P5 Verilog 流水线 CPU 设计文档

1. 模块与层次结构

本次 CPU 设计以 Verilog 单周期 CPU（32 位）设计为基础，旨在提高部件利用效率，将 CPU 设计成流水线，分为 5 个流水级，采用系统的层次化、结构化、流水级的设计，整体结构如下。可支持的指令集：**{addu, subu, ori, lw, sw, beq, lui, jal,jr,nop}**。

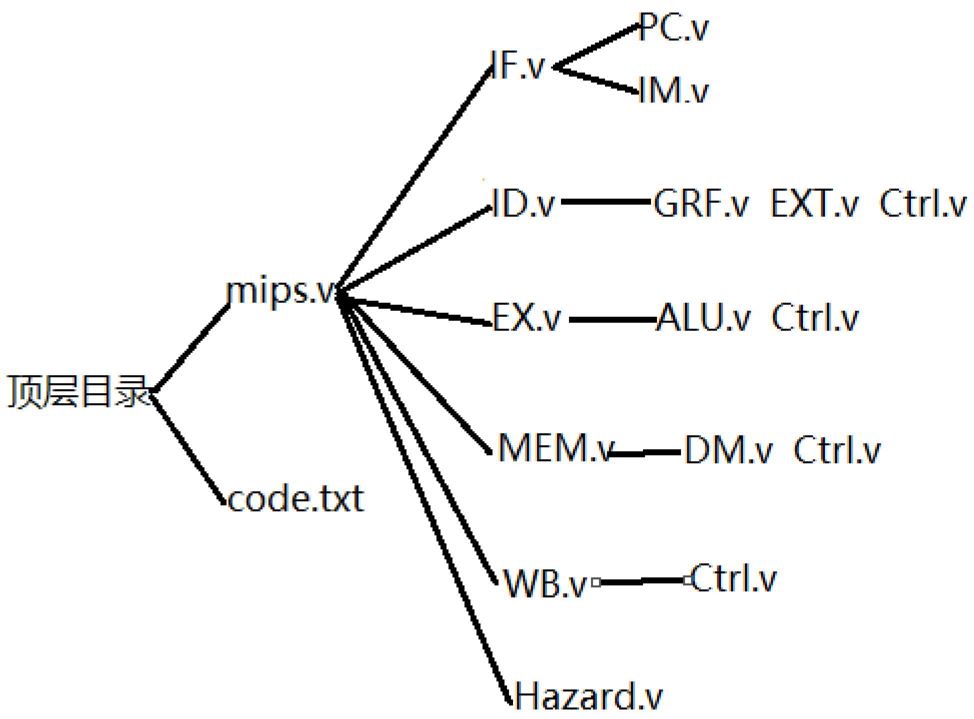


图 1 CPU 模块和层次

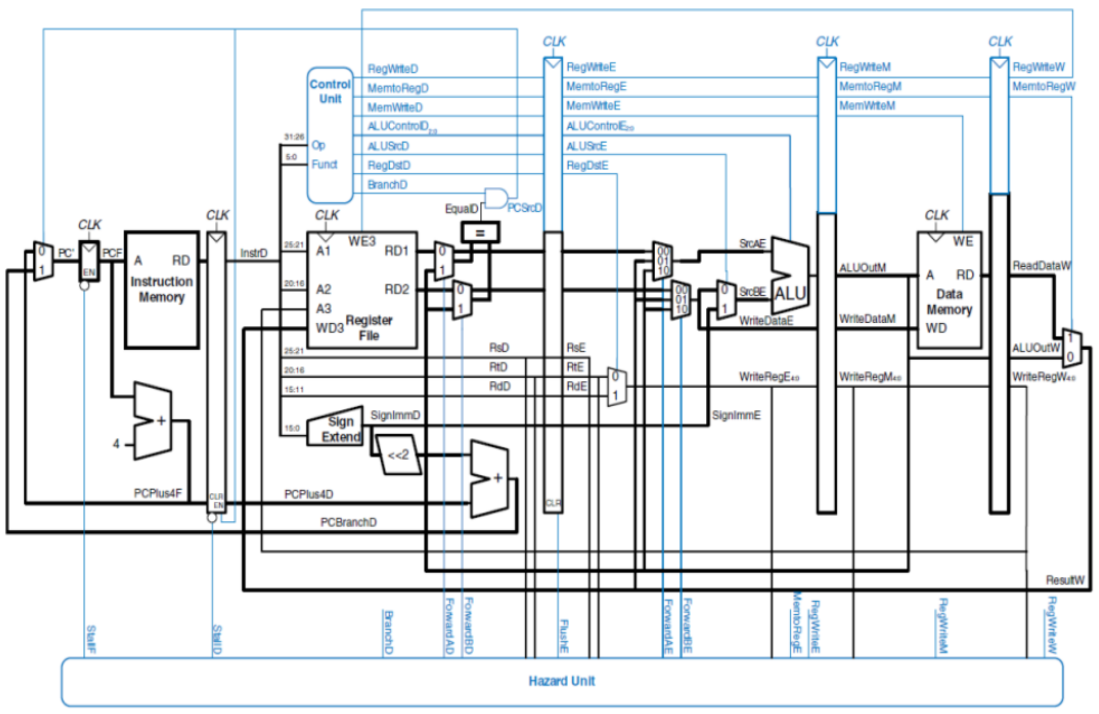


图 2 CPU 数据通路电路图

顶层文件 mips.v 模块接口定义：

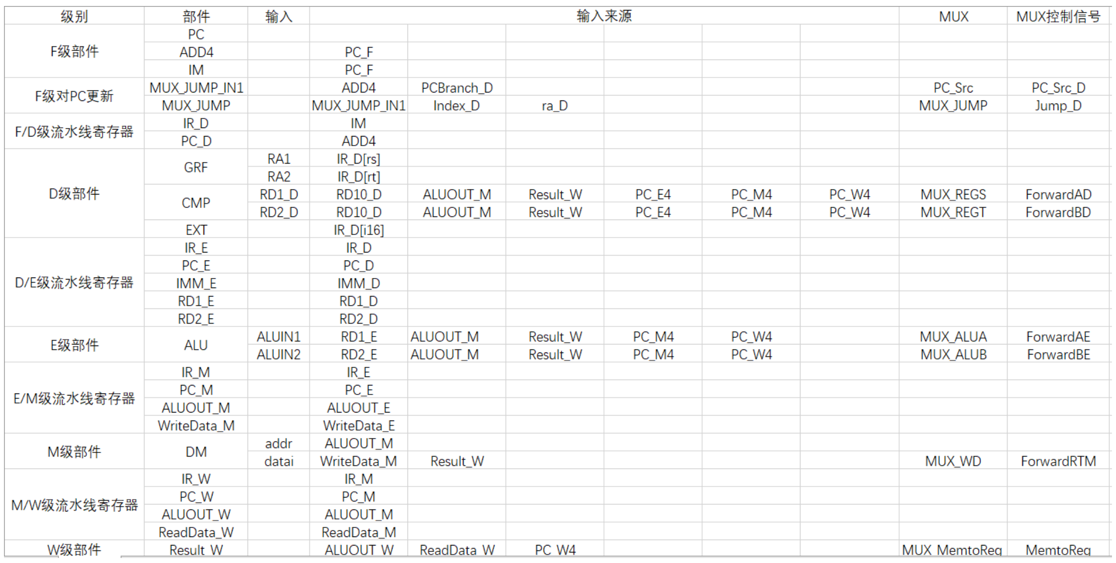
表 1 mips.v 模块接口定义

文件	模块接口定义

mips.v	`module mips(clk, reset);`
	`    input clk,reset;//clk, reset`

2. 数据通路设计

由于 P4 已经完成相关设计，由此得出的数据通路图进行设计。数据通路分为 5 个流水级：IF.v, ID.v, EX.v, MEM.v, WB.v。分别搭建流水级，和各流水级所需要的部件，然后交给 mips 统一管理，自底向上设计。



级别	部件	输入	输入来源						MUX	MUX控制信号
F级部件	PC									
	ADD4		PC_F							
	IM		PC_F							
F级对PC更新	MUX_JUMP_IN1		ADD4	PCBranch_D					PC_Src	PC_Src_D
	MUX_JUMP		MUX_JUMP_IN1	Index_D	ra_D				MUX_JUMP	Jump_D
F/D级流水线寄存器	IR_D		IM							
	PC_D		ADD4							
D级部件	GRF	RA1	IR_D[rs]							
		RA2	IR_D[rt]							
	CMP	RD1_D	RD10_D	ALUOUT_M	Result_W	PC_E4	PC_M4	PC_W4	MUX_REGS	ForwardAD
		RD2_D	RD10_D	ALUOUT_M	Result_W	PC_E4	PC_M4	PC_W4	MUX_REGT	ForwardBD
	EXT		IR_D[i16]							
D/E级流水线寄存器	IR_E		IR_D							
	PC_E		PC_D							
	IMM_E		IMM_D							
	RD1_E		RD1_D							
	RD2_E		RD2_D							
E级部件	ALU	ALUIN1	RD1_E	ALUOUT_M	Result_W	PC_M4	PC_W4		MUX_ALUA	ForwardAE
		ALUIN2	RD2_E	ALUOUT_M	Result_W	PC_M4	PC_W4		MUX_ALUB	ForwardBE
E/M级流水线寄存器	IR_M		IR_E							
	PC_M		PC_E							
	ALUOUT_M		ALUOUT_E							
	WriteData_M		WriteData_E							
M级部件	DM	addr	ALUOUT_M							
		datai	WriteData_M	Result_W					MUX_WD	ForwardRTM
M/W级流水线寄存器	IR_W		IR_M							
	PC_W		PC_M							
	ALUOUT_W		ALUOUT_M							
	ReadData_W		ReadData_M							
W级部件	Result_W		ALUOUT_W	ReadData_W	PC_W4				MUX_MemtoReg	MemtoReg

图 3 CPU 数据通路数据来源

IF.v 模块接口定义：

表 2 IF.v 模块接口定义

文件	模块接口定义

IF.v	module IF(clk,reset,en,PCSrc_D,PCBranch_D,Index_D,ra_D,Jump_D,PC_D,IR_D);
	    input clk,reset,en,PCSrc_D;//clk,reset,PC 使能信号,PC 跳转信号
	    input [31:0]PCBranch_D,Index_D,ra_D;//16 位立即数偏转地址,26 位立即数偏转地址,jr 指令读取的寄存器存的地址
	    input [1:0]Jump_D;//J 类跳转选择信号
	    output reg[31:0]PC_D,IR_D;//D 级流水线寄存器,存入 PC 和指令

PC.v 模块接口定义：

表 3 PC.v 模块接口定义

文件	模块接口定义

PC.v	module PC(clk,reset,en,NPC, PC);
	    input clk,reset,en;//clk,reset,使能信号
	    input [31:0]NPC;//下一个 PC 地址的输入
	    output [31:0]PC;//当前 PC 地址的输出

IM.v 模块接口定义：

表 4 IM.v 模块接口定义

文件	模块接口定义

IM.v	module IM(PC,Instr);
	    input [31:0]PC;//current pc address
	    output [31:0]Instr;//binary instructions

ID.v 模块接口定义：

表 5 ID.v 模块接口定义

文件	模块接口定义

ID.v	module ID(clk,reset,clr,PC_D,IR_D,ALUOUT_M,Result_W,RegWrite_W,WREG_W,PCBranch_D,ra_D,Index_D,PCSrc_D,Jump,RD1_E,RD2_E,IMM_E,IR_E,PC_E,ForwardAD,ForwardBD);
	    input clk,reset,clr;//clk,reset,流水线寄存器清零信号
	    input [31:0]PC_D,IR_D,ALUOUT_M,Result_W;//D 流水级 PC 值，D 流水级指令，M 流水级的 ALU 计算结果，W 流水级的写入寄存器堆的数据
	    input [4:0]WREG_W;//W 级流水线得出的要写入数据的寄存器编号
	    input RegWrite_W;//W 级流水线写寄存器使能信号
	    input [1:0]ForwardAD,ForwardBD;//GRF[rs],GRF[rt]和转发数据选择信号的输入
	    output reg[31:0]RD1_E,RD2_E,IMM_E,IR_E,PC_E;//E 级流水线寄存器
	    output [31:0]PCBranch_D,ra_D,Index_D;//接入 F 级流水线的 16 位立即数得到的偏转地址，读 rs 寄存器得到的 PC 地址，26 位立即数得到的偏转地址
	    output PCSrc_D;//接入 F 级流水线的 PC 选择信号
	    output [1:0]Jump;//接入 F 级流水线 J 类跳转选择信号

GRF.v 模块接口定义：

表 6 GRF.v 模块接口定义

文件	模块接口定义

GRF.v	module GRF(RA1,RA2,WA,WD,clk,reset,RegWrite,RD1,RD2);
	    input [4:0]RA1,RA2,WA;//read address 1, read address 2, write address
	    input clk,reset,RegWrite;//clk, reset, GRF write enable
	    input [31:0]WD;//data to write
	    output [31:0]RD1,RD2;//the data of read address 1, the data of read address 2

EXT.v 模块接口定义：

表 7 EXT.v 模块接口定义

文件	模块接口定义

EXT.v	module EXT(imme,ExtOp,datao);
	    input [15:0]imme;//16 位立即数的输入
	    input [1:0]ExtOp;//扩展选择信号,00:零扩展 01: 符号扩展 10: 低 16 位加载至高 16 位
	    output [31:0]datao;//扩展结果的输出

EX.v 模块接口定义：

表 8 EX.v 模块接口定义

文件	模块接口定义

EX.v	module EX(clk,IR_E,RD1_E,RD2_E,IMM_E,PC_E,WREG_M,Result_W,IR_M,ALUOUT_M,WriteData_M,PC_M,ForwardAE,ForwardBE);
	    input [31:0]IR_E,RD1_E,RD2_E,IMM_E,PC_E;//E 级流水线指令，从 GRF 读出的两个数据，扩展后的立即数，E 级流水线 PC 值
	    input [31:0]Result_W;//W 级流水线写入寄存器堆的数据
	    input [1:0]ForwardAE,ForwardBE;//从 GRF 读出的两个数据和转发进来的数据的选择信号
	    input clk;//clk
	    output reg[31:0]ALUOUT_M,WriteData_M,IR_M,PC_M;//M 级流水线寄存器：ALU 计算结果，写入 DM 的数据，指令，PC 值
	    output reg[4:0]WREG_M;//M 级流水线寄存器存入要写入数据的寄存器编号

ALU.v 模块接口定义：

表 9 ALU.v 模块接口定义

文件	模块接口定义

ALU.v	module ALU(ALUctr,A,B,C);
	    input [1:0]ALUctr;//alu 运算选择信号
	    input [31:0]A,B;//alu 运算的输入数 A，B，00：符号加法运算 01：符号减法运算 10：按位与运算
	    output [31:0]C;//alu 运算结果的输出

MEM.v 模块接口定义：

表 10 MEM.v 模块接口定义

文件	模块接口定义

MEM.v	module MEM(clk,reset,IR_M,ALUOUT_M,WriteData_M,,Result_W,WREG_M,PC_M,ReadData_W,ALUOUT_W,WREG_W,IR_W,PC_W,ForwardRTM);
	    input clk,reset,ForwardRTM;//clk,reset,写入 DM 的数据的选择信号
	    input [31:0]ALUOUT_M,WriteData_M,IR_M,PC_M,Result_W;//M 级流水线的 ALU 计算结果，写入 DM 的数据，指令，PC 值，W 级流水线写入寄存器堆的数据
	    input [4:0]WREG_M;//M 级流水线写寄存器编号
	    output reg[31:0]ReadData_W,ALUOUT_W,IR_W,PC_W;//W 级流水线寄存器，从 DM 读取的数据，ALU 计算结果，指令，PC 值
	    output reg[4:0]WREG_W;//W 级流水线写寄存器编号

DM.v 模块接口定义：

表 11 DM.v 模块接口定义

文件	模块接口定义

DM.v	`module DM(addr,datai,clk,reset,MemWrite,datao);`
	`    input [9:0]addr;//10 位地址的输入`
	`    input [31:0]datai;//要存入 datamemory32 位数据的输入`
	`    input clk,reset;//clk, reset`
	`    input MemWrite;//写使能信号`
	`    output [31:0]datao;//32 位数据的输出`

WB.v 模块接口定义：

表 12 WB.v 模块接口定义

文件	模块接口定义

WB.v	`module WB(ReadData_W,ALUOUT_W,WREG_W,IR_W,PC_W,RegWrite_W,Result_W);`
	`    input [31:0]ReadData_W,ALUOUT_W,IR_W,PC_W;//W 级流水线从 DM 读取的数据，ALU 计算结果，指令，PC 值`
	`    input [4:0]WREG_W;//W 级流水线写寄存器编号`
	`    output RegWrite_W;//W 级流水线写寄存器使能信号`
	`    output [31:0]Result_W;//W 级流水线写入寄存器的数据`

3. 控制器设计

控制器延续之前的设计，增添了指令分类功能，控制的本质就是一个译码的过程，将指令包含的信息转为 CPU 各部分的控制信号，在 Verilog 中有多种实现方式，这里依旧采用与或门阵来实现。

表 13 Controller 信号说明

序号	信号名称	功能描述
1	Instr	32 位指令的输入
2	RegDst	选择寄存器堆(GRF)的写地址 00:rt 01:rd 10:31
3	ALUSrc	选择输入 ALU 的立即数 0:RD2 1:立即数
4	Memtoreg	选择从 DM 读取写入 GRF 的数据 00:aluout 01:dmout 10:GRF[rs]
5	RegWrite	GRF 写使能信号
6	MemWrite	DM 写使能信号
7	Branch	Beq 指令的表征信号
8	Jump	跳转选择信号 00:in1 01:26 位立即数抵制 10:GRF[rs]
9	ExtOp	扩展立即数选择信号 00:零扩展 01:符号扩展 10:加载至高位
10	ALUctr	ALU 运算选择信号 00:符号加法 01:符号减法 10:按位与

表 14 Ctrl.v 模块端口定义

文件	模块接口定义

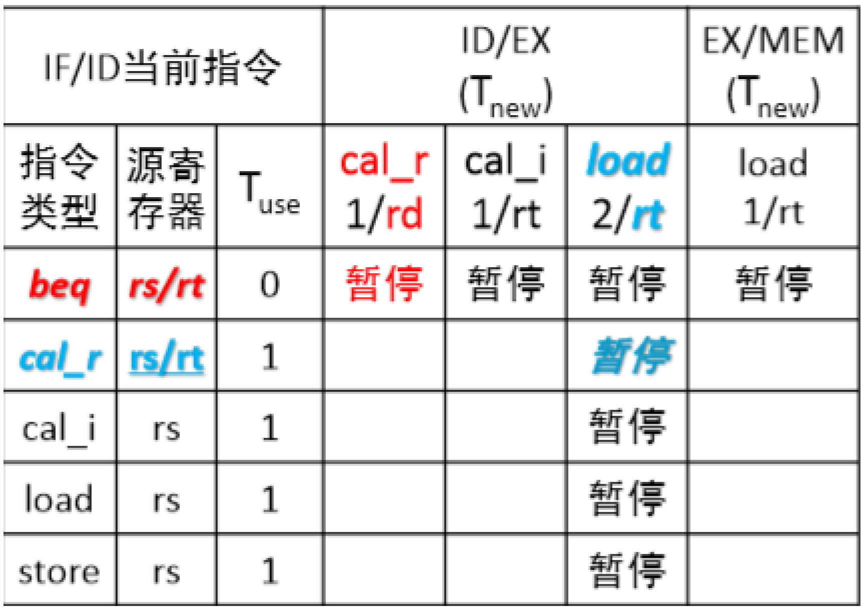
Ctrl.v	module
	ctrl(op,func,RegDst,ALUSrc,MemtoReg,RegWrite,MemWrite,Branch,Jump,ExtOp,ALUctr,b_type,cal_r,cal_i,ld,st);
	    input [5:0]op,func;//6 位 option 和 function
	    output ALUSrc,RegWrite,MemWrite,Branch;//1 位控制信号
	    output [1:0]ExtOp,ALUctr,MemtoReg,RegDst,Jump;//2 位控制信号
	    output b_type,cal_r,cal_i,ld,st;//指令分类信号

表 15 Controller 真值表

op	000000	000000	001101	100011	101011	000100	001111	000011	000000
func	100001	100011	N/A						001000
	addu	subu	ori	lw	sw	beq	lui	jal	jr
RegDst	01	01	00	00	xx	xx	00	10	xx
ALUSrc	0	0	1	1	1	0	1	x	0
MemtoReg	00	00	00	01	00	00	00	10	00
RegWrite	1	1	1	1	0	0	1	1	0
MemWrite	0	0	0	0	1	0	0	0	0
Branch	0	0	0	0	0	1	0	0	0
Jump	00	00	00	00	00	00	00	01	10
ExtOp	xx	xx	00	01	01	01	10	xx	xx
ALUctr	00	01	10	00	00	00	10	xx	00

其中 b_type:beq, jr; cal_r:addu, subu; cal_i:ori, lui; ld:lw; st:sw, jw:jal

4. 冲突模块设计



IF/ID当前指令			ID/EX (T_{new})			EX/MEM (T_{new})
指令类型	源寄存器	T_{use}	cal_r 1/rd	cal_i 1/rt	load 2/rt	load 1/rt
beq	rs/rt	0	暂停	暂停	暂停	暂停
cal_r	rs/rt	1			暂停	
cal_i	rs	1			暂停	
load	rs	1			暂停	
store	rs	1			暂停	

暂停模块设计如图，jal 指令不涉及暂停操作，判断暂停条件如下：

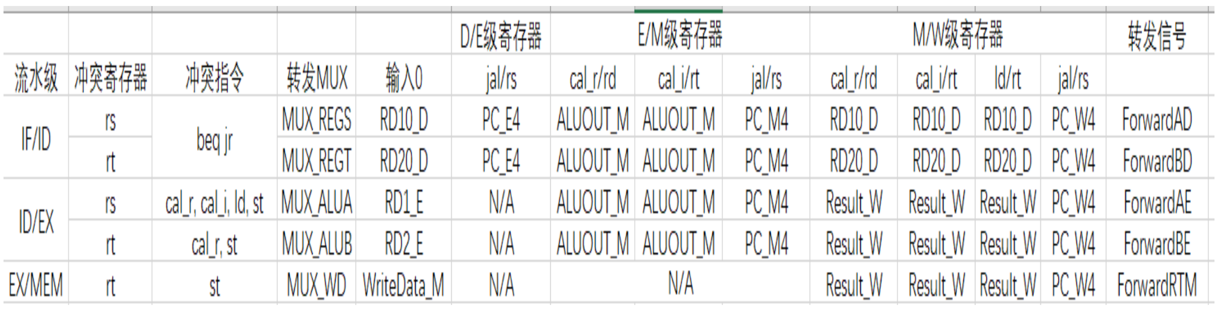
```
 assign
stall_b=b_type&cal_r_E&((IR_D[`rs]==IR_E[`rd])|(IR_D[`rt]
==IR_E[`rd]))|

b_type&cal_i_E&((IR_D[`rs]==IR_E[`rt])|(IR_D[`rt]==IR_E[`
rt]))|

   b_type&ld_E&((IR_D[`rs]==IR_E[`rt])|(IR_D[`rt]==IR_E[`
rt]))|

   b_type&ld_M&((IR_D[`rs]==IR_M[`rt])|(IR_D[`rt]==IR_M[`
rt]));
      assign
stall_cal_r=cal_r_D&ld_E&((IR_D[`rs]==IR_E[`rt])|(IR_D[`r
t]==IR_E[`rt]));
      assign
stall_cal_i=cal_i_D&ld_E&(IR_D[`rs]==IR_E[`rt]);
      assign stall_ld=ld_D&ld_E&(IR_D[`rs]==IR_E[`rt]);
      assign stall_st=st_D&ld_E&(IR_D[`rs]==IR_E[`rt]);
```

```
    assign
stall=stall_b|stall_cal_r|stall_cal_i|stall_ld|stall_st;
```



					D/E级寄存器	E/M级寄存器			M/W级寄存器				转发信号
流水级	冲突寄存器	冲突指令	转发MUX	输入0	jal/rs	cal_r/rd	cal_i/rt	jal/rs	cal_r/rd	cal_i/rt	ld/rt	jal/rs	
IF/ID	rs	beq jr	MUX_REGS	RD10_D	PC_E4	ALUOUT_M	ALUOUT_M	PC_M4	RD10_D	RD10_D	RD10_D	PC_W4	ForwardAD
	rt		MUX_REGT	RD20_D	PC_E4	ALUOUT_M	ALUOUT_M	PC_M4	RD20_D	RD20_D	RD20_D	PC_W4	ForwardBD
ID/EX	rs	cal_r, cal_i, ld, st	MUX_ALUA	RD1_E	N/A	ALUOUT_M	ALUOUT_M	PC_M4	Result_W	Result_W	Result_W	PC_W4	ForwardAE
	rt	cal_r, st	MUX_ALUB	RD2_E	N/A	ALUOUT_M	ALUOUT_M	PC_M4	Result_W	Result_W	Result_W	PC_W4	ForwardBE
EX/MEM	rt	st	MUX_WD	WriteData_M	N/A	N/A			Result_W	Result_W	Result_W	PC_W4	ForwardRTM

转发来源如图，代码设计：

```
assign ForwardAD=

b_type&cal_r_M&IR_D[`rs]!=0&IR_D[`rs]==IR_M[`rd]?1:

b_type&cal_i_M&IR_D[`rs]!=0&IR_D[`rs]==IR_M[`rt]?1:

b_type&cal_r_W&IR_D[`rs]!=0&IR_D[`rs]==IR_W[`rd]?2:

b_type&cal_i_W&IR_D[`rs]!=0&IR_D[`rs]==IR_W[`rt]?2:
      b_type&                                         ld_W
&IR_D[`rs]!=0&IR_D[`rs]==IR_W[`rt]?2:
      b_type&jw_E   &IR_D[`rs]==   31    ?3:
      b_type&jw_M   &IR_D[`rs]==   31    ?4:
      b_type&jw_W   &IR_D[`rs]==   31    ?5:0;
   assign ForwardBD=

b_type&cal_r_M&IR_D[`rt]!=0&IR_D[`rt]==IR_M[`rd]?1:

b_type&cal_i_M&IR_D[`rt]!=0&IR_D[`rt]==IR_M[`rt]?1:

b_type&cal_r_W&IR_D[`rt]!=0&IR_D[`rt]==IR_W[`rd]?2:
   b_type&cal_i_W&IR_D[`rt]!=0&IR_D[`rt]==IR_W[`rt]?2:
         b_type&                                      ld_W
&IR_D[`rt]!=0&IR_D[`rt]==IR_W[`rt]?2:
         b_type&jw_E   &IR_D[`rt]==   31    ?3:
         b_type&jw_M   &IR_D[`rt]==   31    ?4:
         b_type&jw_W   &IR_D[`rt]==   31    ?5:0;
      assign ForwardAE=

   (cal_r_E|cal_i_E|ld_E|st_E)&cal_r_M&IR_E[`rs]!=0&IR_E[
`rs]==IR_M[`rd]?1:

   (cal_r_E|cal_i_E|ld_E|st_E)&cal_i_M&IR_E[`rs]!=0&IR_E[
`rs]==IR_M[`rt]?1:

   (cal_r_E|cal_i_E|ld_E|st_E)&cal_r_W&IR_E[`rs]!=0&IR_E[
`rs]==IR_W[`rd]?2:

   (cal_r_E|cal_i_E|ld_E|st_E)&cal_i_W&IR_E[`rs]!=0&IR_E[
`rs]==IR_W[`rt]?2:
         (cal_r_E|cal_i_E|ld_E|st_E)&                 ld_W
&IR_E[`rs]!=0&IR_E[`rs]==IR_W[`rt]?2:
         (cal_r_E|cal_i_E|ld_E|st_E)&jw_M
&IR_E[`rs]==31      ?3:
         (cal_r_E|cal_i_E|ld_E|st_E)&jw_W
&IR_E[`rs]==31      ?4:0;
      assign ForwardBE=

   (cal_r_E|st_E)&cal_r_M&IR_E[`rt]!=0&IR_E[`rt]==IR_M[`r
d]?1:
   (cal_r_E|st_E)&cal_i_M&IR_E[`rt]!=0&IR_E[`rt]==IR_M[`r
t]?1:

   (cal_r_E|st_E)&cal_r_W&IR_E[`rt]!=0&IR_E[`rt]==IR_W[`r
d]?2:

   (cal_r_E|st_E)&cal_i_W&IR_E[`rt]!=0&IR_E[`rt]==IR_W[`r
t]?2:
         (cal_r_E|st_E)&                              ld_W
&IR_E[`rt]!=0&IR_E[`rt]==IR_W[`rt]?2:
         (cal_r_E|st_E)&jw_M   &IR_E[`rt]==31      ?3:
         (cal_r_E|st_E)&jw_W   &IR_E[`rt]==31      ?4:0;
      assign ForwardRTM=
        st_M&cal_r_W&IR_M[`rt]!=0&IR_M[`rt]==IR_W[`rd]?1:
        st_M&cal_i_W&IR_M[`rt]!=0&IR_M[`rt]==IR_W[`rt]?1:
         st_M& ld_W  &IR_M[`rt]!=0&IR_M[`rt]==IR_W[`rt]?1:
         st_M&jw_W   &IR_M[`rt]==31      ?2:0;
```

5. CPU 测试

测试代码：

```
#jr
  jal lable6
  ori $s1,$zero,4768
  addu $s1,$s1,$zero
  jr $ra
  ori $s1,$zero,347
  lui $s2,1
  addu $s3,$s1,$s2
lable6:
  ori $s1,$zero,3214
  jal lable8
  nop
  ori $s2,$s3,0
  addu $s2,$s2,$zero
  j end
  nop
lable8:
  ori $s3,$zero,12324
  jr $s3
  ori $s3,$zero,43
  addu $s3,$s3,$zero
end:
  addu $s1,$zero,$zero
  jal lable9
  ori $s1,$zero,34
  addu $s1,$s1,$zero
  addu $s2,$s1,$s1
  j main
  nop
lable9:
  jr $ra
  nop
main:
  #sw
  ori $s0,100
  sw $s0,0($zero)
  sw $s0,4($zero)
  ori $s1,$zero,200
  addu $s1,$s1,$zero
```

```
    sw $s1,8($zero)
    sw $s1,12($zero)
    lui $s2,2
    sw $s2,16($zero)
    sw $s2,20($zero)
    addu $s2,$s1,$s0
    sw $s2,24($zero)
    sw $s2,28($zero)
    subu $s3,$s2,$s1
    sw $s3,32($zero)
    sw $s3,36($zero)
    jal lable1
lable1:
    sw $ra,40($zero)
    sw $ra,44($zero)
    lw $s5,44($zero)
    sw $s5,48($zero)
    sw $s5,52($zero)
#addu
    ori $s0,201
    addu $s1,$s0,$s0
    addu $s2,$s1,$s0
    lui $s1,3
    addu $s2,$s1,$s0
    addu $s3,$s1,$s2
    subu $s4,$s3,$s3
    addu $s5,$s4,$s3
    addu $s6,$s5,$s4
    addu $s6,$s5,$s5
    addu $s7,$s6,$s5
    addu $s7,$s7,$s6
    lw $s3,36($zero)
    addu $s4,$s3,$s3
    addu $s5,$s4,$s3
    jal lable2
lable2:
    addu $s0,$ra,$ra
    addu $s1,$ra,$s0
    #subu
    ori $s0,326
    subu $s1,$s0,$s0
    subu $s2,$s1,$s0
    lui $s1,3
    subu $s2,$s1,$s0
    subu $s3,$s1,$s2
    subu $s4,$s3,$s3
    subu $s5,$s4,$s3
    subu $s6,$s5,$s4
    addu $s6,$s5,$s5
    subu $s7,$s6,$s5
    subu $s7,$s7,$s6
    lw $s3,36($zero)
    subu $s4,$s3,$s3
    subu $s5,$s4,$s3
    jal lable3
lable3:
    subu $s0,$ra,$ra
    subu $s1,$ra,$s0
#ori
    ori $s0,201
    ori $s1,$s0,327
    ori $s2,$s0,236
    lui $s1,3
    ori $s2,$s1,327
    ori $s3,$s1,236
    subu $s4,$s3,$s3
    ori $s5,$s4,2442
    ori $s6,$s4,32
    addu $s6,$s5,$s5
    ori $s6,$s6,213
    ori $s7,$s6,432
    lw $s3,52($zero)
    ori $s4,$s3,56
    ori $s5,$s3,432
    jal lable4
lable4:
    ori $s0,$ra,4231
    ori $s1,$ra,234
#lui(无）
#lw
    ori $s0,$zero,324
    lw $s1,0($s0)
    lw $s2,4($s0)
    addu $s1,$s1,$s0
    lw $s2,8($s1)
    lw $s3,12($s1)
    subu $s2,$s1,$s0
    lw $s3,16($s2)
    lw $s4,20($s2)
    lui $s3,1
    lw $s4,24($zero)
    lw $s5,28($zero)
    lw $s1,32($zero)
    lw $s5,36($s1)
    lw $s6,40($s1)
    jal lable5
lable5:
    lw $s0,0($s0)
#beq&j
    ori $t1,$zero,5
    addu $s0,$zero,$t1
    ori $s1,$zero,0
for_1_begin:
    beq $s1,$s0,for_1_end
    ori $t1,$zero,50
    subu $s2,$s3,$t1
    ori $t1,$zero,1
    addu $s1,$s1,$t1
    j for_1_begin
for_1_end:
    ori $t1,$zero,5
    addu $s0,$zero,$t1
    ori $s1,$zero,0
for_2_begin:
    beq $s0,$s1,for_2_end
    ori $t1,$zero,50
    subu $s2,$s3,$t1
    ori $t1,$zero,1
    addu $s1,$s1,$t1
    j for_2_begin
```

```
for_2_end:
    ori $t1,$zero,5
    addu $s0,$zero,$t1
    sw $s0,56($zero)
    lw $s0,56($zero)
    ori $s1,$zero,0
for_3_begin:
    beq $s1,$s0,for_3_end
    ori $t1,$zero,50
    subu $s2,$s3,$t1
    ori $t1,$zero,1
    addu $s1,$s1,$t1
    j for_3_begin
for_3_end:
    ori $s1,$zero,5
    subu $s2,$s1,4
for_4_begin:
    beq $s1,$s2,for_4_end
    subu $s5,$s4,$s2
    ori $t1,$zero,1
    addu $s2,$s2,$t1
    j for_4_begin
for_4_end:
    ori $s0,$zero,0
    ori $s1,$zero,6
```

测试期望：

```
90@00003000: $31 <= 00003008
110@00003004: $17 <= 000012a0
130@0000301c: $17 <= 00000c8e
150@00003020: $31 <= 00003028
190@00003038: $19 <= 00003024
```

```
250@00003040: $19 <= 0000002b
290@00003028: $18 <= 0000002b
310@0000302c: $18 <= 0000002b
370@00003048: $17 <= 00000000
390@0000304c: $31 <= 00003054
410@00003050: $17 <= 00000022
470@00003054: $17 <= 00000022
490@00003058: $18 <= 00000044
550@0000306c: $16 <= 00000064
550@00003070: *00000000 <= 00000064
570@00003074: *00000004 <= 00000064
610@00003078: $17 <= 000000c8
630@0000307c: $17 <= 000000c8
630@00003080: *00000008 <= 000000c8
650@00003084: *0000000c <= 000000c8
690@00003088: $18 <= 00020000
690@0000308c: *00000010 <= 00020000
710@00003090: *00000014 <= 00020000
750@00003094: $18 <= 0000012c
750@00003098: *00000018 <= 0000012c
770@0000309c: *0000001c <= 0000012c
810@000030a0: $19 <= 00000064
810@000030a4: *00000020 <= 00000064
830@000030a8: *00000024 <= 00000064
870@000030ac: $31 <= 000030b4
870@000030b0: *00000028 <= 000030b4
890@000030b0: *00000028 <= 000030b4
910@000030b4: *0000002c <= 000030b4
950@000030b8: $21 <= 000030b4
950@000030bc: *00000030 <= 000030b4
970@000030c0: *00000034 <= 000030b4
1010@000030c4: $16 <= 000000ed
1030@000030c8: $17 <= 000001da
1050@000030cc: $18 <= 000002c7
1070@000030d0: $17 <= 00030000
1090@000030d4: $18 <= 000300ed
1110@000030d8: $19 <= 000600ed
1130@000030dc: $20 <= 00000000
1150@000030e0: $21 <= 000600ed
1170@000030e4: $22 <= 000600ed
1190@000030e8: $22 <= 000c01da
1210@000030ec: $23 <= 001202c7
1230@000030f0: $23 <= 001e04a1
1250@000030f4: $19 <= 00000064
```

```
1290@000030f8: $20 <= 000000c8
1310@000030fc: $21 <= 0000012c
1330@00003100: $31 <= 00003108
1350@00003104: $16 <= 00006210
1370@00003104: $16 <= 00006210
1390@00003108: $17 <= 00009318
1410@0000310c: $16 <= 00006356
1430@00003110: $17 <= 00000000
1450@00003114: $18 <= ffff9caa
1470@00003118: $17 <= 00030000
1490@0000311c: $18 <= 00029caa
1510@00003120: $19 <= 00006356
1530@00003124: $20 <= 00000000
1550@00003128: $21 <= ffff9caa
1570@0000312c: $22 <= ffff9caa
1590@00003130: $22 <= ffff3954
1610@00003134: $23 <= ffff9caa
1630@00003138: $23 <= 00006356
1650@0000313c: $19 <= 00000064
1690@00003140: $20 <= 00000000
1710@00003144: $21 <= ffffff9c
1730@00003148: $31 <= 00003150
1750@0000314c: $16 <= 00000000
1770@0000314c: $16 <= 00000000
1790@00003150: $17 <= 00003150
1810@00003154: $16 <= 000000c9
1830@00003158: $17 <= 000001cf
1850@0000315c: $18 <= 000000ed
1870@00003160: $17 <= 00030000
1890@00003164: $18 <= 00030147
1910@00003168: $19 <= 000300ec
1930@0000316c: $20 <= 00000000
1950@00003170: $21 <= 0000098a
1970@00003174: $22 <= 00000020
1990@00003178: $22 <= 00001314
2010@0000317c: $22 <= 000013d5
2030@00003180: $23 <= 000013f5
2050@00003184: $19 <= 000030b4
2090@00003188: $20 <= 000030bc
2110@0000318c: $21 <= 000031b4
2130@00003190: $31 <= 00003198
2150@00003194: $16 <= 0000319f
2170@00003194: $16 <= 0000319f
2190@00003198: $17 <= 000031fa
```

```
2210@0000319c: $16 <= 00000144
2230@000031a0: $17 <= 00000000
2250@000031a4: $18 <= 00000000
2270@000031a8: $17 <= 00000144
2290@000031ac: $18 <= 00000000
2310@000031b0: $19 <= 00000000
2330@000031b4: $18 <= 00000000
2350@000031b8: $19 <= 00020000
2370@000031bc: $20 <= 00020000
2390@000031c0: $19 <= 00010000
2410@000031c4: $20 <= 0000012c
2430@000031c8: $21 <= 0000012c
2450@000031cc: $17 <= 00000064
2490@000031d0: $21 <= 00000000
2510@000031d4: $22 <= 00000000
2530@000031d8: $31 <= 000031e0
2550@000031dc: $16 <= 00000000
2590@000031dc: $16 <= 00000064
2610@000031e0: $ 9 <= 00000005
2630@000031e4: $16 <= 00000005
2650@000031e8: $17 <= 00000000
2710@000031f0: $ 9 <= 00000032
2730@000031f4: $18 <= 0000ffce
2750@000031f8: $ 9 <= 00000001
2770@000031fc: $17 <= 00000001
2810@00003204: $ 9 <= 00000005
2850@000031f0: $ 9 <= 00000032
2870@000031f4: $18 <= 0000ffce
2890@000031f8: $ 9 <= 00000001
2910@000031fc: $17 <= 00000002
2950@00003204: $ 9 <= 00000005
2990@000031f0: $ 9 <= 00000032
3010@000031f4: $18 <= 0000ffce
3030@000031f8: $ 9 <= 00000001
3050@000031fc: $17 <= 00000003
3090@00003204: $ 9 <= 00000005
3130@000031f0: $ 9 <= 00000032
3150@000031f4: $18 <= 0000ffce
3170@000031f8: $ 9 <= 00000001
3190@000031fc: $17 <= 00000004
3230@00003204: $ 9 <= 00000005
3270@000031f0: $ 9 <= 00000032
3290@000031f4: $18 <= 0000ffce
3310@000031f8: $ 9 <= 00000001
```

```
3330@000031fc: $17 <= 00000005
3370@00003204: $ 9 <= 00000005
3410@000031f0: $ 9 <= 00000032
3430@00003204: $ 9 <= 00000005
3450@00003208: $16 <= 00000005
3470@0000320c: $17 <= 00000000
3530@00003214: $ 9 <= 00000032
3550@00003218: $18 <= 0000ffce
3570@0000321c: $ 9 <= 00000001
3590@00003220: $17 <= 00000001
3630@00003228: $ 9 <= 00000005
3670@00003214: $ 9 <= 00000032
3690@00003218: $18 <= 0000ffce
3710@0000321c: $ 9 <= 00000001
3730@00003220: $17 <= 00000002
3770@00003228: $ 9 <= 00000005
3810@00003214: $ 9 <= 00000032
3830@00003218: $18 <= 0000ffce
3850@0000321c: $ 9 <= 00000001
3870@00003220: $17 <= 00000003
3910@00003228: $ 9 <= 00000005
3950@00003214: $ 9 <= 00000032
3970@00003218: $18 <= 0000ffce
3990@0000321c: $ 9 <= 00000001
4010@00003220: $17 <= 00000004
4050@00003228: $ 9 <= 00000005
4090@00003214: $ 9 <= 00000032
4110@00003218: $18 <= 0000ffce
4130@0000321c: $ 9 <= 00000001
4150@00003220: $17 <= 00000005
4190@00003228: $ 9 <= 00000005
4230@00003214: $ 9 <= 00000032
4250@00003228: $ 9 <= 00000005
4270@0000322c: $16 <= 00000005
4270@00003230: *00000038 <= 00000005
4310@00003234: $16 <= 00000005
4330@00003238: $17 <= 00000000
4390@00003240: $ 9 <= 00000032
4410@00003244: $18 <= 0000ffce
4430@00003248: $ 9 <= 00000001
4450@0000324c: $17 <= 00000001
4490@00003254: $17 <= 00000005
4550@00003240: $ 9 <= 00000032
4570@00003254: $17 <= 00000005
```

```
4590@00003258: $ 1 <= 00000000
4610@0000325c: $ 1 <= 00000004
4630@00003260: $18 <= 00000001
4690@00003268: $21 <= 0000012b
4710@0000326c: $ 9 <= 00000001
4730@00003270: $18 <= 00000002
4770@00003278: $16 <= 00000000
4810@00003268: $21 <= 0000012a
4830@0000326c: $ 9 <= 00000001
4850@00003270: $18 <= 00000003
4890@00003278: $16 <= 00000000
4930@00003268: $21 <= 00000129
4950@0000326c: $ 9 <= 00000001
4970@00003270: $18 <= 00000004
5010@00003278: $16 <= 00000000
5050@00003268: $21 <= 00000128
5070@0000326c: $ 9 <= 00000001
5090@00003270: $18 <= 00000005
5130@00003278: $16 <= 00000000
5170@00003268: $21 <= 00000127
5190@00003278: $16 <= 00000000
    5210@0000327c: $17 <= 00000006
```

6. 思考题

1. 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。

```
测试类型 R_M_RS  subu $1, $2, $3

                 addu $4, $1, $2

         R_M_RT  subu $1, $2, $3

                 addu $4, $2, $3

         R_W_RS  subu $1, $2, $3

                 Instr 无关

                 addu $4, $1, $2

         R_M_RT  subu $1, $2, $3

                 Instr 无关

                 Addu $4, $1, $2
```

```
I_M_RS    ori $1, $2, 100
          Addu $4, $1, $2
I_M_RT    ori $1, $2, 100
          Addu $4, $2, $1
I_W_RS    ori $1, $2, 100
          Instr 无关
          Addu $4, $1, $2
I_W_RT    lw $1, 0($0)
          Instr 无关
          Addu $4, $2, $1
LD_M_RS    lw $1, 0($0)
          Addu $4, $1, $2
LD_M_RT    lw $1, 0($0)
          Addu $4, $2, $1
LD_W_RS    lw $1, 0($0)
          Instr 无关
          Addu $4, $1, $2
LD_W_RT    lw $1, 0($0)
          Instr 无关
          Addu $4, $2, $1

 ST_W_RD    Addu $4, $1, $2
           Instr 无关
           sw $4, 0($0)
  B_M_RS    subu $4, $1, $2
            Instr 无关
            Beq $4, $5, label
```

```
B_M_RT    subu $4, $1, $2
          Instr 无关
          Beq $5, $4, label
B_W_RS    subu $4, $1, $2
          Instr 无关
          Beq $4, $5, label

B_W_RT    subu $4, $1, $2
          Instr 无关
          Beq $5, $4, label
```

除了以上转发情况，还有暂停情况：

```
Addu $1, $2, $3
Beq $1, $4, label

Ori $1, $0, 5
Beq $1, $0, label

Lw $5, 0($0)
Beq $5, $4, label

Lw $2, 0($0)
Instr 无关
Beq $3, $2, label

Lw $2, 0($0)
Addu $4, $2, $3

Lw $2, 0($0)
```

```
ori $4, $2, 5

Lw $2, 0($0)
Lw $3, 0($2)

Lw $2, 0($0)
sw $3, 0($2)
```