regwrite	I	写控制信号 1：写入 0：无效
writedata[31:0]	I	写入的 32 位数据
data1[31:0]	O	32 位寄存器 1 输出
data2[31:0]	O	32 位寄存器 2 输出

2）功能定义

表格 6grf 功能定义

序号	功能名称	功能描述
1	复位	复位信号有效时，32 个寄存器被置为 0x00000000
2	写寄存器	写寄存器控制信号有效时，把 32 位数据写入寄存器
3	读寄存器	根据输入的地址读出两个寄存器中的值

4. alu（算术逻辑单元）

1）端口说明

表格 7alu 端口说明

信号名	方向	描述
A[31:0]	I	32 位写入数据 1
B[31:0]	I	32 位写入数据 2
ALUOp[2:0]	I	控制信号 000：加运算 001：减运算 010：或运算 011：输出写入数据 2 100：异或运算
AO[31:0]	O	32 位输出数据

2）功能定义

表格 8alu 功能定义

序号	功能名称	功能描述
1	加运算	A+B
2	减运算	A-B
3	或运算	A|B

4	输出写入数据 2	B
5	异或运算	A^B

5. dm（数据存储器）

1）端口说明

表格 9dm 端口说明

信号名	方向	描述
D1[31:0]	I	32 位输入数据 1
D2[31:0]	I	32 位输入数据 2
CO	O	比较结果 0：不相等 1：相等

2）功能定义

表格 10dm 功能定义

序号	功能名称	功能描述
1	比较	比较两输入数据大小，相等输出 1，否则输出 0

6. cmp（比较器）

1）端口说明

表格 11dm 端口说明

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 1：复位 0：无效
ADDR [31:0]	I	32 位写入内存地址
din[31:0]	I	32 位写入数据
PC8[31:0]	I	当前 PC+8
MemWrite	I	写内存控制信号 1：写入 0：无效
dout[31:0]	O	32 位输出数据

2）功能定义

表格 12dm 功能定义

序号	功能名称	功能描述
1	复位	复位信号有效时，内存和读出内存的寄存器被置为 0x00000000
2	写内存	写内存控制信号有效时，根据输入的地址写入 32 位数据
3	读内存	根据输入的地址读出内存数据

7. ext（位扩展器）

1）端口说明

表格 13ext 端口说明

信号名	方向	描述
immed16[15:0]	I	16 位写入立即数
EXTop[2:0]	I	扩展控制信号 000：无符号扩展 001：低 16 位补 0 010：有符号扩展 011：有符号扩展后逻辑左移两位
EO[31:0]	O	32 位输出立即数

2）功能定义

表格 14ext 功能定义

序号	功能名称	功能描述
1	无符号扩展	高 16 位补 0
2	低 16 位补 0	低 16 位补 0
3	有符号扩展	Immed[15]为 1 时高 16 位补 1，为 0 时高 16 位补 0
4	有符号扩展后逻辑左移两位	Immed[15]为 1 时高 16 位补 1，为 0 时高 16 位补 0，再左移两位，溢出舍去，低 2 位补 0

8. mux（多路选择器）

1）端口说明

表格 15mux 端口说明

信号名	方向	描述
A[4:0]	I	5 位输入 A
B[4:0]	I	5 位输入 B
C[4:0]	I	5 位输入 C
Op[1:0]	I	选择控制信号 10：输出 C 01：输出 B 00：输出 A
O [4:0]	O	5 位输出 O
A[31:0]	I	32 位输入 A
B[31:0]	I	32 位输入 B
C[31:0]	I	32 位输入 C
D[31:0]	I	32 位输入 D
E[31:0]	I	32 位输入 E
op[2:0]	I	选择控制信号 100：输出 E 011：输出 D 010：输出 C 001：输出 B 000：输出 A
O[31:0]	O	32 位输出 O

2）功能定义

表格 16mux 功能定义

序号	功能名称	功能描述
1	5 位输入 3 选 1	option 为 10 输出 C，为 01 输出 B，为 00 输出 A
2	32 位输入 2 选 1	option 为 1 输出 B，为 0 输出 A
3	32 位输入 3 选 1	option 为 10 输出 C，为 01 输出 B，为 00 输出 A
4	32 位输入 4 选 1	option 为 11 输出 D，为 10 输出 C，为 01 输出 B，为 00 输出 A
5	32 位输入 5 选 1	option100 输出 E，011 输出 D，010 输出 C，001 输出 B，000 输出 A

9. Dregs（D 级流水线寄存器）

1）端口说明

表格 17 Dregs 端口说明

信号名	方向	描述

clk	I	时钟信号
reset	I	复位信号 1：复位 0：无效
IR[31:0]	I	D 级部件将使用的指令
PC8[31:0]	I	D 级部件对应指令的 PC+8
Stall	I	冒险单元输入的暂停信号
PCsel[1:0]	I	将传回 F 级部件的 PC 选择信号
I16[15:0]	I	将传回 F 级部件的 16 位立即数
I26[25:0]	I	将传回 F 级部件的 26 位立即数
Beq	I	将传回 F 级部件的 beq 识别信号
EXTop[2:0]	I	扩展控制信号 000：无符号扩展 001：低 16 位补 0 010：有符号扩展 011：有符号扩展后逻辑左移两位
WRsel[1:0]	I	选择寄存器 A3 端口信号 10:31 号寄存器 01：IR_W[rd] 00：IR_W[rt]
IR_D [31:0]	O	输出到 E 级寄存器的指令
PC8_D[31:0]	O	输出到 E 级寄存器的 PC+8
PCsel_D[1:0]	O	输出到 F 级寄存器的 PC 选择信号
i16_D	O	将传回 F 级部件的 16 位立即数
I26_D	O	将传回 F 级部件的 26 位立即数
Beq_D	O	将传回 F 级部件的 beq 识别信号
EXTop_D[2:0]	O	扩展控制信号 000：无符号扩展 001：低 16 位补 0 010：有符号扩展 011：有符号扩展后逻辑左移两位
WRsel_D[1:0]	O	选择寄存器 A3 端口信号 10:31 号寄存器 01：IR_W[rd] 00：IR_W[rt]

2）功能定义

表格 18 Dregs 功能定义

序号	功能名称	功能描述
1	存储结果	存储 F 级部件结果，发送到 D 级部件或 F 级部件

10. Eregs（E 级流水线寄存器）

1）端口说明

表格 19 Eregs 端口说明

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 1：复位 0：无效
IR[31:0]	I	E 级部件将使用的指令
PC8[31:0]	I	E 级部件对应指令的 PC+8
Stall	I	冒险单元输入的暂停信号
V1[31:0]	I	MFcmp1D 转发而来的结果
V2[31:0]	I	MFcmp2D 转发而来的结果
E32[31:0]	I	EXT 结果
RFA3[4:0]	I	E 级指令要写入的寄存器编号
Bsel	I	Alub 多选器的选择信号
ALUctr [2:0]	I	控制信号 000：加运算 001：减运算 010：或运算 011：输出写入数据 2 100:异或运算
IR_E[31:0]	O	输出到 E 级寄存器的指令
PC8_E[31:0]	O	输出到 E 级寄存器的 PC+8
RFA3E[4:0]	O	输出到 E 级的要写入的寄存器编号
V1[31:0]	O	MFcmp1D 转发而来的结果
V2[31:0]	O	MFcmp2D 转发而来的结果
Bsel_E	O	Alub 多选器的选择信号
ALUctr_E[2:0]	O	alu 控制信号 000：加运算 001：减运算 010：或运算 011：输出写入数据 2 100:异或运算
Res_E [1:0]	O	E 级指令对部件的产生结果位置 ALU：在 alu 产生结果 DM：在 dm 产生结果 PC：产生 PC 结果 NW：nowrite，不产生结果

2）功能定义

表格 20Eregs 功能定义

序号	功能名称	功能描述

1	存储结果	存储 D 级部件结果，发送到 E 级部件
2	产生控制转发信号	计算 E 级指令对部件的产生结果位置

11.Mregs（M 级流水线寄存器）

1）端口说明

表格 21Mregs 端口说明

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 1：复位 0：无效
IR[31:0]	I	M 级部件将使用的指令
PC8[31:0]	I	M 级部件对应指令的 PC+8
AO[31:0]	I	ALU 的结果
V2[31:0]	I	MFcmp2D 转发而来的结果
RFA3[4:0]	I	M 级指令要写入的寄存器编号
MemWrite	I	Dm 写入控制信号 1：写入 0：无效
IR_M[31:0]	O	输出到 W 级寄存器的指令
PC8_M[31:0]	O	输出到 W 级寄存器的 PC+8
RFA3M[4:0]	O	输出到 M 级的要写入的寄存器编号
AO_M[31:0]	O	输出到 W 级的 alu 结果
V2[31:0]	O	输出到 W 级的 MFcmp2D 转发而来的结果
MemWrite	O	输出到 M 级部件的 Dm 写入控制信号 1：写入 0：无效
Res_M[1:0]	O	M 级指令对部件的产生结果位置 ALU：在 alu 产生结果 DM：在 dm 产生结果 PC：产生 PC 结果 NW：nowrite，不产生结果

2）功能定义

表格 22Mregs 功能定义

序号	功能名称	功能描述
1	存储结果	存储 E 级部件结果，发送到 M 级部件

2	产生控制转发信号	计算 M 级指令对部件的产生结果位置

12.Wregs（W 级流水线寄存器）

1）端口说明

表格 23Wregs 端口说明

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号 1：复位 0：无效
IR[31:0]	I	W 级部件将使用的指令
PC8[31:0]	I	W 级部件对应指令的 PC+8
AO[31:0]	I	ALU 的结果
DR[31:0]	I	DM 的结果
RFA3[4:0]	I	W 级指令要写入的寄存器编号
RegWrite	I	寄存器堆写入控制信号 1：写入 0：无效
WDsel[1:0]	I	寄存器堆写入数据选择信号 10: PC8_W 01：DR_W 00：AO_W
PC8_W[31:0]	O	输出到 W 级部件的 PC+8
RFA3W[4:0]	O	输出到 W 级的要写入的寄存器编号
AO_W[31:0]	O	输出到 W 级部件的 alu 结果
DR_W[31:0]	O	输出到 W 级部件 DM 的结果
RegWrite	O	输出到 W 级部件的寄存器写入控制信号 1：写入 0：无效
WDsel_W[1:0]	O	寄存器堆写入数据选择信号 10: PC8_W 01：DR_W 00：AO_W
Res_W[1:0]	O	W 级指令对部件的产生结果位置 ALU：在 alu 产生结果 DM：在 dm 产生结果 PC：产生 PC 结果 NW：nowrite，不产生结果

2）功能定义

表格 24Wregs 功能定义

序号	功能名称	功能描述
1	存储结果	存储 M 级部件结果，发送到 W 级部件

2	产生控制转发信号	计算 W 级指令对部件的产生结果位置

（三） 冒险单元

1. 端口说明

表格 25hazard 端口说明

信号名	方向	描述
IR[31:0]	I	D 级部件将使用的指令
Res_E[1:0]	I	E 级传来的控制信号
Res_M[1:0]	I	M 级传来的控制信号
Res_W[1:0]	I	W 级传来的控制信号
A3_E[4:0]	I	E 级指令要写入的寄存器编号
A3_M[4:0]	I	M 级指令要写入的寄存器编号
A3_W[4:0]	I	W 级指令要写入的寄存器编号
A1_D[4:0]	I	D 级指令要读入的寄存器编号 1
A2_D[4:0]	I	D 级指令要读入的寄存器编号 2
A1_E[4:0]	I	E 级指令要读入的寄存器编号 1
A2_E[4:0]	I	E 级指令要读入的寄存器编号 2
A2_M[4:0]	I	M 级指令要读入的寄存器编号
stall	O	暂停信号 1：暂停 0：无效
Fcmp1D[2:0]	O	输出到 D 级的 cmp 编号 1 转发信号 000：RF.V1 001：M_Wdata 010：AO_M 011：PC8_M 100: PC8_E
Fcmp2D[2:0]	O	输出到 D 级的 cmp 编号 2 转发信号 000：RF.V2 001：M_Wdata 010：AO_M 011：PC8_M 100: PC8_E
FaluaE[1:0]	O	输出到 E 级 ALUa 的转发信号 00：V1_E 01：M_Wdata 10：AO_M 11：PC8_M
FalubE[1:0]	O	输出到 E 级 ALUb 的转发信号 00：V2_E 01：M_Wdata 10：AO_M 11：PC8_M
FPCF [2:0]	O	输出到 F 级 PC 的转发信号 000：RF.V1 001：M_Wdata 010：AO_M 011：PC8_M 100: PC8_E
FdmdataM	O	输出到 M 级 DMin 的转发信号 0：V2_M 1：M_Wdata

2. 功能定义

表格 26hazard 功能定义

序号	功能名称	功能描述
1	产生暂停信号	根据 Tuse 和 Tnew 产生暂停信号
2	产生转发信号	根据 Tuse 和 Tnew 产生转发信号

（四） 控制器设计

1. 端口说明

表格 27ctr 端口说明

信号名	方向	描述
instruction[31:0]	I	32 位指令
RegDst	O	grf 写寄存器决定信号 1：rd 0：rt
ALUSrc	O	alu 输入数据 B 决定信号 1：32 位立即数 0：GRF 寄存器 2 输出值
MemtoReg	O	grf 写入数据决定信号 1：DM 输出数据 0：ALU 输出数据
RegWrite	O	grf 写寄存器信号 1：写寄存器 0：无效
MemWrite	O	写内存 dm 信号 1：写入内存 0：无效
nPC_sel	O	跳转信号 Zero 为 1 时： 1：跳转 0：无效 Zero 为 0 时：无效
EXTop[1:0]	O	扩展控制信号 00：无符号扩展 01：低 16 位补 0 10：有符号扩展 11：有符号扩展后逻辑左移两位
ALUctr[2:0]	O	alu 控制信号 000：加运算 001：减运算 010：或运算 011：输出写入数据 2 100:异或运算

2. 真值表

表格 28ctr 真值表

func	100001	100011	001000	001001								
op	000000	000000	000000	000000	001101	100011	101011	000100	001111	000000	001110	000011	000010
	addu	subu	jr	Jarl	ori	lw	sw	beq	lui	nop	xori	jal	j

RegDst	1	1	0	1	0	0	0	0	0	0	0	0	0
ALUSrc	0	0	0	0	1	1	1	0	1	0	1	0	0
MemtoReg	0	0	0	0	0	1	0	0	0	0	0	0	0
RegWrite	1	1	0	1	1	1	0	0	1	0	1	1	0
MemWrite	0	0	0	0	0	0	1	0	0	0	0	0	0
EXTop	00	00	00	00	00	10	10	11	01	00	00	00	00
ALUctr	000	001	000	000	010	000	000	000	011	000	100	000	000

（五） 测试程序

（一） 测试程序

功能测试：

```
.data
.text
#功能测试
        #测试ori指令，第三个立即数是无符号扩展，不存在负数的情况
        ori $a0, $0, 123                #测试与0进行or运算
        ori $a1, $a0, 456               #测试两个非0数的or运算
        ori $0, $0, 123                 #测试写入0号寄存器
        #测试lui指令
        lui $a2, 123                    #测试，构造正数
        lui $a3, 0xffff                 #测试，构造负数
        #测试addu指令，无符号相加，不存在负数情况
        addu $s0, $a0, $a2              #测试正数相加
        #测试subu指令，无符号相减，不存在负数情况
        subu $s1, $a2, $a0              #测试正数相减
        #测试sw指令
        ori $t0, $t0, 0x0000            #构造0
        sw $a0, 0($t0)                  #测试存入内存
        sw $a1, 4($t0)                  #测试存入内存
        #测试lw指令
        lw $s0, 0($t0)                  #测试内存里数写入寄存器
        lw $s1, 4($t0)                  #测试内存里数写入寄存器
        #测试beq指令
        ori $a0, $0, 1                  #构造1
        ori $a1, $0, 2                  #构造2
        ori $a2, $0, 1                  #构造1
```

```
        beq $a0, $a1, 11                #测试不跳转
        nop
        beq $a0, $a2, 12                #测试跳转
        nop
11:ori $a0, $0, 0
12:ori $a1, $0, 0
        #测试jal,jr指令
        ori $s2, $2, 1
        ori $s3, $2, 2
        addu $s4, $s2, $s3
        jal 14
        nop
        jal 15
        nop
        ori $s2, $2, 3
14:     ori $s3, $2, 4
        jr $ra
15:     nop
        #测试jalr
        ori $t4, $0, 0x3090
        jalr $t5, $t4
        nop
        ori $t6, $0, 1

13:     ori $t4, $0, 1
```

转发暂停测试：

```
#转发暂停测试
#初始化内存单元
ori  $t0, $0, 16
sw  $t0, 0($0)
ori  $t0, $0, 432
sw  $t0, 4($0)
ori  $t0, $0, 858
sw  $t0, 8($0)
ori  $t0, $0, 656
sw  $t0, 12($0)
ori  $t0, $0, 8419
sw  $t0, 16($0)
ori  $t0, $0, 3420
sw  $t0, 20($0)
ori  $t0, $0, 29812
sw  $t0, 24($0)
ori  $t0, $0, 1812
sw  $t0, 28($0)
ori  $t0, $0, 213
sw  $t0, 32($0)

#addu
#addu RSuse,RTuse在E
ori  $t1, $0, 516
ori  $t2, $0, 233
```

```
#R-M-RS
subu  $t0, $t1, $t2
addu  $t3, $t0, $t1
#R-M-RT
subu  $t0, $t2, $t1
addu  $t3, $t1, $t0
#R-W-RS
subu  $t0, $t1, $t2
nop
addu  $t3, $t0, $t1
#R-W-RT
subu  $t0, $t2, $t1
nop
addu  $t3, $t1, $t0
#I-M-RS
ori  $t0, $t1, 101
addu  $t3, $t0, $t1
#I-M-RT
ori  $t0, $t1, 102
addu  $t3, $t1, $t0
#I-W-RS
ori  $t0, $t1, 103
nop
addu  $t3, $t0, $t1
#I-W-RT
```

```
ori $t0, $t1, 104
nop
addu $t3, $t1, $t0
#LD-M-RS
lw $t0, 0($0)
addu $t3, $t0, $t1
#LD-M-RT
lw $t0, 4($0)
addu $t3, $t1, $t0
#LD-W-RS
lw $t0, 8($t0)
nop
addu $t3, $t0, $t1
#LD-W-RT
lw $t0, 12($t0)
nop
addu $t3, $t1, $t0
#JAL-M-RS
jal loop1
addu $t2, $ra, $t0
#JAL-M-RT
jal loop2
addu $t2, $t0, $ra
#JAL-W-RS
jal loop1
```

```
nop
addu $t3, $t0, $t1
#I-W-RT
ori $t0, $t1, 104
nop
addu $t3, $t1, $t0
#LD-M-RS
lw $t0, 0($0)
addu $t3, $t0, $t1
#LD-M-RT
lw $t0, 4($0)
addu $t3, $t1, $t0
#LD-W-RS
lw $t0, 8($t0)
nop
addu $t3, $t0, $t1
#LD-W-RT
lw $t0, 12($t0)
nop
addu $t3, $t1, $t0
#JAL-M-RS
jal loop1
addu $t2, $ra, $t0
#JAL-M-RT
jal loop2
```

```
addu $t2, $t0, $ra
#JAL-W-RS
jal loop1
nop
addu $t2, $ra, $t0
#JAL-W-RT
jal loop2
nop
addu $t2, $t0, $ra

#ori
#ori RSuse在EE
#R-M-RS
addu $t0, $t1, $t2
ori $t1, $t0, 233
#R-W-RS
addu $t0, $t1, $t2
nop
ori $t1, $t0, 242
#I-M-RS
lui $t0, 454
ori $t1, $t0, 234
#I-W-RS
lui $t0, 49
nop
```

```
ori $t1, $t0, 34
#LD-M-RS
lw $t0, 16($0)
ori $t0, 342
#LD-W-RS
lw $t0, 20($0)
nop
ori $t0, 984
#JAL-M-RS
jal loop1
ori $t0, $ra, 5995
#JAL-W-RS
jal loop2
nop
ori $t0, $ra, 488

#lw
#lwRSuse在EE
ori $t0, $0, 0
ori $t1, $0, 4
#R-M-RS
addu $t0, $t0, $t1
lw $t3, 0($t0)
#R-W-RS
addu $t0, $t0, $t1
```

```
nop
lw $t3, 0($t0)
#I-M-RS
ori $t0, $0, 16
lw $t3, 0($t0)
#I-W-RS
ori $t0, $0, 20
nop
lw $t3, 0($t0)
#LD-M-RS
lw $t0, 40($0)
lw $t3, 0($t0)
#LD-W-RS
lw $t0, 0($0)  #t0=4
nop
lw $t3, 0($t0)
#SD-M-RS
sw $t3, 36($0)
lw $t4, 36($0)
#SD-W-RS
sw $t3, 40($0)
nop
lw $t4, 40($0)

#beq
```

```
#beq, RSuse/RTuse/ED
#R-E-RS
ori $t0, $0, 0
ori $t1, $0, 0
ori $t2, $0, 1
ori $t3, $0, 2
addu $t0, $t0, $t2
beq $t0, $t1, wrong
nop
#R-E-RT
ori $t0, $0, 0
addu $t1, $t1, $t3
beq $t0, $t1, wrong
nop
#R-M-RS
ori $t0, $0, 0
ori $t1, $0, 0
addu $t0, $t0, $t2
nop
beq $t0, $t1, wrong
nop
#R-M-RT
ori $t0, $0, 0
addu $t1, $t1, $t3
nop
```

```
beq $t0, $t1, wrong
nop
#R-W-RS
ori $t0, $0, 0
ori $t1, $0, 0
addu $t0, $t0, $t2
nop
nop
beq $t0, $t1, wrong
nop
#R-W-RT
ori $t0, $0, 0
addu $t1, $t1, $t3
nop
nop
beq $t0, $t1, wrong
nop
#LD-E-RS
ori $t0, $0, 1
ori $t1, $0, 1
lw $t0, 100($0)
beq $t0, $t1, wrong
nop
#LD-M-RS
ori $t0, $0, 1
```

```
ori $t1, $0, 1
lw $t0, 100($0)
nop
beq $t0, $t1, wrong
nop
#LD-W-RS
ori $t0, $0, 1
ori $t1, $0, 1
lw $t0, 100($0)
nop
nop
beq $t0, $t1, wrong
nop
#JAL-E-RS
ori $t0, $0, 0
ori $ra, $0, 0
jal loop1
beq $t0, $ra, wrong
nop
#JAL-M-RS
ori $ra, $0, 0
jal loop4
nop
loop4:beq $t0, $ra, wrong
nop
```

```
#JAL-W-RS
ori  $ra, $0, 0
jal  loop5
nop
loop5:nop
beq  $t0, $ra, wrong
nop


#sw
#sw, RSuse在EE, RTuse在EM
#R-M-RS
addu  $t0, $0, 4
sw  $t4, 0($t0)
#R-M-RT
addu  $t3, $0, 423
sw  $t3, 0($t0)
#R-W-RS
addu  $t0, $t0, 4
nop
sw  $t3, 0($t0)
#R-W-RT
addu  $t3, $t0, 24214
nop
sw  $t3, 0($t0)
```

```
#I-M-RS
ori  $t0, $0, 16
sw  $t3, 0($t0)
#I-M-RT
ori  $t3, $0, 235
sw  $t4, 0($t0)
#I-W-RS
ori  $t0, $0, 20
nop
sw  $t3, 0($t0)
#I-W-RT
ori  $t3, $0, 9885
nop
sw  $t3, 0($t0)
#LD-M-RS
lw  $t0, 64($0)
sw  $t3, 0($t0)
#LD-M-RT
lw  $t0, 24($0)
sw  $t0, 32($0)
#LD-W-RS
lw  $t0, 80($0)
nop
sw  $t3, 4($t0)
#LD-W-RT
```

```
lw $t3, 32($0)
nop
sw $t3, 0($0)
#JAL-M-RT
jal loop1
sw $ra, 0($0)
#JAL-W-RT
jal loop3
nop
loop3:
sw $ra, 0($0)


j end
nopa

wrong:

loop1:
jr $ra
nop
nop

loop2:
jr $ra
```

```
loop2:
jr $ra
nop
nop
end:
```

（二） 期望输出

```
$ 4 <= 0000007b
$ 5 <= 000001fb
$ 0 <= 0000007b
$ 6 <= 007b0000
$ 7 <= ffff0000
$16 <= 007b007b
$17 <= 007aff85
$ 8 <= 00000000
*00000000 <= 0000007b
*00000004 <= 000001fb
$16 <= 0000007b
$17 <= 000001fb
$ 4 <= 00000001
$ 5 <= 00000002
$ 6 <= 00000001
$ 5 <= 00000000
$18 <= 00000001
$19 <= 00000002
$20 <= 00000003
$31 <= 00003068
$19 <= 00000004
$31 <= 00003070
$12 <= 00003090
$13 <= 0000308c
$12 <= 00000001
$ 8 <= 00000010
*00000000 <= 00000010
$ 8 <= 000001b0
*00000004 <= 000001b0
$ 8 <= 0000035a
*00000008 <= 0000035a
$ 8 <= 00000290
*0000000c <= 00000290
$ 8 <= 000020e3
*00000010 <= 000020e3
$ 8 <= 00000d5c
*00000014 <= 00000d5c
$ 8 <= 00007474
*00000018 <= 00007474
$ 8 <= 00000714
*0000001c <= 00000714
$ 8 <= 000000d5
*00000020 <= 000000d5
$ 9 <= 00000204
$10 <= 000000e9
$ 8 <= 0000011b
$11 <= 0000031f
```

```
$ 8 <= fffffee5
$11 <= 000000e9
$ 8 <= 0000011b
$11 <= 0000031f
$ 8 <= fffffee5
$11 <= 000000e9
$ 8 <= 00000265
$11 <= 00000469
$ 8 <= 00000266
$11 <= 0000046a
$ 8 <= 00000267
$11 <= 0000046b
$ 8 <= 0000026c
$11 <= 00000470
$ 8 <= 00000010
$11 <= 00000214
$ 8 <= 000001b0
$11 <= 000003b4
$ 8 <= 00000000
$11 <= 00000204
$ 8 <= 00000290
$11 <= 00000494
$31 <= 00003164
$10 <= 000033f4
$31 <= 0000316c
$10 <= 000033fc
$31 <= 00003174
$10 <= 00003404
$31 <= 00003180
$10 <= 00003410
$ 8 <= 00003614
$ 9 <= 000036fd
$ 8 <= 00006b0d
$ 9 <= 00006bff
$ 8 <= 01c60000
$ 9 <= 01c600ea
$ 8 <= 00310000
$ 9 <= 00310022
$ 8 <= 000020e3
$ 8 <= 000021f7
$ 8 <= 00000d5c
$ 8 <= 00000fdc
$31 <= 000031c8
$ 8 <= 000037eb
$31 <= 000031d0
$ 8 <= 000031f8
$ 8 <= 00000000
```

```
$ 9 <= 00000004
$ 8 <= 00000004
$11 <= 000001b0
$ 8 <= 00000008
$11 <= 0000035a
$ 8 <= 00000010
$11 <= 000020e3
$ 8 <= 00000014
$11 <= 00000d5c
$ 8 <= 00000000
$11 <= 00000010
$ 8 <= 00000010
$11 <= 000020e3
*00000024 <= 000020e3
$12 <= 000020e3
*00000028 <= 000020e3
$12 <= 000020e3
$ 8 <= 00000000
$ 9 <= 00000000
$10 <= 00000001
$11 <= 00000002
$ 8 <= 00000001
$ 8 <= 00000000
$ 9 <= 00000002
$ 8 <= 00000000
$ 9 <= 00000000
$ 8 <= 00000001
$ 8 <= 00000000
$ 9 <= 00000002
$ 8 <= 00000000
$ 9 <= 00000000
$ 8 <= 00000001
$ 8 <= 00000000
$ 9 <= 00000002
$ 8 <= 00000001
$ 9 <= 00000001
$ 8 <= 00000000
$ 8 <= 00000001
$ 9 <= 00000001
$ 8 <= 00000000
$ 8 <= 00000001
$ 9 <= 00000001
$ 8 <= 00000000
$ 8 <= 00000000
$31 <= 00000000
$31 <= 00003310
$31 <= 00000000
```

```
$31 <= 00003320
$31 <= 00000000
$31 <= 00003334
$ 1 <= 00000000
$ 1 <= 00000004
$ 8 <= 00000004
*00000004 <= 000020e3
$ 1 <= 00000000
$ 1 <= 000001a7
$11 <= 000001a7
*00000004 <= 000001a7
$ 1 <= 00000000
$ 1 <= 00000004
$ 8 <= 00000008
*00000008 <= 000001a7
$ 1 <= 00000000
$ 1 <= 00005e96
$11 <= 00005e9e
*00000008 <= 00005e9e
$ 8 <= 00000010
*00000010 <= 00005e9e
$11 <= 000000eb
*00000010 <= 000020e3
$ 8 <= 00000014
*00000014 <= 000000eb
$11 <= 0000269d
*00000014 <= 0000269d
$ 8 <= 00000000
*00000000 <= 0000269d
$ 8 <= 00007474
*00000020 <= 00007474
$ 8 <= 00000000
*00000004 <= 0000269d
$11 <= 00007474
*00000000 <= 00007474
$31 <= 000033e0
*00000000 <= 000033e0
$31 <= 000033e8
*00000000 <= 000033e8
```

（三） Verilog 输出

```
 45@00003000: $ 4 <= 0000007b
 55@00003004: $ 5 <= 000001fb
 65@00003008: $0 <= 00000000
 75@0000300c: $ 6 <= 007b0000
 85@00003010: $ 7 <= ffff0000
 95@00003014: $16 <= 007b007b
105@00003018: $17 <= 007aff85
115@0000301c: $ 8 <= 00000000
115@00003020: *00000000 <= 0000007b
125@00003024: *00000004 <= 000001fb
145@00003028: $16 <= 0000007b
155@0000302c: $17 <= 000001fb
165@00003030: $ 4 <= 00000001
175@00003034: $ 5 <= 00000002
185@00003038: $ 6 <= 00000001
235@00003050: $ 5 <= 00000000
245@00003054: $18 <= 00000001
255@00003058: $19 <= 00000002
265@0000305c: $20 <= 00000003
275@00003060: $31 <= 00003068
295@00003074: $19 <= 00000004
325@00003068: $31 <= 00003070
355@00003080: $12 <= 00003090
375@00003084: $13 <= 0000308c
395@00003090: $12 <= 00000001
405@00003094: $ 8 <= 00000010
405@00003098: *00000000 <= 00000010
425@0000309c: $ 8 <= 000001b0
425@000030a0: *00000004 <= 000001b0
445@000030a4: $ 8 <= 0000035a
445@000030a8: *00000008 <= 0000035a
465@000030ac: $ 8 <= 00000290
465@000030b0: *0000000c <= 00000290
485@000030b4: $ 8 <= 000020e3
485@000030b8: *00000010 <= 000020e3
505@000030bc: $ 8 <= 00000d5c
505@000030c0: *00000014 <= 00000d5c
525@000030c4: $ 8 <= 00007474
525@000030c8: *00000018 <= 00007474
545@000030cc: $ 8 <= 00000714
545@000030d0: *0000001c <= 00000714
565@000030d4: $ 8 <= 000000d5
565@000030d8: *00000020 <= 0000
585@000030dc: $ 9 <= 00000204
595@000030e0: $10 <= 000000e9
605@000030e4: $ 8 <= 0000011b
615@000030e8: $11 <= 0000031f
625@000030ec: $ 8 <= fffffee5
635@000030f0: $11 <= 000000e9
645@000030f4: $ 8 <= 0000011b
665@000030fc: $11 <= 0000031f
675@00003100: $ 8 <= fffffee5
695@00003108: $11 <= 000000e9
705@0000310c: $ 8 <= 00000265
715@00003110: $11 <= 00000469
725@00003114: $ 8 <= 00000266
735@00003118: $11 <= 0000046a
745@0000311c: $ 8 <= 00000267
765@00003124: $11 <= 0000046b
775@00003128: $ 8 <= 0000026c
795@00003130: $11 <= 00000470
805@00003134: $ 8 <= 00000010
825@00003138: $11 <= 00000214
835@0000313c: $ 8 <= 000001b0
855@00003140: $11 <= 000003b4
865@00003144: $ 8 <= 00000000
885@0000314c: $11 <= 00000204
895@00003150: $ 8 <= 00000290
915@00003158: $11 <= 00000494
925@0000315c: $31 <= 00003164
935@00003160: $10 <= 000033f4
965@00003164: $31 <= 0000316c
975@00003168: $10 <= 000033fc
1005@0000316c: $31 <= 00003174
1045@00003174: $10 <= 00003404
1055@00003178: $31 <= 00003180
1095@00003180: $10 <= 00003410
1105@00003184: $ 8 <= 00003614
1115@00003188: $ 9 <= 000036fd
1125@0000318c: $ 8 <= 00006b0d
```

```
1145@00003194: $ 9 <= 00006bff
1155@00003198: $ 8 <= 01c60000
1165@0000319c: $ 9 <= 01c600ea
1175@000031a0: $ 8 <= 00310000
1195@000031a8: $ 9 <= 00310022
1205@000031ac: $ 8 <= 000020e3
1225@000031b0: $ 8 <= 000021f7
1235@000031b4: $ 8 <= 00000d5c
1255@000031bc: $ 8 <= 00000fdc
1265@000031c0: $31 <= 000031c8
1275@000031c4: $ 8 <= 000037eb
1305@000031c8: $31 <= 000031d0
1345@000031d0: $ 8 <= 000031f8
1355@000031d4: $ 8 <= 00000000
1365@000031d8: $ 9 <= 00000004
1375@000031dc: $ 8 <= 00000004
1385@000031e0: $11 <= 000001b0
1395@000031e4: $ 8 <= 00000008
1415@000031ec: $11 <= 0000035a
1425@000031f0: $ 8 <= 00000010
1435@000031f4: $11 <= 000020e3
1445@000031f8: $ 8 <= 00000014
1465@00003200: $11 <= 00000d5c
1475@00003204: $ 8 <= 00000000
1495@00003208: $11 <= 00000010
1505@0000320c: $ 8 <= 00000010
1525@00003214: $11 <= 000020e3
1525@00003218: *00000024 <= 000020e3
1545@0000321c: $12 <= 000020e3
1545@00003220: *00000028 <= 000020e3
1575@00003228: $12 <= 000020e3
1585@0000322c: $ 8 <= 00000000
1595@00003230: $ 9 <= 00000000
1605@00003234: $10 <= 00000001
1615@00003238: $11 <= 00000002
1625@0000323c: $ 8 <= 00000001
1665@00003248: $ 8 <= 00000000
1675@0000324c: $ 9 <= 00000002
1715@00003258: $ 8 <= 00000000
1725@0000325c: $ 9 <= 00000000
1735@00003260: $ 8 <= 00000001
1775@00003270: $ 8 <= 00000000
1785@00003274: $ 9 <= 00000002
1825@00003284: $ 8 <= 00000000
1835@00003288: $ 9 <= 00000000
1845@0000328c: $ 8 <= 00000001
1895@000032a0: $ 8 <= 00000000
1905@000032a4: $ 9 <= 00000002
1955@000032b8: $ 8 <= 00000001
1965@000032bc: $ 9 <= 00000001
1975@000032c0: $ 8 <= 00000000

2025@000032cc: $ 8 <= 00000001
2035@000032d0: $ 9 <= 00000001
2045@000032d4: $ 8 <= 00000000
2095@000032e4: $ 8 <= 00000001
2105@000032e8: $ 9 <= 00000001
2115@000032ec: $ 8 <= 00000000
2165@00003300: $ 8 <= 00000000
2175@00003304: $31 <= 00000000
2185@00003308: $31 <= 00003310
2235@00003314: $31 <= 00000000
2245@00003318: $31 <= 00003320
2285@00003328: $31 <= 00000000
2295@0000332c: $31 <= 00003334
2345@00003340: $ 1 <= 00000000
2355@00003344: $ 1 <= 00000004
2365@00003348: $ 8 <= 00000004
2365@0000334c: *00000004 <= 000020e3
2385@00003350: $ 1 <= 00000000
2395@00003354: $ 1 <= 000001a7
2405@00003358: $11 <= 000001a7
2405@0000335c: *00000004 <= 000001a7
2425@00003360: $ 1 <= 00000000
2435@00003364: $ 1 <= 00000004
2445@00003368: $ 8 <= 00000008
2455@00003370: *00000008 <= 000001a7
2475@00003374: $ 1 <= 00000000
2485@00003378: $ 1 <= 00005e96
2495@0000337c: $11 <= 00005e9e
2505@00003384: *00000008 <= 00005e9e
```

```
2525@00003388: $ 8 <= 00000010
2525@0000338c: *00000010 <= 00005e9e
2545@00003390: $11 <= 000000eb
2545@00003394: *00000010 <= 000020e3
2565@00003398: $ 8 <= 00000014
2575@000033a0: *00000014 <= 000000eb
2595@000033a4: $11 <= 0000269d
2605@000033ac: *00000014 <= 0000269d
2625@000033b0: $ 8 <= 00000000
2635@000033b4: *00000000 <= 0000269d
2655@000033b8: $ 8 <= 00007474
2655@000033bc: *00000020 <= 00007474
2675@000033c0: $ 8 <= 00000000
2685@000033c8: *00000004 <= 0000269d
2705@000033cc: $11 <= 00007474
2715@000033d4: *00000000 <= 00007474
2735@000033d8: $31 <= 000033e0
2735@000033dc: *00000000 <= 000033e0
2775@000033e0: $31 <= 000033e8
2785@000033e8: *00000000 <= 000033e8
```

（四） 结论

期望输出与实际输出相同。

二、 思考题

（一） 在本实验中你遇到了哪些不同指令组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？请有条理的罗列出来。

cal_r 型

编号	类型	前 序指令	冲突时本指令位置	冲突寄存器	测试样例	解决方法
1	R-M-RS	cal_r	MEM	rs	Addu $1,$2,$3 Addu $4,$1,$5	从 M 级将 AO_M 转发到 E 级部件 alua
2	R-M-RT	cal_r	MEM	rt	Addu $1,$2,$3 Addu $4,$5,$1	从 M 级将 AO_M 转发到 E 级部件 alub
3	R-W-RS	cal_r	WB	rs	Addu $1,$2,$3 nop	从 W 级将 AO_W 转发到 E 级部件 alua

					Addu $4,$1,$5	

4	R-W-RT	cal_r	WB	rt	Addu $1,$2,$3
					nop
					Addu $4,$5,$1
						从 W 级将 AO_W 转发到 E 级部件 alub

5	I-M-RS	cal_i	MEM	rs	Ori $1,$2,1 Addu $3,$1,$4	从 M 级将 AO_M 转发到 E 级部件 alua
6	I-M-RT	cal_i	MEM	rt	Ori $1,$2,1 Addu $3,$4,$1	从 M 级将 AO_M 转发到 E 级部件 alub
7	I-W-RS	cal_i	WB	rs	Ori $1,$2,1 nop Addu $3,$1,$4	从 W 级将 AO_W 转发到 E 级部件 alua
8	I-W-RT	cal_i	WB	rt	Ori $1,$2,1 nop Addu $3,$4,$1	从 W 级将 AO_W 转发到 E 级部件 alub
9	LD-M-RS	load	MEM	rs	Lw $1,0($2) Addu $3,$1,$4	暂停 从 W 级将 DR_W 转发到 E 级部件 alua
10	LD-M-RT	load	MEM	rt	Lw $1,0($2) Addu $3,$4,$1	暂停 从 W 级将 DR_W 转发到 E 级部件 alub

11	LD-W-RS	load	WB	rs	Lw $1,0($2)
					nop
					Addu $3,$1,$4
						从 W 级将 DR_W 转发到 E 级部件 alua

12	LD-W-RT	load	WB	rt	Lw $1,0($2)
					nop
					Addu $3,$4,$1
						从 W 级将 DR_W 转发到 E 级部件 alub

13	JAL-M-RS	jal	MEM	rs	Jal loop Addu $1,$2,$31	从 M 级将 PC8_M 转发到 E 级部件 alua
14	JAL-M-RT	jal	MEM	rt	Jal loop Addu $1,$31,$2	从 M 级将 PC_M 转发到 E 级部件 alub
15	JAL-W-RS	jal	WB	rs	Jal loop nop Addu $1,$2,$31	从 W 级将 PC8_W 转发到 E 级部件 alua
16	JAL-W-RT	jal	WB	rt	Jal loop nop Addu $1,$31,$2	从 W 级将 PC8_W 转发到 E 级部件 alub

cal_i 型

编号	类型	前序指令	冲突时本指令位置	冲突寄存器	测试样例	解决方法
1	R-M-RS	cal_r	MEM	rs	Addu $1,$2,$3 Ori $4,$1,1	从 M 级将 AO_M 转发到 E 级部件 alua
2	R-W-RS	cal_r	WB	rs	Addu $1,$2,$3 nop Ori $4,$1,1	从 W 级将 AO_W 转发到 E 级部件 alua
3	I-M-RS	cal_i	MEM	rs	Ori $1,$2,1	从 M 级将 AO_M 转发到 E 级部件 alua

					Ori $3,$1,1	
4	I-W-RS	cal_i	WB	rs	Ori $1,$2,1 nop Ori $3,$1,1	从 W 级将 AO_W 转发到 E 级部件 alua
5	LD-M-RS	load	MEM	rs	Lw $1,0($2) Ori $3,$1,1	暂停 从 W 级将 DR_W 转发到 E 级部件 alua
6	LD-W-RS	load	WB	rs	Lw $1,0($2) nop Ori $3,$1,1	从 W 级将 DR_W 转发到 E 级部件 alua
7	JAL-M-RS	jal	MEM	rs	Jal loop Ori $2,$31,1	从 M 级将 PC8_M 转发到 E 级部件 alua
8	JAL-W-RS	jal	WB	rs	Jal loop nop Ori $2,$31,1	从 W 级将 PC8_W 转发到 E 级部件 alua

Load 型

编号	类型	前序指令	冲突时本指令位置	冲突寄存器	测试样例	解决方法
1	R-M-RS	cal_r	MEM	rs	Addu $1,$2,$3 Lw $4,0($1)	从 M 级将 AO_M 转发到 E 级部件 alua

2	R-W-RS	cal_r	WB	rs	Addu $1,$2,$3
					nop
					Lw $4,0($1)
						从 W 级将 AO_W 转发到 E 级部件 alua

3	I-M-RS	cal_i	MEM	rs	Ori $1,$2,1 Lw $3,0($1)	从 M 级将 AO_M 转发到 E 级部件 alua
4	I-W-RS	cal_i	WB	rs	Ori $1,$2,1 nop Lw $3,0($1)	从 W 级将 AO_W 转发到 E 级部件 alua

5	LD-M-RS	load	MEM	rs	Lw $1,0($2)
					
					Lw $3,0($1)
						暂停
						
						从 W 级将 DR_W 转发到 E 级部件 alua

6	LD-W-RS	load	WB	rs	Lw $1,0($2)
					
					nop
					
					Lw $3,0($1)
						从 W 级将 DR_W 转发到 E 级部件 alua

7	JAL-M-RS	jal	MEM	rs	Jal loop Lw $2,0($31)	从 M 级将 PC8_M 转发到 E 级部件 alua
8	JAL-W-RS	jal	WB	rs	Jal loop nop Lw $2,0($31)	从 W 级将 PC8_W 转发到 E 级部件 alua

Store 型

编号	类型	前 序指令	冲突时本指令位置	冲突寄存器	测试样例	解决方法

1	R-M-RS	cal_r	MEM	rs	Addu $1,$2,$3
					
					Sw $4,0($1)
						从 M 级将 AO_M 转发到 E 级部件 alua

2	R-W-RT	cal_r	WB	rt	Addu $1,$2,$3
					
					Sw $1,0($4)
						从 W 级将 AO_W 转发到 M 级部件 dmin

3	R-W-RS	cal_r	WB	rs	Addu $1,$2,$3
					nop
					Sw $4,0($1)
						从 W 级将 AO_W 转发到 E 级部件 alua

4	I-M-RS	cal_i	MEM	rs	Ori $1,$2,1 Sw $3,0($1)	从 M 级将 AO_M 转发到 E 级部件 alua
5	I-W-RT	cal_i	WB	rt	Ori $1,$2,1 Sw $1,0($3)	从 W 级将 AO_W 转发到 M 级部件 dmin
6	I-W-RS	cal_i	WB	rs	Ori $1,$2,1 nop Sw $3,0($1)	从 W 级将 AO_W 转发到 E 级部件 alua
7	LD-M-RS	load	MEM	rs	Lw $1,0($2) Sw $3,0($1)	暂停 从 W 级将 DR_W 转发到 E 级部件 alua
8	LD-W-RT	load	WB	rt	Lw $1,0($2) Sw $1,0($3)	从 W 级将 DR_W 转发到 M 级部件 dmin

9	LD-W-RS	load	WB	rs	Lw $1,0($2)
					nop
					Sw $3,0($1)
						从 W 级将 DR_W 转发到 E 级部件 alua

10	JAL-M-RS	jal	MEM	rs	Jal loop Sw $2,0($31)	从 M 级将 PC8_M 转发到 E 级部件 alua
11	JAL-W-RT	jal	WB	rt	Jal loop Sw $31,0($2)	从 W 级将 PC8_W 转发到 M 级部件 dmin

12	JAL-W-RS	jal	WB	rs	Jal loop nop Sw $2,0($31)	从 W 级将 PC8_W 转发到 E 级部件 alua

Beq

编号	类型	前 序指令	冲突时本指令位置	冲 突 寄存器	测试样例	解决方法
1	R-M-RS	cal_r	MEM	rs	Addu $1,$2,$3 Beq $1,$4,loop	暂停 从 M 级将 AO_M 转发到 D 级部件 cmpD1
2	R-M-RT	cal_r	MEM	rt	Addu $1,$2,$3 Beq $4,$1,loop	暂停 从 M 级将 AO_M 转发到 D 级部件 cmpD2
3	R-M-RS	cal_r	MEM	rs	Addu $1,$2,$3 nop Beq $1,$4,loop	从 M 级将 AO_M 转发到 D 级部件 cmpD1
4	R-M-RT	cal_r	MEM	rt	Addu $1,$2,$3 nop Beq $1,$4,loop	从 M 级将 AO_M 转发到 D 级部件 cmpD2
5	I-M-RS	cal_i	MEM	rs	Ori $1,$2,1 Beq $1,$3,loop	暂停 从 M 级将 AO_M 转发到 D 级部件 cmpD1
6	I-M-RT	cal_i	MEM	rt	Ori $1,$2,1 Beq $3,$1,loop	暂停 从 M 级将 AO_M 转发到 D 级部件 cmpD2

7	I-M-RS	cal_i	MEM	rs	Ori $1,$2,1 nop Beq $1,$3,loop	从 M 级将 AO_M 转发到 D 级部件 cmpD1
8	I-M-RT	cal_i	MEM	rt	Ori $1,$2,1 nop Beq $3,$1,loop	从 M 级将 AO_M 转发到 D 级部件 cmpD2
9	LD-M-RS	load	MEM	rs	Lw $1,0($2) Beq $1,$3,loop	暂停 暂停 从 W 级将 DR_W 转发到 D 级部件 cmpD1
10	LD-M-RT	load	MEM	rt	Lw $1,0($2) Beq $3,$1,loop	暂停 暂停 从 W 级将 DR_W 转发到 D 级部件 cmpD2
11	LD-W-RS	load	WB	rs	Lw $1,0($2) nop Beq $1,$3,loop	暂停 从 D 级将 DR_W 转发到 E 级部件 cmpD1
12	LD-W-RT	load	WB	rt	Lw $1,0($2) nop Beq $3,$1,loop	暂停 从 W 级将 DR_W 转发到 D 级部件 cmpD2
13	JAL-E-RS	jal	EX	rs	Jal loop Beq $31,$2,loop	从 E 级将 PC8_E 转发到 D 级部件 cmpD1
14	JAL-E-	jal	EX	rt	Jal loop	从 E 级将 PC8_E 转

	RT				Beq $2,$31,loop	发到D级部件 cmpD2
15	JAL-M-RS	jal	MEM	rs	Jal loop nop Beq $31,$2,loop	从 M 级将 PC8_M 转发到D级部件 cmpD1
16	JAL-M-RT	jal	MEM	rt	Jal loop nop Beq $2,$31,loop	从 M 级将 PC8_M 转发到D级部件 cmpD2

Jr

编号	类型	前 序指令	冲突时本指令位置	冲突寄存器	测试样例	解决方法
1	R-M-RS	cal_r	MEM	rs	Addu $1,$2,$3 Jr $1	暂停 从M级将AO_M转发到D级部件 cmpD1
2	R-M-RS	cal_r	MEM	rs	Addu $1,$2,$3 nop Beq $1,$4,loop	从M级将AO_M转发到D级部件 cmpD1
3	I-M-RS	cal_i	MEM	rs	Ori $1,$2,1 Beq $1,$3,loop	暂停 从M级将AO_M转发到D级部件 cmpD1
4	I-M-RS	cal_i	MEM	rs	Ori $1,$2,1 nop Beq $1,$3,loop	从M级将AO_M转发到D级部件 cmpD1
5	LD-M-RS	load	MEM	rs	Lw $1,0($2)	暂停

					Beq $1,$3,loop	暂停 从 W 级将 DR_W 转发到 D 级部件 cmpD1
6	LD-W-RS	load	WB	rs	Lw $1,0($2) nop Beq $1,$3,loop	暂停 从 W 级将 DR_W 转发到 D 级部件 cmpD1
7	JAL-E-RS	jal	EX	rs	Jal loop Beq $31,$2,loop	从 E 级将 PC8_E 转发到 D 级部件 cmpD1
8	JAL-M-RS	jal	MEM	rs	Jal loop nop Beq $31,$2,loop	从 M 级将 PC8_M 转发到 D 级部件 cmpD1