计算机组成原理 P6 实验报告

一、 模块详述

1. PC 模块

表 1 PC 模块功能表

序号	功能名称	功能描述
1	clk	时钟信号。当 clk 上升沿到来时，PC 寄存器完成更新
2	reset	复位信号。当其为 1 时，在时钟上升沿将 PC 寄存器复位，置为 0x00003000
3	shall	当其为 1 时，PC 寄存器暂停更新
4	newpc[31:0]	下一次更新的 PC 值
5	pc[31:0]	输出当前 PC 值

2. IM 模块

表 2 IM 功能表格

序号	功能名称	功能描述
1	pc[31:0]	输入当前 PC 的值
2	instr[31:0]	输出当前指令

3. F 模块

表 3 F 模块功能表格

序号	功能名称	功能描述
1	clk	时钟信号。当其上升沿到来时完成 pc 的更新
2	reset	复位信号。时钟上升沿到来时若为 1，pc 重置
3	npc[31:0]	输入 D 级流水线跳转指令的下一个 pc
4	nPc_sel	当其为 1，pc 更新来自于 npc；为 0，则为 pc+4
5	shall	若其为 1，则 pc 暂停更新
6	instr[31:0]	输出当前 IM 指令
7	pc[31:0]	输出当前 pc 的值
8	pc4[31:0]	输出当前 pc+4 的值

4. GRF 模块

表 4 GRF 功能表格

序号	功能名称	功能描述
1	reset	复位信号。当其为 1 时，在时钟上升沿来临时将所有寄存器值置为 0
2	RA1[4:0]	要读取的第一个寄存器的地址
3	RD1[31:0]	输出第一个寄存器的值
4	RA2[4:0]	要读取的第二个寄存器的地址

5	RD2[31:0]	输出第二个寄存器的值
6	WA[4:0]	要写入的寄存器的地址
7	WD[31:0]	写入寄存器的值
8	RegWrite	表示是否将 WD 的值写入寄存器。若为 1，则写入，否则不写入
9	clk	时钟信号。当 clk 上升沿来临时完成对寄存器堆的写入
10	pc[31:0]	输入当前 pc 的值

5. EXT 模块

表 5 EXT 功能表格

序号	功能名称	功能描述
1	imm[15:0]	输入被拓展的 16 位数
2	extout[32:0]	输出拓展后的 32 位数
3	ExtOp	拓展控制信号。若为 1，则进行符号拓展，否则进行无符号拓展
4	iflui	lui 指令判断信号。若为 1，则表示执行 lui 指令，此时将 16 位 in 后拼接 16 位 0 输出

6. CMP 模块

表 6 CMP 功能表格

序号	功能名称	功能描述
1	D1[15:0]	输入被比较的第一个数
2	D2[15:0]	输入被比较的第二个数
3	result	输出比较的结果
4	CMPOp[31:0]	比较器比较运算控制信号

7. NPC 模块

表 7 NPC 功能表格

序号	功能名称	功能描述
1	pc[31:0]	输入当前 pc 值
2	pc4[31:0]	输入当前 pc+4 的值
3	instr[31:0]	输入当前指令
4	offset32[31:0]	输入 32 位 jr 指令跳转偏移量
5	ifj	若为 1，则表示是 j 类指令
6	ifjr	若为 1，则表示是 jr 指令
7	ifb	若为 1，则表示是 b 指令
8	branch	若为 1，则跳转
9	npc[31:0]	输出需要跳转时 32 位的 pc 值

8. D 模块

表 8 D 模块功能表

序号	功能名称	功能描述
1	clk	时钟信号。当 clk 上升沿到来时，更新该级流水线寄存器
2	reset	复位信号。当时钟上升沿来临时，若其为 1，则更新该流水级所有寄存器
3	instr[31:0]	输入来自 F 级的指令
4	pc4[31:0]	输入来自 F 级的 pc+4 的值
5	pc[31:0]	输入来自 F 级的 pc 值
6	WA[4:0]	输入写入寄存器的地址
7	WD[31:0]	输入写入寄存器的数据
8	RegWrite	寄存器写入控制信号，若为 1 则写入
9	ExtOp	拓展控制信号，若为 1 则进行符号拓展；否则进行无符号拓展
10	iflui	若为 1 则表示当前指令为 lui
11	ifj	若为 1 则表示当前指令为 j 或 jal
12	ifjr	若为 1 则表示当前指令为 jr
13	ifb	若为 1 则表示当前指令为 b 类型
14	shall	若为 1，则暂停对该流水级寄存器的更新
15	write_pc[31:0]	输入写入寄存器的 pc 值
16	RD1[31:0]	输出读寄存器的第一个数值
17	RD2[31:0]	输出读寄存器的第二个数值
18	ext[31:0]	输出拓展后的 32 位数
19	instr_out[31:0]	输出该流水级指令
20	pc4_out[31:0]	输出该流水级 pc+4
21	pc_out[31:0]	输出该流水级 pc
22	npc[31:0]	输出 npc
23	AO_M[31:0]	输入 M 级流水线 AO 寄存器的值
24	PC4_M[31:0]	输入 M 级流水线 PC4 寄存器的值
25	mf_CMP_A[31:0]	CMP 第一个数的转发控制信号
26	mf_CMP_B[31:0]	CMP 第二个数的转发控制信号
27	mf_JR[31:0]	jr 指令转发控制信号
28	CMPOp[31:0]	比较器比较运算的选择信号

9. Con_D 模块

表 9 Con_D 模块功能表

序号	功能名称	功能描述
1	instr[31:0]	输入 D 级流水线当前指令
2	nPc_sel	F 级流水线 pc 跳转指令控制信号，为 1 来自于 npc；否则来自于 pc+4
3	ExtOp	拓展信号，为 1 则进行符号拓展，为 0 进行无符号拓展
4	ifb	为 1 表示当前执行 b 指令
5	iflui	为 1 表示当前执行 lui 指令

6	ifj	为 1 表示当前执行 j 或 jal 指令
7	ifjr	为 1 表示当前指令 jr 指令

10. ALU 模块

表 10 ALU 模块功能表

序号	功能名称	功能描述
1	A[31:0]	输入运算的第一个 32 位数
2	B[31:0]	输入运算的第二个 32 位数
3	ALUOp[31:0]	运算选择信号。
4	result[31:0]	输出 32 位运算结果
5	s[4:0]	输入指令的 10-6 位，用作左移右移

11. E 模块

表 11 E 模块功能表

序号	功能名称	功能描述
1	clk	时钟信号。当 clk 上升沿到来时，该级流水线寄存器完成更新
2	reset	复位信号。当时钟上升沿来临时，若其为 1，则更新该流水级所有寄存器
3	instr[31:0]	输入来自 D 级的指令
4	pc4[31:0]	输入来自 D 级的 pc+4 的值
5	pc[31:0]	输入来自 D 级的 pc 值
6	RS[31:0]	输入来自 D 级的 RS 值
7	RT[31:0]	输入来自 D 级的 RT 值
8	ext[31:0]	输入来自 D 级的 ext 值
9	ALUOp[1:0]	ALU 运算选择信号
10	ALUSrc	ALU 的第二个操作数的选择信号
11	shall	暂停信号，若为 1，清除 IR_E
12	instr_out[31:0]	输出该流水级指令
13	pc4_out[31:0]	输出该流水级 pc+4
14	pc_out[31:0]	输出该流水级 pc
15	aluout[31:0]	输出该流水级 ALU 计算的值
16	rt_out[31:0]	输出该流水级 rt 的值
17	AO_M[31:0]	输入 M 级流水线 AO 的值
18	PC4_M[31:0]	输入 M 级流水线 PC4 的值
19	WD_W[31:0]	输入 W 级流水线 WD 的值
20	PC4_W	输入 W 级流水线 PC4 的值
21	mf_ALU_A[31:0]	ALU 第一个操作数转发控制信号
22	mf_ALU_B[31:0]	ALU 第二个操作数转发控制信号
23	mf_RTout_E[31:0]	输出的 rt 的值的转发控制信号
24	MDOp[1:0]	乘除模块的运算选择信号
25	HIWrite	HI 寄存器写入信号

26	LOWrite	LO 寄存器写入信号
27	MDSrc	HI，LO 寄存器写入寄存器的选择信号
28	AluoutSrc[1:0]	输出到下一级的 AO_M 的选择信号

12. Con_E 模块

表 12 Con_E 模块功能表

序号	功能名称	功能描述
1	instr[31:0]	输入 E 级流水线当前指令
2	ALUOp[31:0]	输出 ALU 的控制信号
3	ALUSrc	输出 ALU 第二个操作数的控制信号
4	MDOp[1:0]	乘除模块的运算选择信号
5	HIWrite	HI 寄存器写入信号
6	LOWrite	LO 寄存器写入信号
7	MDSrc	HI，LO 寄存器写入寄存器的选择信号
8	AluoutSrc[1:0]	输出到下一级的 AO_M 的选择信号

13. DM 模块

表 13 DM 功能表格

序号	功能名称	功能描述
1	Address[31:0]	读入或者写入 dm 的 32 位地址
2	WD[31:0]	写入 dm 的 32 位数据
3	Data[31:0]	读出 dm 的 32 位数据
4	MemWrite	dm 写入控制信号。若为 1，则将 WD 的内容写入
5	reset	复位信号。当其为 1 时，当时钟上升沿来临时将 dm 的所有数据清零
6	clk	时钟信号。当 clk 上升沿来临时，完成对 dm 的写入
7	pc[31:0]	输入当前 PC 的值
8	DMin_Src[31:0]	DM 写入数据的来源，也就是 sh，sb 的选择信号
9	DMout_Src[31:0]	DM 输出的数据来源，也就是 lh，lb 等的选择信号

14. M 模块

表 14 M 模块功能表

序号	功能名称	功能描述
1	clk	时钟信号。当 clk 上升沿到来时，该级流水线寄存器完成更新
2	reset	复位信号。当时钟上升沿来临时，若其为 1，则更新该流水级所有寄存器
3	instr[31:0]	输入来自 E 级的指令
4	pc4[31:0]	输入来自 E 级的 pc+4 的值

5	pc[31:0]	输入来自 E 级的 pc 值
6	aluout[31:0]	输入来自 E 级的 alu 的值
7	rt[31:0]	输入来自 E 级的 rt 的值
8	MemWrite	数据存储器堆的写入信号
9	instr_out[31:0]	输出 M 级指令
10	pc_out[31:0]	输出 M 级 PC 值
11	pc4_out[31:0]	输出 M 级 PC4 的值
12	alu_out[31:0]	输出 M 级 ALU 的值
13	dmout[31:0]	输出 M 级 DM 的值
14	WD_W[31:0]	输入 W 级 WD 的值
15	PC4_W[31:0]	输入 W 级 PC4_W 的值
16	mf_DMin_M	DM 写入数据来源的转发控制信号
17	DMin_Src[31:0]	DM 写入数据的来源，也就是 sh，sb 的选择信号
18	DMout_Src[31:0]	DM 输出的数据来源，也就是 lh，lb 等的选择信号

15. Con_M 模块

表 15 Con_M 模块功能表

序号	功能名称	功能描述
1	instr[31:0]	输入 M 级流水线当前指令
2	MemWrite	数据存储器写入控制信号，为 1 则写入 DM
3	DMin_Src[31:0]	DM 写入数据的来源，也就是 sh，sb 的选择信号
4	DMout_Src[31:0]	DM 输出的数据来源，也就是 lh，lb 等的选择信号

16. W 模块

表 16 W 模块功能表

序号	功能名称	功能描述
1	clk	时钟信号。当 clk 上升沿到来时，该级流水线寄存器完成更新
2	reset	复位信号。当时钟上升沿来临时，若其为 1，则更新该流水级所有寄存器
3	instr[31:0]	输入来自 M 级的指令
4	pc4[31:0]	输入来自 M 级的 pc+4 的值
5	pc[31:0]	输入来自 M 级的 pc 值
6	ao[31:0]	输入来自 M 级的 ALU 的值
7	dr[31:0]	输入来自 M 级的 DM 的值
8	RegDst	写入寄存器地址的控制信号
9	ifjal	表示是否的 jal 信号，若是则为 1
10	MemtoReg	写入寄存器数据的控制信号
11	WA[4:0]	输出寄存器堆的写入地址

12	WD[4:0]	输出寄存器堆的写入数据
13	instr_out[31:0]	输出 W 级的指令
14	pc_out[31:0]	输出 W 级的 pc 值

17. Con_W 模块

表 17 Con_W 模块功能表

序号	功能名称	功能描述
1	instr[31:0]	输入 W 级流水线当前指令
2	RegDst	写入寄存器堆地址来源的控制信号
3	ifjal	判断是否为 jal 指令，若是则为 1
4	MemtoReg	写入寄存器堆数据来源的控制信号
5	RegWrite	寄存器堆写入控制信号，为 1 则写入

18. Stall 模块

表 18 Stall 模块功能表

序号	功能名称	功能描述
1	instr_D[31:0]	输入来自 D 级流水线的指令
2	instr_E[31:0]	输入来自 E 级流水线的指令
3	instr_M[31:0]	输入来自 M 级流水线的指令
4	shall	输出是否暂停的控制信号

19. MOVE 模块

表 19 MOVE 模块功能表

序号	功能名称	功能描述
1	instr_D[31:0]	输入来自 D 级流水线的指令
2	instr_E[31:0]	输入来自 E 级流水线的指令
3	instr_M[31:0]	输入来自 M 级流水线的指令
4	instr_W[31:0]	输入来自 W 级流水线的指令
5	mf_ALU_A[31:0]	ALU 第一个操作数转发控制信号
6	mf_ALU_B[31:0]	ALU 第二个操作数转发控制信号
7	mf_RTout_E[31:0]	输出的 rt 的值的转发控制信号
8	mf_CMP_A[31:0]	CMP 第一个数的转发控制信号
9	mf_CMP_B[31:0]	CMP 第二个数的转发控制信号
10	mf_DMin_M[31:0]	DM 输入的数据来源转发的控制信号
11	mf_JR[31:0]	JR 指令转发控制信号

20. MD 模块

表 20 MD 模块功能表

序号	功能名称	功能描述
1	clk	时钟信号。当其时钟上升沿来临时，完成 HI，LO 寄存器更新
2	reset	复位信号。时钟上升沿当其为 1 时，清空 HI，LO 寄存器
3	A[31:0]	乘除运算的第一个操作数
4	B[31:0]	乘除运算的第二个操作数
5	MDOp[1:0]	乘除模块的运算选择信号
6	HIWrite	HI 寄存器写入信号
7	LOWrite	LO 寄存器写入信号
8	MDSrc	HI，LO 寄存器写入寄存器的选择信号
9	HI[31:0]	输出 HI 寄存器的值
10	LO[31:0]	输出 LO 寄存器的值

21. Busy 模块

表 21 Busy 模块功能表

序号	功能名称	功能描述
1	clk	时钟信号。当其上升沿来临时，完成对计数寄存器的更新
2	reset	复位信号。时钟上升沿来临时若其为 1，清空计数寄存器
3	instr[31:0]	输入来自 D 级流水线的指令
4	busy	输出 busy 信号。若其为 1，则暂停流水线

22. 部分控制信号

ALUOp	0	1	2	3	4	5	6	7	8
	加法	减法	sll	srl	sra	sllv	srlv	srav	and
	9	10	11	12	13				
	or	xor	nor	slt	sltu				
	0	1	2	3	4	5	6		
CMPOp	其他指令	beq	bne	bgez	bgtz	blez	bltz		
	0	1	10	11					
MDOp	mult	multu	div	divu					
	0	1	10						
AluoutSrc	alu	hi	lo						
	0	1	2						
DMin_Src	sw	sh	sb						
	0	1	2	3	4				
DMout_Src	lw	lh	lhu	lb	lbu				

图 1 部分控制信号

23. 转发信号设计

D级当前指令			E				M				W			
指令类型	源寄存器	Tuse	Tnew				Tnew				Tnew			
			cal_r	cal_i	ld	jal	cal_r	cal_i	ld	jal	cal_r	cal_i	ld	jal
			1/rd	1/rt	2/rt	0/r31	0/rd	0/rt	1/rt	0/r31	0/rd	0/rt	0/rt	0/r3
cal_r	rs/rt	1	AO_M->ALU_A/B	AO_M->ALU_A/B	暂停	PC8_M->ALU_A/B	WD_W->ALU_A/B	WD_W->ALU_A/B	WD_W->ALU_A/B	PC8_W->ALU_A/B	内	部	转	发
cal_i	rs	1	AO_M->ALU_A	AO_M->ALU_A	暂停	PC8_M->ALU_A	WD_W->ALU_A	WD_W->ALU_A	WD_W->ALU_A	PC8_W->ALU_A				
beq	rs/rt	0	暂停	暂停	暂停	不可能	AO_M->CMP_A/B	AO_M->CMP_A/B	暂停	PC8_M->CMP_A/B				
ld	rs	1	AO_M->ALU_A	AO_M->ALU_A	暂停	PC8_M->ALU_A	WD_W->ALU_A	WD_W->ALU_A	WD_W->ALU_A	PC8_W->ALU_A				
st	rt	2	WD_W->DMin_M	WD_W->DMin_M	WD_W->DMin_M	PC8_W->DMin_M	WD_W->RT_out_E	WD_W->RT_out_E	WD_W->RT_out_E	PC8_W->RT_out_E				
	rs	1	AO_M->ALU_A	AO_M->ALU_A	暂停	PC8_M->ALU_A	WD_W->ALU_A	WD_W->ALU_A	WD_W->ALU_A	PC8_W->ALU_A				
jr	rs	0	暂停	暂停	暂停	不可能	AO_M->JR	AO_M->JR	暂停	PC8_M->JR				
			转发信号		1	2	3	4	0					
			mf_ALU_A	AO_M		PC8_M	WD_W	PC8_W	原来的数值					
			mf_ALU_B	AO_M		PC8_M	WD_W	PC8_W						
			mf_CMP_A	AO_M		PC8_M								
			mf_CMP_B	AO_M		PC8_M								
			mf_DMin_M	WD_W		PC8_W								
			mf_Rtout_E	WD_W		PC8_W								
			mf_JR	AO_M		PC8_M								

图 2 转发设计表格

24. 数据通路设计总览

级别	部件	输入	LW	SW	ADDU	SUBU	ORI	LUI	BEQ	J	JAL	JR	JALR		输入来源	
F级部件	PC															
	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级对PC的更新	PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4		
D级流水线寄存器	IR_D		IM	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM	IM		
	PC4_D								ADD4		ADD4		ADD4	ADD4		
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]	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							GRF.RD1					GRF.RD1		
		D2							GRF.RD2					GRF.RD2		
	NPC	PC4							ADD4					ADD4		
		IR_D[i26]							IR_D[16]	IR_D[i26]	IR_D[i26]			IR_D[16]	IR_D[i26]	
E级对PC的更新	PC								NPC	NPC	NPC	GRF.RD1	GRF.RD1	NPC	GRF.RD1	
E级流水线寄存器	IR_E		IR_D	IR_D	IR_D	IR_D	IR_D	IR_D	IR_D		IR_D		IR_D	IR_D		
	PC4_E										PC4_D		PC4_D	PC4_D		
	RS_E		GRF.RD1	GRF.RD1	GRF.RD1	GRF.RD1	GRF.RD1	GRF.RD1						GRF.RD1		
	RT_E			GRF.RD2	GRF.RD2	GRF.RD2								GRF.RD2		
	EXT_E		EXT	EXT			EXT	EXT						EXT		
E级部件	ALU	A	RS_E	RS_E	RS_E	RS_E	RS_E	RS_E						RS_E		
		B	EXT_E	EXT_E	RT_E	RT_E	EXT_E	EXT_E						RT_E	EXT_E	
M级流水线寄存器	IR_M		IR_E	IR_E	IR_E	IR_E	IR_E	IR_E			IR_E		IR_E	IR_E		
	PC4_M										PC4_E		PC4_E	PC4_E		
	AO_M		ALU	ALU	ALU	ALU	ALU	ALU						ALU		
	RT_M			RT_E										RT_E		
M级部件	DM	A	AO_M	AO_M										AO_M		
		WD		RT_E										RT_E		
W级流水线寄存器	IR_W		IR_M		IR_M	IR_M	IR_M	IR_M			IR_M		IR_M	IR_M		
	PC4_W										PC4_M		PC4_M	PC4_M		
	AO_W				AO_M	AO_M	AO_M	AO_M						AO_M		
	DR_W															
W级部件	GRF	WA	IR_W[rt]		IR_W[rd]	IR_W[rd]	IR_W[rt]	IR_W[rt]			0x1f		IR_W[rd]	IR_W[rt]	IR_W[rd]	0x1f
		WD	DR_W		AO_W	AO_W	AO_W	AO_W			PC4_W		PC4_W	DR_W	AO_W	PC4_W

图 3 数据通路总览

二、思考题

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

因为乘除法所用周期时间过长，若整合在 ALU 中，且没有独立的 HI，LO 寄存器，那么在运行乘除法时，其他指令无法进行 ALU 的运算或寄存器堆的读写，这样就会使得效率变得很低。

2. 参照你对延迟槽的理解，试解释“乘除槽”。

我对乘除槽的理解是，在执行完乘除法运算后，且在结果未运算出之前，先

执行一些其他的无关指令，这样可以使得流水线最大可能的利用，提高了效率。

3. 为何上文文末提到的 lb 等指令使用的数据扩展模块应在 MEM/WB 之后，而不能在 DM 之后?

因为 DM 的读写所耗费的时间最长，而整个流水线的时钟周期是以最长的时钟周期为划分，所以在 MEM 阶段的执行时间也就是流水线的时钟周期。如果 lb 的数据拓展在该阶段执行，便会使得流水线的整个时钟周期变长。

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

按字节访问更加灵活，例如 c 语言中字符串一个长度为一的字符串就占一个字节，这样来进行读写的话就会更加方便。如果按字来读写，很有可能还需要对读出的字进行截取，增加了不便。

5. 如何概括你所设计的 CPU 的设计风格？为了对抗复杂性你采取了哪些抽象和规范手段？

我属于第二种，也就是 planner 型。为了对抗复杂性，我将每一类指令归类，且在每一条以及每一类指令后加上后缀_D,_E,_M,_W,这样就变得很清晰了。

6. 你对流水线 CPU 设计风格有何见解？

我的这种 CPU 设计风格简单易读，且不容易出错，能够完全覆盖所有的情况，并且由于归类的原因，添加指令时也只需要译码即可。不足的地方就是代码行数过长，但是在实际过程中，直接复制粘贴就可以，添加起来也很容易。总的来说这种风格我认为十分优秀。

三、测试程序

1.cal_r 型（以 addu 为例）

编号	测试类型	前序指令	冲突位置	冲突寄存器	测试序列	测试结果(10 为一周期)
1	R-M-RS	addu	M	RS	addu $3,$1,$2 addu $4,$3,$0	45@00003000: $ 1 <= 00000005 55@00003004: $ 2 <= 00000002 65@00003008: $ 3 <= 00000007 75@0000300c: $ 4 <= 00000007
2	R-M-RT	subu	M	RT	subu $3,$1,$2 addu $4,$0,$3	45@00003000: $ 1 <= 00000005 55@00003004: $ 2 <= 00000002 65@00003008: $ 3 <= 00000003 75@0000300c: $ 4 <= 00000003

3	R-W-RS	addu	W	RS	addu $3,$1,$2
					nop
					addu $4,$3,$0
						45@00003000: $ 1 <= 00000005
						55@00003004: $ 2 <= 00000002
						65@00003008: $ 3 <= 00000007
						85@00003010: $ 4 <= 00000007

4	R-W-RT	subu	W	RT	subu $3,$1,$2
					nop
					addu $4,$0,$3
						45@00003000: $ 1 <= 00000005
						55@00003004: $ 2 <= 00000002
						65@00003008: $ 3 <= 00000003
						85@00003010: $ 4 <= 00000003

5	R-W-RS	addu	W	RS	addu $3,$1,$2
					nop
					nop
					addu $4,$3,$0
						45@00003000: $ 1 <= 00000005
						55@00003004: $ 2 <= 00000002
						65@00003008: $ 3 <= 00000007
						95@00003014: $ 4 <= 00000007

6	R-W-RT	subu	W	RT	subu $3,$1,$2
					nop
					nop
					addu $4,$0,$3
						45@00003000: $ 1 <= 00000005
						55@00003004: $ 2 <= 00000002
						65@00003008: $ 3 <= 00000003
						95@00003014: $ 4 <= 00000003

7	I-M-RS	ori	M	RS	ori $3,$0,4 addu $4,$3,$0	45@00003000: $ 3 <= 00000004 55@00003004: $ 4 <= 00000004
8	I-M-RT	lui	M	RT	lui $3,4 addu $4,$0,$3	65@00003008: $ 3 <= 00040000 75@0000300c: $ 4 <= 00040000
9	I-W-RS	ori	W	RS	ori $3,$0,4 nop addu $4,$3,$0	85@00003010: $ 3 <= 00000004 105@00003018: $ 4 <= 00000004
10	I-W-RT	lui	W	RT	lui $3,4 nop addu $4,$0,$3	115@0000301c: $ 3 <= 00040000 135@00003024: $ 4 <= 00040000
11	I-W-RS	ori	W	RS	ori $3,$0,4 nop nop addu $4,$3,$0	145@00003028: $ 3 <= 00000004 175@00003034: $ 4 <= 00000004
12	I-W-RT	lui	W	RT	lui $3,4 nop nop addu $4,$0,$3	185@00003038: $ 3 <= 00040000 215@00003044: $ 4 <= 00040000

13	LD-W-RS	lw	W	RS	lw $3,0($0)
					nop
					addu $4,$3,$0
						65@00003008: $ 3 <= 00000006
						85@00003010: $ 4 <= 00000006

14	LD-W-RT	lw	W	RT	lw $3,0($0)
					nop
					addu $4,$0,$3
						95@00003014: $ 3 <= 00000006
						115@0000301c: $ 4 <= 00000006

15	LD-W-RS	lw	W	RS	lw $3,0($0)
					nop
					nop
					addu $4,$3,$0
						125@00003020: $ 3 <= 00000006
						155@0000302c: $ 4 <= 00000006

16	LD-W-RT	lw	W	RT	lw $3,0($0) nop	165@00003030: $ 3 <= 00000006 195@0000303c: $ 4 <= 00000006

					nop addu $4,$0,$3	

表中未列出暂停测试，本地已测试正确；前驱指令为 jal 在 31 号寄存器产生的冲突未列举，本地已测试正确。

2. cal_i 型（以 ori 为例）

编号	测试类型	前序指令	冲突位置	冲突寄存器	测试序列	测试结果(10 为一周期)
1	R-M-RS	addu	M	RS	addu $3,$1,$2 ori $4,$3,16	75@0000300c: $ 3 <= 00000005 85@00003010: $ 4 <= 00000015
2	R-W-RS	subu	W	RS	subu $3,$1,$2 nop ori $4,$3,16	95@00003014: $ 3 <= 00000001 115@0000301c: $ 4 <= 00000011
3	R-W-RS	addu	W	RS	addu $3,$1,$2 nop nop ori $4,$3,16	125@00003020: $ 3 <= 00000005 155@0000302c: $ 4 <= 00000015
4	I-M-RS	ori	M	RS	ori $3,$0,4 ori $4,$3,16	165@00003030: $ 3 <= 00000004 175@00003034: $ 4 <= 00000014
5	I-W-RS	lui	W	RS	lui $3,4 nop ori $4,$3,16	185@00003038: $ 3 <= 00040000 205@00003040: $ 4 <= 00040010
6	I-W-RS	lui	W	RS	lui $3,4 nop nop ori $4,$3,16	215@00003044: $ 3 <= 00040000 245@00003050: $ 4 <= 00040010
7	LD-W	lw	W	RS	lw $3,0($0) nop ori $4,$3,16	255@00003054: $ 3 <= 00000002 275@0000305c: $ 4 <= 00000012
8	LD-W	lw	W	RS	lw $3,0($0) nop nop ori $4,$3,16	285@00003060: $ 3 <= 00000002 315@0000306c: $ 4 <= 00000012

表中未列出暂停测试，本地已测试正确；前驱指令为 jal 在 31 号寄存器产生的冲突未列举，本地已测试正确。

3. ld 型（以 lw 指令为例）

编号	测试类型	前序指令	冲突位置	冲突寄存器	测试序列	测试结果(10 为一周期)
1	R-M-RS	addu	M	RS	addu $3,$1,$0 lw $4,0($3)	85@00003010: $ 3 <= 00000004 95@00003014: $ 4 <= 00000010
2	I-M-RS	ori	M	RS	ori $3,$0,4 lw $4,0($3)	105@00003018: $ 3 <= 00000004 115@0000301c: $ 4 <= 00000010
3	LD-M-RS	lw	M	RS	lw $3,0($0) lw $4,0($3)	125@00003020: $ 3 <= 00000004 145@00003024: $ 4 <= 00000010

4	R-W-RS	addu	W	RS	addu $3,$1,$0
					nop
					lw $4,0($3)
						155@00003028: $ 3 <= 00000004
						175@00003030: $ 4 <= 00000010

5	I-W-RS	ori	W	RS	ori $3,$0,4 nop lw $4,0($3)	185@00003034: $ 3 <= 00000004 205@0000303c: $ 4 <= 00000010

6	LD-W-RS	lw	W	RS	lw $3,0($0)
					nop
					lw $4,0($3)
						215@00003040: $ 3 <= 00000004
						235@00003048: $ 4 <= 00000010

7	R-W-RS	addu	W	RS	addu $3,$1,$0
					nop
					nop
					lw $4,0($3)
						245@0000304c: $ 3 <= 00000004
						275@00003058: $ 4 <= 00000010

8	I-W-RS	ori	W	RS	ori $3,$0,4 nop nop lw $4,0($3)	285@0000305c: $ 3 <= 00000004 315@00003068: $ 4 <= 00000010

9	LD-W-RS	lw	W	RS	lw $3,0($0)
					nop
					nop
					lw $4,0($3)
						325@0000306c: $ 3 <= 00000004
						355@00003078: $ 4 <= 00000010

表中未给出 jal 为前驱指令的冲突，本地测试已正确。

4. st 型（以 sw 为例，同时由于 sw 和 lw 在 rs 寄存器产生冲突的解决方案完全一致，上面已经测试过了，在此不再重复测试，下面仅针对 rt 寄存器产生的冲突进行测试）

编号	测试类型	前序指令	冲突位置	冲突寄存器	测试序列	测试结果(10 为一周期)
1	R-W-RT	addu	W	RT	addu $3,$0,$2 sw $3,0($0)	85@00003014: *00000000 <= 00000005 105@00003018: $ 3 <= 00000004

2	I-W-RT	ori	W	RT	ori $3,$0,4 sw $3,0($0)	105@00003018: $ 3 <= 00000004 105@0000301c: *00000000 <= 00000004
3	LD-W-RT	lw	W	RT	lw $3,4($0) sw $3,0($0)	125@00003020: $ 3 <= 00000009 125@00003024: *00000000 <= 00000009
4	JAL-W-31	jal	W	31	jal label sw $31,0($0)	145@00003028: $31 <= 00003030 145@0000302c: *00000000 <= 00003030

5	R-W-RT	addu	W	RT	addu $3,$0,$2
					nop
					sw $3,0($0)
						85@00003010: $ 3 <= 00000005
						95@00003018: *00000000 <= 00000005

6	I-W-RT	ori	W	RT	ori $3,$0,4 nop sw $3,0($0)	115@0000301c: $ 3 <= 00000004 125@00003024: *00000000 <= 00000004

7	LD-W-RT	lw	W	RT	lw $3,4($0)
					nop
					sw $3,0($0)
						145@00003028: $ 3 <= 00000009
						155@00003030: *00000000 <= 00000009

8	JAL-W-RT	jal	W	31	jal label nop label: sw $31,0($0)	175@00003034: $31 <= 0000303c 185@0000303c: *00000000 <= 0000303c

9	R-W-RT	addu	W	RT	addu $3,$0,$2
					nop
					nop
					sw $3,0($0)
						85@00003010: $ 3 <= 00000005
						105@0000301c: *00000000 <= 00000005

10	I-W-RT	ori	W	RT	ori $3,$0,4 nop nop sw $3,0($0)	125@00003020: $ 3 <= 00000004 145@0000302c: *00000000 <= 00000004

11	LD-W-RT	lw	W	RT	lw $3,4($0)
					nop
					nop
					sw $3,0($0)
						165@00003030: $ 3 <= 00000009
						185@0000303c: *00000000 <= 00000009

12	JAL-W-RT	jal	W	31	jal label nop label: nop sw $31,0($0)	205@00003040: $31 <= 00003048 225@0000304c: *00000000 <= 00003048

5, 乘除类指令（乘除法指令的冲突与 r 型指令类似，故不重点对冲突测试，仅对模拟延迟进行测试）

下给出测试程序与运行结果：

```
.text
lui $2,0x8004
lui $3,0x8009
mult $2,$3
mflo $4
mfhi $5
div $2,$3
multu $2,$3
mflo $4
mfhi $5
div $2,$3
mflo $4
mfhi $5
mult $2,$3
divu $2,$3
mflo $4
mfhi $5
```

```
 9@00003000: $ 2 <= 80040000
11@00003004: $ 3 <= 80090000
25@0000300c: $ 4 <= 00000000
27@00003010: $ 5 <= 3ff98024
63@0000301c: $ 4 <= 00000000
65@00003020: $ 5 <= 40068024
89@00003028: $ 4 <= 00000001
91@0000302c: $ 5 <= fffb0000
127@00003038: $ 4 <= 00000000
129@0000303c: $ 5 <= 80040000
```

通过对周期的比对，发现符合预期，同时结果也与 Mars 行为保持一致，故正确。