计算机组成原理实验报告

16061007 王文珺

一、 CPU 设计文档

（一） 数据通路设计

数据通路的设计主要为构建表格，在表头写出每一个功能模块，以及它们的数据输入，需要注意的是我们把读功能和写功能分别写在 D 级和 W 级。在构造数据通路时，我们暂且忽略流水线带来的各种冒险，纯粹考虑数据的流动，并且为可能有多个数据来源的输入端口构造多选器，下图为考虑过冒险而构造的数据通路（黄色为转发）

级别	部件	输入	lw	sw	addu	subu	ori	beq	j	jal	jr	MUX	0	1	2
IF级部件	PC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4/NPC	NPC	NPC	RF.RD1	MUX_PC	ADD4	NPC	RF.RD1(MF_PC_R)
	ADD4		PC	PC	PC	PC	PC	PC	PC	PC	PC				
	IM		PC	PC	PC	PC	PC	PC	PC	PC					
F/D级流水线寄存器	IR		IM	IM	IM	IM	IM	IM	IM	IM	IM				
	NPC		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4					
D级部件	RF	A1	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D			IR[rs]@D				
		A2		IR[rt]@D	IR[rt]@D	IR[rt]@D		IR[rt]@D							
	EXT		IR[116]@I	IR[116]@D			IR[116]@D								
	CMP	D1						RF.RD1				MF_CMP1_D			
		D2						RF.RD2				MF_CMP2_D			
	NPC	PC4						NPC@D	NPC@D	NPC@D					
		I26						IR[116]@I	IR[126]@I	IR[126]@D					
D/E级流水线寄存器	V1		RF.RD1	RF.RD1	RF.RD1	RF.RD1	RF.RD1					MF_CMP1_D			
	V2			RF.RD2	RF.RD2	RF.RD2						MF_CMP2_D			
	A1		IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D	IR[rs]@D								
	A2			IR[rt]@D	IR[rt]@D	IR[rt]@D									
	A3		IR[rt]@D		IR[rd]@D	IR[rd]@D	IR[rt]@D			31		MUX_A3_E	IR[rt]@D	IR[rd]@D	31
	E32		EXT	EXT			EXT								
	PC4									NPC@D					
E级部件	ALU	A	V1@E	V1@E	V1@E	V1@E	V1@E					MF_ALU_A_E			
		B	E32@E	E32@E	V2@E	V2@E	E32@E					MUX_ALU_B	V2@E(MF_ALU_B_E)	E32@E	
E/M级流水线寄存器	V2			V2@E								MF_V2_M			
	A2			A2@E											
	A0		ALU	ALU	ALU	ALU	ALU								
	A3		A3@E		A3@E	A3@E	A3@E			A3@E					
	PC4									PC4@E					
M级部件	DM	A	A0@M	A0@M	A0@M	A0@M	A0@M								
		WD		V2@M								MF_WD_M			
M/W级流水线寄存器	A3		A3@M		A3@M	A3@M	A3@M								
	PC4									PC4@M					
	A0				A0@M	A0@M	A0@M								
	DR		DM												
W级部件	RF	A3	A3@W		A3@W	A3@W	A3@W			A3@W					
		WD	DR@W		A0@W	A0@W	A0@W			PC4@W		MUX_RF_WD	DR@W	A0@W	PC4@W

接下来我们考虑流水线的暂停和转发机制，流水线产生冒险的原因，就是后面指令需要的数据恰好被前面的指令所供给，当后面的指令需要使用数据时（例如进入 ALU 运算），前面的指令还没有将该数据存入寄存器堆，从而导致后面的指令无法正常读到正确数据。对于这个问题我们可以通过暂停和转发来解决，当后面的数据已经算出来，但是没有存入寄存器堆时，我们可以通过转发来从后面的流水级将计算结果发给前面的需求者使用，如果后面的数据还没有算出来，我们便只能暂停流水线，等待需要的数据计算完毕，再开始下面的工作。至此，我们需要引入供求时间-需求时间模型来判定转发和暂停。

在此定义 Tuse 为：当这条指令位于 D 级的时候，再经过多少个时钟周期就必须要使用相应的数据，例如对于 beq 指令，立即就要进行判断是否相等，所以 Tuse=0，对于 addu 指令，当它进入 E 级 ALU 才需要使用数据，所以 Tuse=1。

在此定义 Tnew 为，位于某个流水级的某个指令，经过多少个时钟周期可以算出结果并

且存储到流水级寄存器里。例如对于 addu 指令，当其处于 E 级，结果还没有存入寄存器中，此时的 Tnew=1，而当它处于后面的级别时，结果已经写入流水级寄存器，所以 Tnew=0。

根据这两个定义，我们可以对现有的指令进行整理得到下面的表格。

指令	Tuse		功能部件	Tnew		
	rs	rt		E	M	W
addu	1	1	ALU	1	0	0
subu	1	1	ALU	1	0	0
ori	1		ALU	1	0	0
lw	1		DM	2	1	0
sw	1	2				
beq	0	0				
jal			PC	0	0	0
jr	0					
j						
	{0, 1}	{0, 1, 2}				

当两条指令发生数据冲突时，我们通过判断 Tuse 和 Tnew 的值就可以判断进行什么样的操作。①Tnew=0 时，结果已经算出，可以通过转发或内部转发解决。②Tnew<=Tuse，说明需要的数据可以及时苏拿出，可以通过转发结果来解决。③Tnew>Tuse，说明需要的数据不能及时算出，所以需要暂停流水线等待数据计算。由此我们可以构造 rs 和 rt 的策略矩阵。

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

Tuse_rs0=beq+jr

Tuse_rs1=addu+subu+ori+lw+sw

rt	E			M			W		
	ALU	DM	PC	ALU	DM	PC	ALU	DM	PC
	1	2	0	0	1	0	0	0	0
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

Tuse_rt0=beq

Tuse_rt1=addu+subu

Tuse_rt2=sw

具体的实现转发和暂停控制信号我们在控制器设计模块进行详细阐述，接下来给出数据通路中所有部件的端口说明。

IF：

序号	端口名称	I\0	功能名称	功能描述
1	Clk	I	时钟	提供时钟信号
2	Reset	I	复位	当复位信号有效时，PC 被设置为 0x00003000
3	Branch	I	跳转	当 Branch 信号有效时，将跳转地址赋给 PC

4	Stall	I	暂停	当 Stall 信号有效时，冻结 PC
5	nextPC	I	跳转地址	提供跳转地址
6	Instr	O	指令	输出正在执行的指令
7	ADD4	O	下条地址	非跳转情况下的下条指令地址

D_PIPE:

序号	端口名称	I\0	功能名称	功能描述
1	Clk	I	时钟	提供时钟信号
2	Reset	I	复位	当复位信号有效时，复位清零 D 级流水寄存器
3	Instr	I	指令	将上级指令存入 D 级流水寄存器
4	Stall	I	暂停	当 Stall 信号有效时，冻结 D 级流水寄存器
5	ADD4	I	下条地址	将上级指令的下条地址存入 D 级流水寄存器
6	Instr_D	O	D 级指令	输出 D 级流水寄存器的指令
7	PC4_D	O	D 级下条地址	输出 D 级流水寄存器的指令的下条地址

NPC:

序号	端口名称	I\0	功能名称	功能描述
1	Imm16	I	16 位立即数	提供 16 位立即数
2	Imm26	I	26 位立即数	提供 26 位立即数
3	PC4	I	当前 PC+4	提供当前的 PC+4
4	jrpc	I	跳转地址	提供从寄存器读出的地址
5	NPC_Sel	I	选择信号	00/beq 跳转，01/j，jal 跳转，10/jr 跳转
6	CMPResult	I	比较结果	0/不进行 beq 跳转，1/进行 beq 跳转
7	nPC	O	下条地址	输出下条地址

GRF:

序号	端口名称	I\0	功能名称	功能描述
1	Rreg1	I	寄存器 1 选择	通过多路选择器选择寄存器 1
2	Rreg2	I	寄存器 2 选择	通过多路选择器选择寄存器 2
3	Wreg	I	选择写入寄存器	当控制信号 RegWrite 有效时， 在时钟上升沿将 Wdata 中的数据写入 Wreg 通过 Decoder 选择的寄存器中，
4	RegWrite	I	写入控制	
5	Wdata	I	写入的数据	
6	Rdata1	O	数据 1 输出	输出寄存器 1 的数据
7	Rdata2	O	数据 2 输出	输出寄存器 2 的数据
8	clk	I	时钟	时钟信号
9	reset	I	复位	当 reset 信号有效时，将所有寄存器值清零
10	PrePC	I	当前地址	用于输出当前指令地址

CMP:

序号	端口名称	I\0	功能名称	功能描述
1	CMPD1	I	比较数 1	提供第一个比较数
2	CMPD2	I	比较数 2	提供第二个比较数
3	CMPResult	O	比较结果	比较结果：0/不相等 1/相等

EXT：

序号	端口名称	I\0	功能名称	功能描述
1	Imm16	I	16 位立即数	提供被扩展的立即数
2	ExtOp	I	扩展信号	0/无符号扩展 1/符号扩展
3	Imm32	O	32 位立即数	输出扩展后的立即数

E_PIPE：

序号	端口名称	I\0	功能名称	功能描述
1	Clk	I	时钟	提供时钟信号
2	Reset	I	复位	当复位信号有效时，复位清零 E 级流水寄存器
3	Instr_D	I	D 级指令	将 D 级指令存入 E 级流水寄存器
4	Stall	I	暂停	当 stall 信号有效，复位清零 E 级流水寄存器
5	PC4_D	I	D 级下条地址	将 D 级下条地址存入 E 级流水寄存器
6	Resin	I	写入 res_e	将 res_e 的值存入 E 级流水寄存器
7	RF_RD_1	I	数据 1	将数据 1 存入 E 级流水寄存器
8	RF_RD_2	I	数据 2	将数据 2 存入 E 级流水寄存器
9	E32	I	32 位立即数	将 32 位立即数存入 E 级流水寄存器
10	MUX_A3	I	写入寄存器	将写入寄存器存入 E 级流水寄存器
11	Res_E	O	E 级 res_e	输出当前 E 级 res_e
12	Instr_E	O	E 级指令	输出当前 E 级指令
13	PC4_E	O	E 级下条地址	输出当前 E 级下条地址
14	V1_E	O	E 级数据 1	输出当前 E 级数据 1
15	V2_E	O	E 级数据 2	输出当前 E 级数据 2
16	A1_E	O	E 级寄存器 1	输出当前 E 级寄存器 1
17	A2_E	O	E 级寄存器 2	输出当前 E 级寄存器 2
18	A3_E	O	E 级写入寄存器	输出当前 E 级写入寄存器
19	E32_E	O	E 级 32 位立即数	输出当前 E 级 32 位立即数

ALU:

序号	端口名称	I\0	功能名称	功能描述
1	A	I	数据 1	提供运算的数据 1
2	B	I	数据 2	提供运算的数据 2
3	ALUOp	I	计算信号	0000/加法，0001/减法，0010/或，0011/加载至高 16 位
4	C	O	计算结果	输出计算结果

M_PIPE：

序号	端口名称	I\0	功能名称	功能描述
1	Clk	I	时钟	提供时钟信号
2	Reset	I	复位	当复位信号有效时，复位清零 M 级流水寄存器
3	Instr_E	I	E 级指令	将 E 级指令存入 M 级流水寄存器
4	PC4_E	I	E 级下条地址	将 E 级下条地址存入 M 级流水寄存器
5	res_E	I	E 级 res_e	将 res_e 的值存入 M 级流水寄存器
6	AO	I	计算结果	将 ALU 输出结果存入 M 级流水寄存器

7	V2_E	I	写入数据	将写入数据存入 M 级流水寄存器
8	A2_E	I	数据对应寄存器	将数据对应寄存器存入 M 级流水寄存器
9	A3_E	I	写入寄存器	将写入寄存器存入 M 级流水寄存器
10	res_M	O	M 级 res_m	输出当前 M 级 res_m
11	Instr_M	O	M 级指令	输出当前 M 级指令
12	PC4_M	O	M 级下条地址	输出当前 M 级下条地址
13	V2_M	O	M 级写入数据	输出当前 M 级写入数据
14	A2_M	O	M 级数据寄存器	输出当前 M 级数据对应寄存器
15	A3_M	O	M 级写入寄存器	输出当前 M 级写入寄存器
16	AO_M	O	M 级计算结果	输出当前 M 级计算结果

DM：

序号	端口名称	I\0	功能名称	功能描述
1	reset	I	复位	当复位信号有效时，DM 中的数据被全部清零
2	Address	I	32 位地址	利用 Splitter 选择 DM 中存储的某条数据
3	WriteData	I	写入的数据	当 MemWrite 信号有效时，将 WriteData 写入 Address 所选择的地址在 DM 中的位置
4	MemWrite	I	数据写入	
5	ReadData	O	读取的数据	输出 DM 被选择的位置所保存的数据
6	clk	I	时钟	时钟信号
7	PC	I	当前地址	用于输出当前指令地址

W_PIPE:

序号	端口名称	I\0	功能名称	功能描述
1	Clk	I	时钟	提供时钟信号
2	Reset	I	复位	当复位信号有效时，复位清零 W 级流水寄存器
3	Instr_M	I	M 级指令	将 E 级指令存入 W 级流水寄存器
4	PC4_M	I	M 级下条地址	将 E 级下条地址存入 W 级流水寄存器
5	res_M	I	M 级 res_m	将 res_m 的值存入 W 级流水寄存器
6	AO_M	I	计算结果	将 ALU 输出结果存入 W 级流水寄存器
7	DM	I	读出数据	将 DM 读出数据存入 W 级流水寄存器
8	A3_M	I	写入寄存器	将写入寄存器存入 W 级流水寄存器
9	res_W	O	W 级 res_w	输出当前 W 级 res_w
10	Instr_W	O	W 级指令	输出当前 W 级指令
11	PC4_W	O	W 级下条地址	输出当前 W 级下条地址
12	A3_W	O	W 级写入寄存器	输出当前 W 级写入寄存器
13	AO_W	O	W 级计算结果	输出当前 W 级计算结果
14	DM_W	O	W 级读出数据	输出当前 W 级读出数据

（二） 控制器设计

控制器设计方面，主要由两种控制器组成，一种生成是正常的控制信号，一种生成转发和暂停信号。正常的控制信号以按照下表，通过和之前 project 一样的方法生成。

	addu	subu	ori	lui	sw	lw	beq	jal	j	jr	nop	
RegWrite	1	1	1	1	0	1	0	1	0	0	0	0/不写 1/写
A3_E_Sel	01	01	00	00	00	00	0	10	00	00	00	00/rt 01/rd 10/31
MemWrite	0	0	0	0	1	0	0	0	0	0	0	0/不写DM 1/写DM
Branch	0	0	0	0	0	0	1	1	1	1	0	0/不跳转 1/跳转
ALU_B_Sel	0	0	1	1	1	1	0	0	0	0	0	0/寄存器数据 1/扩展后的立即数
RF_WD_Sel	01	01	01	01	00	00	00	10	00	00	00	00/DM 01/AO 10/PC4
NPC_Sel	00	00	00	00	00	00	00	01	01	10	00	00/beq 01/j,jal 10/jr
ExtOp	0	0	0	0	1	1	0	0	0	0	0	0/无符号扩展 1/有符号扩展
ALUOp	00	01	10	11	00	00	00	00	00	00	00	00/加 01/减 10/或 11/左移16位

冲突控制器方面，采用高小鹏老师讲义中的方法，设置 E 级 Res 寄存器的初值，并进行简单传递

通过策略矩阵，分别构造 rs 和 rt 的暂停条件：

```
wire [4:0] A1=Instr_D[25:21];

assign Stall_rs0_E1=Tuse_rs0 & (res_e==`ALU) & (A1==A3_E) & (A1!=5'b00000);
assign Stall_rs0_E2=Tuse_rs0 & (res_e==`DM) & (A1==A3_E) & (A1!=5'b00000);
assign Stall_rs1_E2=Tuse_rs1 & (res_e==`DM) & (A1==A3_E) & (A1!=5'b00000);
assign Stall_rs0_M1=Tuse_rs0 &(res_m==`DM) & (A1==A3_M) & (A1!=5'b00000);
assign Stall_rs= Stall_rs0_E1 | Stall_rs0_E2 | Stall_rs1_E2 | Stall_rs0_M1;

assign Stall_rt0_E1=Tuse_rt0 & (res_e==`ALU) & (A1==A3_E) & (A1!=5'b00000);
assign Stall_rt0_E2=Tuse_rt0 & (res_e==`DM) & (A1==A3_E) & (A1!=5'b00000);
assign Stall_rt1_E2=Tuse_rt1 & (res_e==`DM) & (A1==A3_E) & (A1!=5'b00000);
assign Stall_rt0_M1=Tuse_rt0 &(res_m==`DM) & (A1==A3_M) & (A1!=5'b00000);
assign Stall_rt= Stall_rt0_E1 | Stall_rt0_E2 | Stall_rt1_E2 | Stall_rt0_M1;

assign stall=Stall_rs||Stall_rt;
```

转发机制通过判断不同位置转发的条件，并且对不同来源的数据设立优先级，得到控制信号的表达式。本文的 CPU 设计采用了五个转发，分别位于 CMP 的两个输入，ALU 的两个输入和 DM 的输入。

```
assign F_CMP1_D=((A1_D==A3_E)&(res_e==`PC)&(A1_D!=0))?`E2D_PC:
                ((A1_D==A3_M)&(res_m==`PC)&(A1_D!=0))?`M2D_PC:
                ((A1_D==A3_M)&(res_m==`ALU)&(A1_D!=0))?`M2D_ALU:
                ((A1_D==A3_W)&(res_w==`ALU)&(A1_D!=0))?`W2D_ALU:
                ((A1_D==A3_W)&(res_w==`DM)&(A1_D!=0))?`W2D_DM:
                ((A1_D==A3_W)&(res_w==`PC)&(A1_D!=0))?`W2D_PC:0;

assign F_CMP2_D=((A2_D==A3_E)&(res_e==`PC)&(A2_D!=0))?`E2D_PC:
                ((A2_D==A3_M)&(res_m==`PC)&(A2_D!=0))?`M2D_PC:
                ((A2_D==A3_M)&(res_m==`ALU)&(A2_D!=0))?`M2D_ALU:
                ((A2_D==A3_W)&(res_w==`ALU)&(A2_D!=0))?`W2D_ALU:
                ((A2_D==A3_W)&(res_w==`DM)&(A2_D!=0))?`W2D_DM:
                ((A2_D==A3_W)&(res_w==`PC)&(A2_D!=0))?`W2D_PC:0;
```

```
assign F_ALU_A_E=((A1_E==A3_M)&(res_m==`PC)&(A1_E!=0))?`M2E_PC:
                 ((A1_E==A3_M)&(res_m==`ALU)&(A1_E!=0))?`M2E_ALU:
                 ((A1_E==A3_W)&(res_w==`ALU)&(A1_E!=0))?`W2E_ALU:
                 ((A1_E==A3_W)&(res_w==`DM)&(A1_E!=0))?`W2E_DM:
                 ((A1_E==A3_W)&(res_w==`PC)&(A1_E!=0))?`W2E_PC: 0;

assign F_ALU_B_E=((A2_E==A3_M)&(res_m==`PC)&(A2_E!=0))?`M2E_PC:
                 ((A2_E==A3_M)&(res_m==`ALU)&(A2_E!=0))?`M2E_ALU:
                 ((A2_E==A3_W)&(res_w==`ALU)&(A2_E!=0))?`W2E_ALU:
                 ((A2_E==A3_W)&(res_w==`DM)&(A2_E!=0))?`W2E_DM:
                 ((A2_E==A3_W)&(res_w==`PC)&(A2_E!=0))?`W2E_PC: 0;


assign F_WD_M=((A2_M==A3_W)&(res_w==`ALU)&(A2_M!=0))?`W2M_ALU:
              ((A2_M==A3_W)&(res_w==`DM)&(A2_M!=0))?`W2M_DM:
              ((A2_M==A3_W)&(res_w==`PC)&(A2_M!=0))?`W2M_PC:0;
```

由于不同的控制信号是在不同的级别使用的，所以这里使用了分布式译码，仅在需要此控制信号的级别传入该信号。即实例化四个 control 单元。

（三） 测试程序

```
.text
ori $t0,$0,123#测试 ori
ori $t1,$t0,456
lui $t3,123#测试 lui
lui $t4,0xffff
ori $t4,$t4,0xffff #t4=-1
jal ra#测试 jal 及延迟槽
ori $t1,$0,678#执行
addu $s0,$t0,$t3 #测试 addu ++
addu $s1,$t0,$t4 #+-
addu $s2,$t4,$t4 #--
subu $s3,$t0,$t3 #测试 subu++
subu $s4,$t0,$t4 #+-
subu $s5,$t4,$t4 #--
j loop_before#测试 j 及延迟槽
ori $t1,$0,123#执行
loop_before:
ori $a0,$0,0
ori $a1,$0,1
ori $a2,$0,2
ori $s3,$0,0x000c
sw $t0,-8($s3)#测试 sw
sw $t2,0($s3)
sw $t4,8($s3)
lw $a0,-8($s3)#测试 lw
lw $a1,0($s3)
```

```
lw $a2,8($s3)
j end
nop
ra:
jr $ra #测试 jr 及延迟槽
ori $t1,$0,234 #执行
end:
ori $1,$0,1
ori $2,$0,4
ori $3,$0,1
beq $1,$2,jump #测试 beq 不跳转
ori $t1,$0,111 #执行
beq $1,$3,jump #测试 beq 跳转
ori $t1,$0,222 #执行
ori $t1,$0,111 #不执行
jump:
#R-E-RS 冲突
ori $1,$0,100
ori $2,$0,100
ori $3,$0,100
subu $1,$2,$3
beq $1,$2,wrong #若转发则不跳转
nop

#R-E-RT 冲突
ori $1,$0,100
subu $1,$2,$3
beq $2,$1,wrong #若转发则不跳转
nop

#R-M-RS 冲突
ori $1,$0,100
ori $2,$0,100
ori $3,$0,100
subu $1,$2,$3
addu $4,$1,$3

ori $1,$0,100
subu $1,$2,$3
ori $4,$1,123

ori $1,$0,100
subu $1,$2,$3
lw $5,0($1)
```

```
ori $1,$0,357
addu $1,$0,$0
sw $4,0($1)

ori $1,$0,100
subu $1,$2,$3
nop
beq $1,$2,wrong #若转发则不跳转
nop

#R-M-RT 冲突
ori $1,$0,100
subu $1,$2,$3
addu $4,$2,$1

ori $6,$0,123
subu $6,$2,$3
sw $6,0($0)

ori $1,$0,100
subu $1,$2,$3
nop
beq $2,$1,wrong #若转发则不跳转
nop

#R-W-RS 冲突
ori $1,$0,100
subu $1,$2,$3
nop
addu $4,$1,$2

ori $1,$0,100
subu $1,$2,$3
nop
ori $4,$1,123

ori $1,$0,100
subu $1,$0,$0
nop
lw $5,0($1)

ori $1,$0,357
addu $1,$0,$0
```

```
nop
sw $4,0($1)

ori $1,$0,100
subu $1,$2,$3
nop
nop
beq $1,$2,wrong #若转发则不跳转
nop

#R-W-RT 冲突
ori $1,$0,100
subu $1,$2,$3
nop
addu $4,$2,$1

ori $6,$0,123
subu $6,$2,$3
sw $6,0($0)

ori $1,$0,100
subu $1,$2,$3
nop
nop
beq $2,$1,wrong #若转发则不跳转
nop

#R-W-RS 冲突
ori $1,$0,100
subu $1,$2,$3
sw $1,0($2)

#I-E-RS 冲突
ori $1,$0,100
ori $2,$0,100
ori $3,$0,100
ori $1,$0,200
beq $1,$2,wrong
nop

#I-E-RT 冲突
ori $1,$0,100
ori $1,$0,200
beq $2,$1,wrong
```

```
nop

#I-M-RS 冲突
ori $1,$0,100
ori $1,$2,1000
addu $4,$1,$2

ori $1,$0,8
lw $5,0($1)

ori $1,$0,12
sw $4,0($1)

ori $1,$0,100
ori $2,$0,100
ori $3,$0,100
ori $1,$0,200
nop
beq $1,$2,wrong
nop

#I-M-RT 冲突
ori $1,$0,100
ori $1,$2,1000
addu $4,$2,$1

ori $1,$0,8
sw $1,0($0)

ori $1,$0,100
ori $2,$0,100
ori $3,$0,100
ori $1,$0,200
nop
beq $2,$1,wrong
nop

#I-W-RS 冲突
ori $1,$0,100
ori $1,$2,1000
nop
addu $4,$1,$2

ori $1,$0,8
```

```
nop
lw $5,0($1)

ori $1,$0,12
nop
sw $4,0($1)

ori $1,$0,100
ori $2,$0,100
ori $3,$0,100
ori $1,$0,200
nop
nop
beq $1,$2,wrong
nop

#I-W-RT 冲突
ori $1,$0,100
ori $1,$2,1000
nop
addu $4,$2,$1

ori $1,$0,8
nop
sw $1,0($0)

ori $1,$0,12
sw $1,0($0)

ori $1,$0,100
ori $2,$0,100
ori $3,$0,100
ori $1,$0,200
nop
nop
beq $2,$1,wrong
nop

#LD-E-RS
ori $1,$0,200
ori $2,$0,100
sw $1,0($0)
ori $1,$0,100
lw $1,0($0)
```

```
beq $1,$2,wrong
nop

#LD-E-RT
ori $1,$0,200
sw $1,0($0)
ori $1,$0,100
lw $1,0($0)
beq $2,$1,wrong
nop

#LD-M-RS
ori $1,$0,100
lw $1,0($0)
addu $3,$1,$2

ori $1,$0,100
lw $1,0($0)
nop
beq $1,$2,wrong
nop

#LD-M-RT
ori $1,$0,100
lw $1,0($0)
addu $3,$2,$1

ori $1,$0,100
lw $1,0($0)
nop
beq $2,$1,wrong
nop

#LD-W-RS
ori $1,$0,100
lw $1,0($0)
nop
addu $3,$1,$2

ori $1,$0,100
lw $1,0($0)
nop
sw $3,0($1)
```

```
ori $1,$0,100
lw $1,0($0)
nop
nop
beq $1,$2,wrong
nop

#LD-W-RT
ori $1,$0,100
lw $1,0($0)
nop
addu $3,$2,$1

ori $1,$0,100
lw $1,0($0)
sw $1,0($2)

ori $1,$0,100
lw $1,0($0)
nop
nop
beq $2,$1,wrong
nop

#JAL-M-RS
jal jalmrs
addu $3,$31,$1
jalmrs:
#JAL-M-RT
jal jalmrt
addu $3,$1,$31
jalmrt:
#JAL-W-RS
jal jalwrs
nop
jalwrs:
addu $3,$31,$1
#JAL-W-RT
jal jalwrt
nop
jalwrt:
addu $3,$1,$31
wrong:
```

二、 思考题

序号	测试类型	前序指令	前序指令位置	冲突位置	冲突寄存器	测试样例	解决
1	R-E-RS	subu	E	CMP	Rs	Subu $1,$2,$3 Beq $1,$4,label	暂停 M2D 转发 AO@M
2	R-E-RT	subu	E	CMP	Rt	Subu $1,$2,$3 Beq $4,$1,label	暂停 M2D 转发 AO@M
3	R-M-RS	Subu	M	ALU	Rs	Subu $1,$2,$3 Addu $4,$1,$2	M2E 转发 AO@M
4	R-M-RT	Subu	M	ALU	Rt	Subu $1,$2,$3 Addu $4,$2,$1	M2E 转发 AO@M
5	R-M-RS	Subu	M	CMP	Rs	Subu $1,$2,$3 Instr 无关 Beq $1,$4,label	M2D 转发 AO@M
6	R-M-RT	Subu	M	CMP	Rt	Subu $1,$2,$3 Instr 无关 Beq $4,$1,label	M2D 转发 AO@M

7	R-W-RS	Subu	W	ALU	Rs	Subu $1,$2,$3
						Instr 无关
						Addu $4,$1,$2
							W2E 转发 AO@W

8	R-W-RT	Subu	W	ALU	Rt	Subu $1,$2,$3
						Instr 无关
						Addu $4,$2,$1
							W2E 转发 AO@W

9	R-W-RS	Subu	W	CMP	Rs	Subu $1,$2,$3 Instr 无关 Instr 无关 Beq $1,$4,label	W2D 转发 AO@W
0	R-W-RT	Subu	W	CMP	Rt	Subu $1,$2,$3 Instr 无关 Instr 无关 Beq $4,$1,label	W2D 转发 AO@W

11	R-W-RS	Subu	W	ALU	Rs	Subu $1,$2,$3
						Instr 无关
						Sw $4,0($1)
							W2E 转发 AO@W

12	R-W-RT	subu	W	DM	Rt	Subu $1,$2,$3 Sw $1,0($4)	W2M 转发 AO@W
13	I-E-RS	Ori	E	CMP	Rs	Ori $1,$2,100 Beq $1,$3,label	暂停 M2D 转发 AO@M
14	I-E-RT	Ori	E	CMP	Rt	Ori $1,$2,100 Beq $3,$1,label	暂停 M2D 转发 AO@M
15	I-M-RS	Ori	M	ALU	Rs	Ori $1,$2,$100 Addu $4,$1,$2	M2E 转发 AO@M
16	I-M-RT	Ori	M	ALU	Rt	Ori $1,$2,$100 Addu $4,$2,$1	M2E 转发 AO@M
17	I-M-RS	Ori	M	CMP	Rs	Ori $1,$2,100 Instr 无关 Beq $1,$3,label	M2D 转发 AO@M
18	I-M-RT	Ori	M	CMP	Rt	Ori $1,$2,100 Instr 无关 Beq $3,$1,label	M2D 转发 AO@M
19	I-W-RS	Ori	W	ALU	Rs	Ori $1,$2,$100 Instr 无关 Addu $4,$1,$2	W2E 转发 AO@W
20	I-W-RT	Ori	W	ALU	Rt	Ori $1,$2,$100 Instr 无关 Addu $4,$2,$1	W2E 转发 AO@W
21	I-W-RS	Ori	W	CMP	Rs	Ori $1,$2,100 Instr 无关 Instr 无关 Beq $1,$3,label	W2D 转发 AO@W

22	I-W-RT	Ori	W	CMP	Rt	Ori $1,$2,100 Instr 无关 Instr 无关 Beq $3,$1,label	W2D 转发 AO@W
23	I-W-RS	Ori	W	ALU	Rs	Ori $1,$2,100 Instr 无关 Sw $4,0($1)	W2D 转发 AO@W
24	I-W-RT	Ori	W	DM	Rt	Ori $1,$2,100 Sw $1,0($4)	W2D 转发 AO@W
25	LD-E-RS	Lw	E	CMP	Rs	Lw $1,0($2) Beq $1,$3,label	暂停两个周期 W2D 转发 DR@W
26	LD-E-RT	Lw	E	CMP	Rt	Lw $1,0($2) Beq #3,$1,label	暂停两个周期 W2D 转发 DR@W
27	LD-M-RS	Lw	M	ALU	Rs	Lw $1,0($2) Addu $3,$1,$2	暂停一个周期 W2E 转发 DR@W
28	LD-M-RT	Lw	M	ALU	Rt	Lw $!,0($2) Addu $3,$2,$1	暂停一个周期 W2E 转发 DR@W
29	LD-M-RS	Lw	M	CMP	Rs	Lw $1,0($2) Instr 无关 Beq $1,$3,label	暂停一个周期 W2D 转发 DR@W
30	LD-M-RT	Lw	M	CMP	Rt	Lw $1,0($2) Instr 无关 Beq $3,$1,label	暂停一个周期 W2D 转发 DR@W

31	LD-W-RS	Lw	W	ALU	Rs	Lw $1,0($2)
						Instr 无关
						Addu $3,$1,$2
							W2E 转发 DR@W

32	LD-W-RT	Lw	W	ALU	Rt	Lw $1,0($2)
						Instr 无关
						Addu $3,$2,$1
							W2E 转发 DR@W

33	LD-W-RS	Lw	W	ALU	Rs	Lw $1,0($2)
						Instr 无关
						Sw $3,0($1)
							W2E 转发 DR@W

34	LD-W-RT	Lw	W	DM	Rt	Lw $1,0($2)
						Instr 无关
						Sw $1,0($3)
							W2E 转发 DR@W

35	LD-W-RS	Lw	W	CMP	Rs	Lw $1,0($2) Instr 无关 Instr 无关 Beq $1,$3,label	W2D 转发 DR@W
36	LD-W-RT	Lw	W	CMP	Rt	Lw $1,0($2) Instr 无关 Instr 无关 Beq $3,$1,label	W2D 转发 DR@W
37	JAL-E-RS	Jal	E	CMP	Rs	Jal label Beq $ra,$2,label	D2D 转发 PC8
38	JAL-E-RT	Jal	E	CMP	Rt	Jal label Beq $2,$ra,label	D2D 转发 PC8
39	JAL-M-RS	Jal	M	ALU	Rs	Jal label Addu $2,$ra,$1	M2E 转发 PC8
40	JAL-M-RT	Jal	M	ALU	Rt	Jal label Addu $2,$1,$ra	M2E 转发 PC8
41	JAL-M-RS	Jal	M	CMP	Rs	Jal label Instr 无关 Beq $2,$ra,label	M2D 转发 PC8
42	JAL-M-RT	Jal	M	CMP	Rt	Jal label Instr 无关 Beq $ra,$2,label	M2D 转发 PC8
43	JAL-W-RS	Jal	W	ALU	Rs	Jal label	W2E 转发 PC8

						Instr 无关 Addu $2,$ra,$1	
44	JAL-W-RT	Jal	W	ALU	Rt	Jal label Instr 无关 Addu $2,$1,$ra	W2E 转发 PC8
45	JAL-W-RS	Jal	W	CMP	Rs	Jal label Instr 无关 Instr 无关 Beq $ra,$2,label	W2D 转发 PC8
46	JAL-W-RT	Jal	W	CMP	Rt	Jal label Instr 无关 Instr 无关 Beq $2,$ra,label	W2D 转发 PC8
47	JAL-W-RS	Jal	W	ALU	Rs	Jal label Instr 无关 Sw $2,0($ra)	W2E 转发 PC8
48	JAL-W-RT	Jal	W	DM	Rt	Jal Sw $ra,0($2)	W2E 转发 PC8