P5 设计文档

——Verilog 流水线

一、指令集

1. ADDU：不支持溢出的加法

编码	Opcode 000000	rs	rt	rd	Shamt 00000	Func 100001
	6	5	5	5	5	6
描述	GPR[rd] ← GPR[rs] + GPR[rt]					

2. SUBU：不支持溢出的减法

编码	Opcode 000000	rs	rt	rd	Shamt 00000	Func 100011
	6	5	5	5	5	6
描述	GPR[rd] ← GPR[rs] - GPR[rt]					

3. ORI：或立即数

编码	Opcode 001101	rs	rt	immediate
	6	5	5	16
描述	GPR[rt] ← GPR[rs] OR zero_extend(immediate)			

4. LW：加载字

编码	Opcode 100011	base	rt	offset
	6	5	5	16
描述	Addr ← GPR[base] + sign_extend(offset) GPR[rt] ← memory[Addr]			

5. SW：存储字

编码	Opcode 101011	base	rt	offset
	6	5	5	16
描述	Addr ← GPR[base] + sign_extend(offset) memory[Addr] ← GPR[rt]			

6. BEQ：相等时跳转

编码	Opcode 000100	rs	rt	offset
	6	5	5	16

描述	If (GPR[rs] == GPR[rt])
	PC ← PC + 4 + sign_extend(offset||0^2)
	Else
	PC ← PC + 4			

7. LUI：立即数加载至高位

编码	Opcode 001111	0 00000	rt	immediate
	6	5	5	16
描述	GPR[rt] ← immediate||0^16			

8. JAL：跳转并链接

编码	Opcode 000011	index
	6	26
描述	PC ← PC[31:28]||index||00 GPR[31] ← PC+4	

9. J：跳转

编码	Opcode 000010	index
	6	26
描述	PC ← PC[31:28]||index||00	

10. JR：跳转至寄存器

编码	Opcode 000000	rs	0	0	Func 001000
	6	5	10	5	6
描述	PC ← GPR[rs]				

二、模块规格

1. PC：指令地址寄存器（F 级部件）

模块端口定义如下：

信号名	位数	方向	描述
clk		I	内置时钟信号
reset		I	同步复位信号
en		I	写使能信号
PCI	[31:0]	I	下一条指令的地址
PC	[31:0]	O	当前指令的地址

模块功能说明如下：

序号	功能名	描述
1	读指令地址	通过 PC 端口输出当前指令地址
2	写指令地址	时钟上升沿到来时若 en 信号为 1，则更新指令地址
3	同步复位	时钟上升沿到来时若 reset 信号为 1，则复位指令地址至初始状态 0x00003000

2. IM：指令存储器（F 级部件）

模块端口定义如下：

信号名	位数	方向	描述
IMI	[31:0]	I	当前指令地址信号
IM	[31:0]	O	当前指令信号

模块功能说明如下：

序号	功能名	描述
1	读指令	读出指令存储器中 IMI 地址对应的指令并通过 IM 端口输出
2	初始化写指令	初始时向指令存储器中读入所有指令

3. GRF： 寄存器堆（D 级部件，W 级部件）

模块端口定义如下：

信号名	位数	方向	描述
clk		I	内置时钟信号
reset		I	同步复位信号
PC	[31:0]	I	当前指令的地址信号
A1	[4:0]	I	第一个读寄存器地址输入信号
A2	[4:0]	I	第二个读寄存器地址输入信号
A3	[4:0]	I	写寄存器地址输入信号
WD	[31:0]	I	写入数据信号
WE		I	写使能信号
RD1	[31:0]	O	A1 所对应的寄存器的数据信号
RD2	[31:0]	O	A2 所对应的寄存器的数据信号

模块功能说明如下：

序号	功能名	描述
1	读数据	读出 A1，A2 地址对应寄存器数据并通过 RD1，RD2 端口输出
2	写数据	时钟上升沿到来时，若 WE 信号为 1，则向 A3 地址对应寄存器写入数据 WD（0 号寄存器不能被写入）并输出一条与 PC 有关的信息
3	同步复位	时钟上升沿到来时，若 reset 信号为 1，则复位所有寄存器至初始状态 0x00000000

4. EXT：立即数拓展模块（D 级部件）

模块端口定义如下：

信号名	位数	方向	描述
EXTIMM	[15:0]	I	参与拓展的立即数信号
EXTOp	[1:0]	I	拓展方式的选择信号
EXT	[31:0]	O	拓展完成后的数据信号

模块功能说明如下：

序号	功能名	描述
1	符号拓展	若 EXTOp==00，将立即数加载至输出信号低位并用其最高位填充输出信号的高 16 位
2	零拓展	若 EXTOp==01，将立即数加载至输出信号低位并用 0 填充输出信号的高 16 位
3	低位零拓展	若 EXTOp==10，将立即数加载至输出信号高位并用 0 填充输出信号的低 16 位

5. NPC：指令地址计算器（D 级部件）

模块端口定义如下：

信号名	位数	方向	描述
ADD4	[31:0]	I	PC+4 信号
PC4_D	[31:0]	I	F/D 级流水线寄存器中的 PC4 信号
imm_index	[25:0]	I	计算跳转地址的 imm/index 数据信号
M_RD1_D	[31:0]	I	D 级读取 rs 寄存器时的转发信号
ZERO		I	b 类跳转时的相等信号
Is_B		I	b 类指令的判断信号（beq）
Is_J		I	j 类指令的判断信号（j / jal）
PCOp	[1:0]	I	计算下一条指令地址的选择信号
TURE_NPC	[31:0]	O	下一条指令地址

模块功能说明如下：

序号	功能名	描述
1	计算指令地址	若 PCOp==00，输出 PC+4 信号 若 PCOp==01，输出 M_RD1_D 信号 若 PCOp==10 且 is_B==1 且 ZERO==1，则计算 PC4_D+imm 拓展并输出 若 PCOp==10 且 is_J==1，则计算 PC4_D 与 index 拓展信号并输出 若 PCOp==10 且不属于以上情况，则输出 PC4_D+4 信号

6. CMP：数据比较器（D 级部件）

模块端口定义如下：

信号名	位数	方向	描述
D1	[31:0]	I	参与比较的第一个数据信号
D2	[31:0]	I	参与比较的第二个数据信号
CMP		O	比较结果信号

模块功能说明如下：

序号	功能名	描述
1	比较数据	若二数据信号相等，则输出 1，否则输出 0

7. ALU：计算模块（E 级部件）

模块端口定义如下：

信号名	位数	方向	描述
A	[31:0]	I	参与运算的第一个数据信号
B	[31:0]	I	参与运算的第二个数据信号
ALUOp	[2:0]	I	运算方式的选择信号
ALU	[31:0]	O	运算结果的数据信号

模块功能说明如下：

序号	功能名	描述
1	按位与运算	若 ALUOp==000，计算 A&B 并通过 ALU 端口输出
2	按位或运算	若 ALUOp==001，计算 A|B 并通过 ALU 端口输出
3	不支持溢出加法	若 ALUOp==010，计算 A+B 并通过 ALU 端口输出
4	不支持溢出减法	若 ALUOp==011，计算 A-B 并通过 ALU 端口输出

8. DM：数据存储器（M 级部件）

模块端口定义如下：

信号名	位数	方向	描述
clk		I	内置时钟信号
reset		I	同步复位信号
PC	[31:0]	I	当前指令的地址信号
DMA	[31:0]	I	存取的地址信号
DMD	[31:0]	I	存取的数据信号
DMWE		I	存数据使能信号
DM	[31:0]	O	取出的数据信号

模块功能说明如下：

序号	功能名	描述
1	存数据	时钟上升沿到来时，若 DMWE 信号为 1，则向数据存储器 DMA 对应地址中写入 DMD 数据并输出一条与 PC 有关的信息
2	取数据	将数据存储器 DMA 地址对应的数据通过 DM 端口输出
3	同步复位	时钟上升沿到来时，若 reset 信号为 1，则复位数据存储器至初始状态 0x00000000

三、流水线寄存器规格

1. FDreg：F/D 级流水线寄存器

模块端口定义如下：

控制信号	输入信号	输出信号
clk	PC	IR_D
reset	IM	PC_D
en		PC4_D
		PC8_D

2. DEreg：D/E 级流水线寄存器

模块端口定义如下：

控制信号	输入信号	输出信号
clk	IR_D	IR_E
reset	PC_D	PC_E
en	PC4_D	PC4_E
	PC8_D	PC8_E
	RD1	RD1_E
	RD2	RD2_E
	EXT	EXT_E

3. EMreg：E/M 级流水线寄存器

模块端口定义如下：

控制信号	输入信号	输出信号
clk	IR_E	IR_M
reset	PC_E	PC_M
en	PC4_E	PC4_M
	PC8_E	PC8_M
	ALU	ALU_M
	RD2_E	RD2_M

4. MWreg：M/W 级流水线寄存器

模块端口定义如下：

控制信号	输入信号	输出信号
clk	IR_M	IR_W
reset	PC_M	PC_W
en	PC4_M	PC4_W
	PC8_M	PC8_W
	ALU_M	ALU_W
	DM	DM_W

四、冒险处理策略

1. 暂停处理表

F/D			D/E			E/M
类型	寄存器	Tuse	cal_rrd/1	cal_irt/1	load rt/2	load rt/1
cal_r addu subu	rs rt	1			S	
cal_i ori lui	rs	1			S	
load lw	rs	1			S	
store sw	rs	1			S	
store sw	rt	2				
beq	rs rt	0	S	S	S	S
jr	rs	0	S	S	S	S

2. 转发处理表

						D/E	E/M			M/W			
流水级	寄存器	指令	MUX	MUXOp	MUXOp==0	jal 0x1f/0	cal_r rd/0	cal_i rt/0	jal	cal_r rd/0	cal_i rt/0	load rt/0	jal
IR_D	rs	cal_r cal_i beq load store jr	M_RD1_D	M_RD1_D_Op	GRF.RD1	PC8_E	ALU_M	ALU_M	PC8_M	WD	WD	WD	PC8_W
	rt	cal_r store beq	M_RD2_D	M_RD2_D_Op	GRF.RD2	PC8_E	ALU_M	ALU_M	PC8_M	WD	WD	WD	PC8_W
IR_E	rs	cal_r cal_i load store	M_RD1_E	M_RD1_E_Op	RD1_E		ALU_M	ALU_M	PC8_M	WD	WD	WD	PC8_W
	rt	cal_r store	M_RD2_E	M_RD2_E_Op	RD2_E		ALU_M	ALU_M	PC8_M	WD	WD	WD	PC8_W
IR_M	rt	store	M_RD2_M	M_RD2_M_Op	RD2_M					WD	WD	WD	PC8_W
					0	3	1	1	4	2	2	2	5

五、数据通路

1. 数据通路表

级别	部件名	输入端口	addu	subu	lui	ori	lw	sw	beq	j	jal	jr	nop	MUXOp=0	MUXOp=1	MUXOp=2
IF级部件	PC															
	ADD4		PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC			
	IM		PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC			
D更新PC	PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4			
F/D级寄存器	IR_D		IM.IM	IM.IM	IM.IM	IM.IM	IM.IM	IM.IM	IM.IM	IM.IM	IM.IM	IM.IM	IM.IM			
	PC4_D								ADD4	ADD4	ADD4	ADD4	ADD4			
	PC8_D								ADD4+4	ADD4+4	ADD4+4	ADD4+4	ADD4+4			
D级部件	GRF	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]			
		A2	IR_D[rt]	IR_D[rt]				IR_D[rt]	IR_D[rt]				IR_D[rt]			
	EXT				IR_D[imm]	IR_D[imm]	IR_D[imm]	IR_D[imm]					IR_D[imm]			
	CMP	D1							GRF.RD1				M_RD1_D			
		D2							GRF.RD2				M_RD2_D			
	NPC	PC4							PC4_D	PC4_D	PC4_D		PC4_D			
		NPCIMM							IR_D[imm]	IR_D[index]	IR_D[index]		IR_D[imm/index]			
E更新PC	PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	NPC	NPC	NPC	M_RD1_D	PC_MUX	ADD4	M_RD1_D	NPC
D/E级寄存器	IR_E		IR_D	IR_D	IR_D	IR_D	IR_D	IR_D			IR_D		IR_D			
	PC4_E										PC4_D		PC4_D			
	PC8_E										PC8_D		PC8_D			
	RD1_E		GRF.RD1	GRF.RD1		GRF.RD1	GRF.RD1	GRF.RD1					M_RD1_D			
	RD2_E		GRF.RD2	GRF.RD2				GRF.RD2					M_RD2_D			
	EXT_E				EXT.EXT	EXT.EXT	EXT.EXT	EXT.EXT					EXT.EXT			
E级部件	ALU	ALUA	RD1_E	RD1_E		RD1_E	RD1_E	RD1_E					M_RD1_E			
		ALUB	RD2_E	RD2_E	EXT_E	EXT_E	EXT_E	EXT_E					ALUB_MUX	M_RD2_E	EXT_E	
	XALU	D1											/			
		D2											/			
E/M级寄存器	IR_M		IR_E	IR_E	IR_E	IR_E	IR_E	IR_E			IR_E		IR_E			
	PC4_M										PC4_E		PC4_E			
	PC8_M										PC8_E		PC8_E			
	ALUout_M		ALU.ALU	ALU.ALU	ALU.ALU	ALU.ALU	ALU.ALU	ALU.ALU					ALU.ALU			
	XALUout_M												/			
	RD2_M							RD2_E					M_RD2_E			
M级部件	DM	DMA					ALUout_M	ALUout_M					ALUout_M			
		DMD						RD2_M					M_RD2_M			
M/W级寄存器	IR_W		IR_M	IR_M	IR_M	IR_M	IR_M				IR_M		IR_M			
	PC4_W										PC4_M		PC4_M			
	PC8_W										PC8_M		PC8_M			
	ALUout_W		ALUout_M	ALUout_M	ALUout_M	ALUout_M							ALUout_M			
	XALUout_W												/			
	DMout_W						DM.DM						DM.DM			
W级部件	GRF	A3	IR_W[rd]	IR_W[rd]	IR_W[rt]	IR_W[rt]	IR_W[rt]				0x1f		GRFA3_MUX	IR_W[rt]	IR_W[rd]	0x1f
		WD	ALUout_W	ALUout_W	ALUout_W	ALUout_W	DMout_W				PC8_W		GRFWD_MUX	ALUout_W	DMout_W	PC8_W

六、控制器设计

1. 控制信号说明

序号	信号名	位数	描述
1	ALUOp	[2:0]	ALU 计算方式的选择信号
2	GRFA3_MUXOp	[1:0]	GRFA3 口连接信号的选择信号
3	ALUB_MUXOp		ALUB 口连接信号的选择信号
4	GRFWE		GRF 的写使能信号
5	DMWE		DM 的写使能信号
6	GRFWD_MUXOp	[1:0]	GRFWD 口连接信号的选择信号
7	EXTOp	[1:0]	EXT 拓展方式的选择信号
8	Is_B		B 类指令（beq）的特征信号
9	Is_J		J 类指令（j/jal）的特征信号
10	PCOp	[1:0]	PC 计算方式的选择信号

2. 指令与控制信号真值表

func	100001	100011								001000	000000
opcode	000000	000000	001101	001111	100011	101011	000100	000010	000011	000000	000000
指令名	addu	subu	ori	lui	lw	sw	beq	j	jal	jr	nop
GRFA3_MUXOp	01	01	00	00	00	00	00	00	10	01	00
ALUB_MUXOp	0	0	1	1	1	1	0	0	0	0	0
GRFWE	1	1	1	1	1	0	0	0	1	1	0
DMWE	0	0	0	0	0	1	0	0	0	0	0
GRFWD_MUXOp	00	00	00	00	01	00	00	00	10	00	00
EXTOp	00	00	01	10	00	00	00	00	00	00	00
Is_B	0	0	0	0	0	0	1	0	0	0	0
ALUOp	010	011	001	010	010	010	011	000	000	000	000
Is_J	0	0	0	0	0	0	0	1	1	0	0
PCOp	00	00	00	00	00	00	010	10	10	01	00

七、测试样例

1. 暂停组合

（1） D: cal_r　　E: load

```
ori $t1, 1
ori $t2, 2
addu $s1, $t1, $t2
sw $s1, 0($0)
addu $s1, $s1, $s1
nop
nop
nop
nop
nop
lw $s1, 0($0)
addu $s1, $s1, $s1
```

```
@00003000: $ 9 <= 00000001
@00003004: $10 <= 00000002
@00003008: $17 <= 00000003
@0000300c: *00000000 <= 00000003
@00003010: $17 <= 00000006
@00003028: $17 <= 00000003
@0000302c: $17 <= 00000006
```

（2） D: cal_i　　E: load

```
ori $t1, 1
ori $t2, 2
addu $s1, $t1, $t2
sw $s1, 0($0)
addu $s1, $s1, $s1
nop
nop
nop
nop
nop
lw $s1, 0($0)
ori $s1, $s1, 0
```

```
@00003000: $ 9 <= 00000001
@00003004: $10 <= 00000002
@00003008: $17 <= 00000003
@0000300c: *00000000 <= 00000003
@00003010: $17 <= 00000006
@00003028: $17 <= 00000003
@0000302c: $17 <= 00000003
```

（3） D: load　　E: load

```
ori $t1, 2
ori $t2, 2
addu $s1, $t1, $t2
sw $s1, 0($0)
sw $s1, 4($0)
ori $s1, $0, 8
nop
nop
nop
nop
nop
lw $s1, 0($0)
lw $s2, 0($s1)
```

```
@00003000: $ 9 <= 00000002
@00003004: $10 <= 00000002
@00003008: $17 <= 00000004
@0000300c: *00000000 <= 00000004
@00003010: *00000004 <= 00000004
@00003014: $17 <= 00000008
@0000302c: $17 <= 00000004
@00003030: $18 <= 00000004
```

（4） D: store　　E: load

```
ori $t1, 2
ori $t2, 2
addu $s1, $t1, $t2
sw $s1, 0($0)
sw $s1, 4($0)
ori $s1, $0, 8
nop
nop
nop
nop
nop
lw $s1, 0($0)
sw $s2, 0($s1)
```

```
@00003000: $ 9 <= 00000002
@00003004: $10 <= 00000002
@00003008: $17 <= 00000004
@0000300c: *00000000 <= 00000004
@00003010: *00000004 <= 00000004
@00003014: $17 <= 00000008
@0000302c: $17 <= 00000004
@00003030: *00000004 <= 00000000
```

(5) D: beq E: cal_r

```
ori $t1, 2
ori $t2, 2
ori $s2, 4
nop
nop
nop
nop
nop
addu $s1, $t1, $t2
beq $s1, $s2, end
nop
ori $v0, 1
end:
ori $v0, 10
```

```
@00003000: $ 9 <= 00000002
@00003004: $10 <= 00000002
@00003008: $18 <= 00000004
@00003020: $17 <= 00000004
@00003030: $ 2 <= 0000000a
```

(6) D: beq E: cal_i

```
ori $t1, 2
ori $t2, 2
ori $s2, 4
nop
nop
nop
nop
nop
ori $s1, 4
beq $s1, $s2, end
nop
ori $v0, 1
end:
ori $v0, 10
```

```
@00003000: $ 9 <= 00000002
@00003004: $10 <= 00000002
@00003008: $18 <= 00000004
@00003020: $17 <= 00000004
@00003030: $ 2 <= 0000000a
```

(7) D: beq E: load

```
ori $t1, 2
ori $t2, 2
ori $s2, 4
sw $s2, 0($0)
nop
nop
nop
nop
nop
lw $s1, 0($0)
beq $s1, $s2, end
nop
ori $v0, 1
end:
ori $v0, 10
```

```
@00003000: $ 9 <= 00000002
@00003004: $10 <= 00000002
@00003008: $18 <= 00000004
@0000300c: *00000000 <= 00000004
@00003024: $17 <= 00000004
@00003034: $ 2 <= 0000000a
```

（8） D: beq　　M: load

```
ori $t1, 2
ori $t2, 2
ori $s2, 4
sw $s2, 0($0)
nop
nop
nop
nop
nop
lw $s1, 0($0)
addu $s2, $s2, $0
beq $s1, $s2, end
nop
ori $v0, 1
end:
ori $v0, 10
```

```
@00003000: $ 9 <= 00000002
@00003004: $10 <= 00000002
@00003008: $18 <= 00000004
@0000300c: *00000000 <= 00000004
@00003024: $17 <= 00000004
@00003028: $18 <= 00000004
@00003038: $ 2 <= 0000000a
```

（9） D: jr　　E: cal_r

```
ori $t1, 2
ori $t2, 2
jal begin
addu $s1, $t1, $s1
addu $s2, $t1, $s2
addu $s3, $t1, $s3
nop
nop
nop
nop
beq $0, $0, end
begin:
addu $ra, $ra, 4
jr $ra
end:
nop
```

```
@00003000: $ 9 <= 00000002
@00003004: $10 <= 00000002
@00003008: $31 <= 00003010
@0000300c: $17 <= 00000002
@0000302c: $ 1 <= 00000000
@00003030: $ 1 <= 00000004
@00003034: $31 <= 00003014
@00003014: $19 <= 00000002
@0000302c: $ 1 <= 00000000
```

（10） D: jr E: cal_i

略

（11） D: jr E: load

略

（12） D: jr M: load

略

2. 转发组合

（1） D: cal_r E: jal R: rs

```
jal begin
addu $31, $31, $31
begin:
nop
```

```
@00003000: $31 <= 00003008
@00003004: $31 <= 00006010
```

（2） D: cal_i E: jal R: rs

```
jal begin
ori $1, $31, 0
begin:
nop
```

```
@00003000: $31 <= 00003008
@00003004: $ 1 <= 00003008
```

（3） D: beq E: jal R: rs

Undefined Behaviour

（4） D: load E: jal R: rs

略

（5） D: store E: jal R: rs

略

（6） D: jr E: jal R: rs

Undefined Behaviour

（7） D: cal_r E: jal R: rt

```
jal begin
addu $31, $31, $31
begin:
nop
```

```
@00003000: $31 <= 00003008
@00003004: $31 <= 00006010
```

(8) D: store E: jal R: rt

略

(9) D: beq E: jal R: rt

Undefined Behaviour

(10) D: cal_r M: cal_r R: rs

```
ori $t1, $0, 1
nop
nop
nop
nop
nop
addu $s1, $s1, $t1
nop
addu $s2, $s1, $s1
```

```
@00003000: $ 9 <= 00000001
@00003018: $17 <= 00000001
@00003020: $18 <= 00000002
```

(11) D: cal_i M: cal_r R: rs
(12) D: beq M: cal_r R: rs
(13) D: load M: cal_r R: rs
(14) D: store M: cal_r R: rs
(15) D: jr M: cal_r R: rs
(16) D: cal_r M: cal_r R: rt
(17) D: store M: cal_r R: rt
(18) D: beq M: cal_r R: rt
(19) D: cal_r M: cal_i R: rs
(20) D: cal_i M: cal_i R: rs
(21) D: beq M: cal_i R: rs
(22) D: load M: cal_i R: rs
(23) D: store M: cal_i R: rs
(24) D: jr M: cal_i R: rs
(25) D: cal_r M: cal_i R: rt
(26) D: store M: cal_i R: rt
(27) D: beq M: cal_i R: rt
(28) D: cal_r M: jal R: rs
(29) D: cal_i M: jal R: rs
(30) D: beq M: jal R: rs
(31) D: load M: jal R: rs
(32) D: store M: jal R: rs
(33) D: jr M: jal R: rs
(34) D: cal_r M: jal R: rt
(35) D: store M: jal R: rt
(36) D: beq M: jal R: rt

(37) D: cal_r W: cal_r R: rs
(38) D: cal_i W: cal_r R: rs
(39) D: beq   W: cal_r R: rs
(40) D: load   W: cal_r R: rs
(41) D: store W: cal_r R: rs
(42) D: jr   W: cal_r R: rs
(43) D: cal_r W: cal_r R: rt
(44) D: store W: cal_r R: rt
(45) D: beq   W: cal_r R: rt
(46) D: cal_r W: cal_i R: rs
(47) D: cal_i W: cal_i R: rs
(48) D: beq   W: cal_i R: rs
(49) D: load   W: cal_i R: rs
(50) D: store W: cal_i R: rs
(51) D: jr   W: cal_i R: rs
(52) D: cal_r W: cal_i R: rt
(53) D: store W: cal_i R: rt
(54) D: beq   W: cal_i R: rt
(55) D: cal_r W: load   R: rs
(56) D: cal_i W: load   R: rs
(57) D: beq   W: load   R: rs
(58) D: load   W: load   R: rs
(59) D: store W: load   R: rs
(60) D: jr   W: load   R: rs
(61) D: cal_r W: load   R: rt
(62) D: store W: load   R: rt
(63) D: beq   W: load   R: rt
(64) D: cal_r W: jal   R: rs
(65) D: cal_i W: jal   R: rs
(66) D: beq   W: jal   R: rs
(67) D: load   W: jal   R: rs
(68) D: store W: jal   R: rs
(69) D: jr   W: jal   R: rs
(70) D: cal_r W: jal   R: rt
(71) D: store W: jal   R: rt
(72) D: beq   W: jal   R: rt
(73) E: cal_r M: cal_r R: rs
(74) E: cal_i M: cal_r R: rs
(75) E: load   M: cal_r R: rs
(76) E: store M: cal_r R: rs
(77) E: cal_r M: cal_r R: rt
(78) E: store M: cal_r R: rt
(79) E: cal_r M: cal_i R: rs
(80) E: cal_i M: cal_i R: rs

(81) E: load M: cal_i R: rs
(82) E: store M: cal_i R: rs
(83) E: cal_r M: cal_i R: rt
(84) E: store M: cal_i R: rt
(85) E: cal_r M: jal R: rs
(86) E: cal_i M: jal R: rs
(87) E: load M: jal R: rs
(88) E: store M: jal R: rs
(89) E: cal_r M: jal R: rt
(90) E: store M: jal R: rt
(91) E: cal_r W: cal_r R: rs
(92) E: cal_i W: cal_r R: rs
(93) E: load W: cal_r R: rs
(94) E: store W: cal_r R: rs
(95) E: cal_r W: cal_r R: rt
(96) E: store W: cal_r R: rt
(97) E: cal_r W: cal_i R: rs
(98) E: cal_i W: cal_i R: rs
(99) E: load W: cal_i R: rs
(100) E: store W: cal_i R: rs
(101) E: cal_r W: cal_i R: rt
(102) E: store W: cal_i R: rt
(103) E: cal_r W: load R: rs
(104) E: cal_i W: load R: rs
(105) E: load W: load R: rs
(106) E: store W: load R: rs
(107) E: cal_r W: load R: rt
(108) E: store W: load R: rt
(109) E: cal_r W: jal R: rs
(110) E: cal_i W: jal R: rs
(111) E: load W: jal R: rs
(112) E: store W: jal R: rs
(113) E: cal_r W: jal R: rt
(114) E: store W: jal R: rt
(115) M: store W: cal_r R: rt
(116) M: store W: cal_i R: rt
(117) M: store W: load R: rt
(118) M: store W: jal R: rt

思考题

1. 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？

本实验中指令根据其所需要读写的寄存器位置（rs/rt）的不同可分为以下八类：cal_r（addu/subu）、cal_i（ori/lui）、store（sw）、load（lw）、beq、j、jal、jr。在指令需要用到 RD1/RD2 的值时，若此时该数据未完成回写/未产生，则发生了冲突，分别需要用转发/暂停来解决。具体冲突组合见设计文档中的冒险处理策略部分，相应测试样例见设计文档中的测试样例部分。