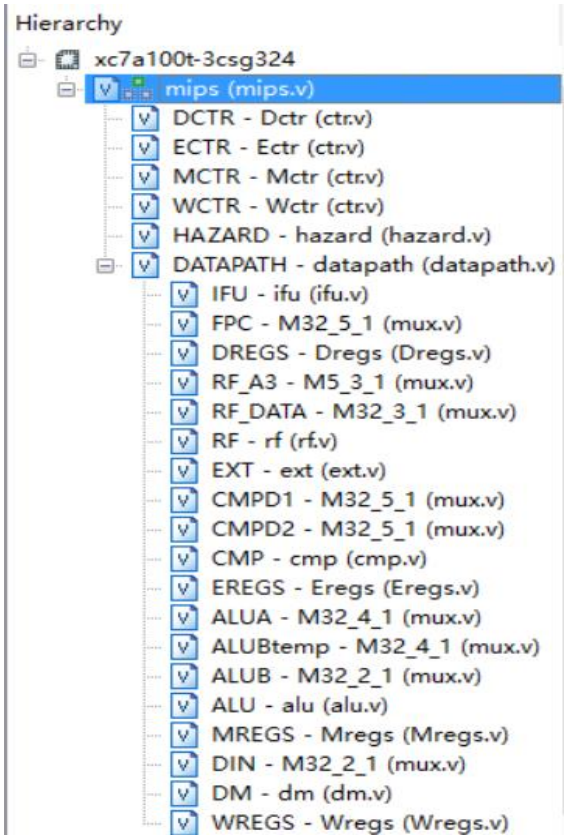
计算机组成原理实验报告

一、 CPU 设计文档

（一） 总体设计



图表 1 模块设计

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	
1			MUX	控制信号	0	1	2		lw	sw	addu/add	subu/sub	ori	lui	beq	j	jal	jr	jalr	lb	lbu	lh	lhu	sb	sh	sll
2	PC		M_nPC	PCSel	ADD4	NPC	MFPCF		ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4/NPC	NPC	NPC	RF.V1	RF.V1	ADD4	ADD4	ADD4	ADD4	ADD4	ADD4	ADD
3	ADD4				PC				PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC
4	IM				PC				PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC
5	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
6		PC8_D			ADD4+4								sh+J2.J3				ADD4+4		ADD4+4							
7		PC4_D			ADD4										ADD4		ADD4		ADD4							
8	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]	IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D[rs]	IR_D
9		A2			IR_D[rt]					IR_D[rt]	IR_D[rt]	IR_D[rt]			IR_D[rt]									IR_D[rt]	IR_D[rt]	IR_D
10	EXT				IR_D[l16]				IR_D[l16]	IR_D[l16]			IR_D[l16]	IR_D[l16]						IR_D[l16]	IR_D[l16]	IR_D[l16]	IR_D[l16]	IR_D[l16]	IR_D[l16]	IR_D[l16]	
11	CMP	D1			MFcmp1D										RF.V1											
12		D2			MFcmp2D										RF.V2											
13	NPC	I26			IR_D[26]										IR_D[l16]	IR_D[26]	IR_D[26]									
14	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	IR_D	IR_D	IR_D	IR_D	IR_D	IR_D	IR_E
15		PC8_E			PC8_D												PC8_D		PC8_D							
16		PC4_E			PC4_D												PC4_D		PC4_D							
17		V1_E			D1				RF.V1	RF.V1	RF.V1	RF.V1	RF.V1							RF.V1	RF.V1	RF.V1	RF.V1	RF.V1	RF.V1	RF.V
18		V2_E			D2					RF.V2	RF.V2	RF.V2												RF.V2	RF.V2	RF.V
19		E32_E			EXT				EXT	EXT			EXT	EXT						EXT	EXT	EXT	EXT	EXT	EXT	
20	ALU	A	M_ALUB	ASel	MFaluaE	IR[10:6]	MFaluaE[4:0]		V1_E	V1_E	V1_E	V1_E	V1_E							V1_E	V1_E	V1_E	V1_E	V1_E	V1_E	IR[1
21		B	M_ALUB	BSel	MFalubE	E32_E			E32_E	E32_E	V2_E	V2_E	E32_E	E32_E						E32_E	E32_E	E32_E	E32_E	E32_E	E32_E	V2_
22	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	IR_E	IR_E	IR_E	IR_E	IR_E	IR_E	IR_E
23		PC8_M			PC8_E												PC8_E		PC8_E							
24		PC4_M			PC4_E												PC4_E		PC4_E							
25		AO_M			ALU				ALU	ALU	ALU	ALU	ALU	ALU						ALU	ALU	ALU	ALU	ALU	ALU	ALU
26		V2_M			MFalubE					V2_E														V2_E	V2_E	
27	DM	A			AO_M				AO_M	AO_M										AO_M	AO_M	AO_M	AO_M	AO_M	AO_M	
28		WD			MFdmdataM					V2_M														V2_M	V2_M	
29	W级寄存器	IR_W			IR_M				IR_M		IR_M	IR_M	IR_M	IR_M			IR_M		IR_M	IR_M	IR_M	IR_M	IR_M			IR_M
30		PC8_W			PC8_M												PC8_M		PC8_M							
31		PC4_W			PC4_M												PC4_M		PC4_M							
32		AO_W			AO_M						AO_M	AO_M	AO_M	AO_M												AO
33		DR_W			DM				DM											DM	DM	DM	DM			
34	DMEXT																									
35	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]	IR_W[rt]	IR_W[rt]	IR_W[rt]	IR_W[rt]			IR_W
36		WD	M_Wdata	WDSel	AO_W	DR_W	PC8_W		DR_W		AO_W	AO_W	AO_W	AO_W			PC8_W		PC8_W	DR_W	DR_W	DR_W	DR_W			AO

数据通路 | 转发暂停 | ctr

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

图表 4 控制器设计

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	func	100001	100011	100000	100010	001000	001001								
2	op	000000	000000	000000	000000	000000	000000	001101	100011	101011	000100	001111	001110	101001	
3		addu	subu	add	sub	jr	jalr	ori	lw	sw	beq	lui	xori	sh	
4	RegDst	1	1	1	1	0	1	0	0	0	0	0	0	0	0rt 1rd 进rf2
5	Asel	00	00	00	00	00	00	00	00	00	00	00	00	00	0grfdata1 1IR[10:6] 2MFaluaE[4:0] 进aluA
6	Bsel	0	0	0	0	0	0	1	1	1	0	1	1	1	0grfdata2 1i32 进aluB
7	MemtoReg	0	0	0	0	0	0	0	1	0	0	0	0	0	
8	RegWrite	1	1	1	1	0	1	1	1	0	0	1	1	0	
9	MemWrite	0	0	0	0	0	0	0	0	1	0	0	0	1	
10	EXTop	00	00	00	00	00	00	00	10	10	11	01	00	10	0无符号 1低16位补0 2有符号 3有符号后左移两位
11	ALUctr	0000	0001	0000	0001	0000	0000	0010	0000	0000	0000	0011	0100	0000	0000加 0001减 0010或 0011B 0100异或 0101逻辑左移 0110逻辑右移 0111算数右移 1000与
12	dmextop	000	000	000	000	000	000	000	000	000	000	000	000	000	000不扩展 001无符号字节扩展 010有符号字节扩展 011无符号半字扩展 100有符号半字扩展
13															
14															
15	func	000000	000000	000000	000000	000000	000000	000011	000010	100000	100100	100001	100101	101000	
16	op	000000	000010	000011	000100	000110	000111	jal	j	lb	lbu	lh	lhu	sb	
17		sll	srl	sra	sllv	srlv	srav	0	0	0	0	0	0	0	
18	RegDst	1	1	1	1	1	1	00	00	00	00	00	00	00	
19	Asel	01	01	01	10	10	10	0	0	1	1	1	1	1	
20	Bsel	0	0	0	0	0	0	0	0	1	1	1	1	0	
21	MemtoReg	0	0	0	0	0	0	1	0	1	1	1	1	0	
22	RegWrite	1	1	1	1	1	1	0	0	0	0	0	0	1	
23	MemWrite	0	0	0	0	0	0	00	00	10	10	10	10	10	
24	EXTop	00	00	00	00	00	00	0000	0000	0000	0000	0000	0000	0000	
25	ALUctr	0101	0110	0111	0101	0110	0111	000	000	010	001	100	011	000	
26	dmextop	000	000	000	000	000	000								
27															
28															
29															
30															
31															

数据通路 | 转发暂停 | ctr

（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

13. Muldiv（乘除单元）

1）端口说明

表格 25Muldiv 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
A[31:0]	I	乘除单元输入数据 1
B[31:0]	I	乘除单元输入数据 2
op[2:0]	I	乘除单元控制信号 000：无效 001：有符号乘 010：无符号乘 011：有符号除 100：无符号除 101：mthi 110：mtlo
HI[31:0]	O	HI 寄存器
LO[31:0]	O	LO 寄存器
busy	O	乘除运算进行信号 1：正在运算 0：无效
start	O	Busy 信号产生信号 1：下一个时钟上升沿 busy 有效 0：无效

2）功能定义

表格 26Muldiv 功能定义

序号	功能名称	功能描述
1	复位	复位信号有效时，HI 和 LO 寄存器和信号被置为 0
2	有符号乘	有符号乘法，其中结果高 32 位保存在 HI，低 32 位保存在 LO，时长 5 周期
3	无符号乘	无符号乘法，其中结果高 32 位保存在 HI，低 32 位保存在 LO，时长 5 周期
4	有符号除	有符号除法，其中结果余数保存在 HI，商保存在 LO，时长 10 周期
5	无符号除	无符号除法，其中结果余数保存在 HI，商保存在 LO，时长 10 周期
6	修改 HI	将 rs 的值存入 HI 寄存器
7	修改 LO	将 rs 的值存入 LO 寄存器
8	读取 HI	读取 HI 寄存器的值，写入 rd
9	读取 LO	读取 LO 寄存器的值，写入 rd

（三） 冒险单元

1. 端口说明

表格 27hazard 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

Busy	I	乘除运算进行信号 1：正在运算 0：无效
start	I	Busy 信号产生信号 1：下一个时钟上升沿 busy 有效 0：无效
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 28hazard 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 29ctr 端口说明

信号名	方向	描述
instruction[31:0]	I	32 位指令
WR_sel	O	grf 写寄存器决定信号 0：rd 1：rt 2：31 号寄存器
A_sel	O	alu 输入数据 A 决定信号 0：MFaluaE 1：IR[10:6] 2：MFaluaE[4:0]
B_sel	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
RegWrite	O	grf 写寄存器信号 1：写寄存器 0：无效

MemWrite	0	写内存 dm 信号 1：写入内存 0：无效
PC_sel	0	PC 跳转信号`ADD4：跳转 PC+4 `NPC：跳转 npc `RFV1：跳转寄存器堆 1
EXTop[1:0]	0	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[3:0]	0	alu 控制信号 0000 加 0001 减 0010 或 0011B 0100 异或 0101 逻辑左移 0110 逻辑右移 0111 算数右移 1000 与 1001 或非 1010 有符号小于置 1 1011 无符号小于置 1
beq	0	Beq 指示信号 0：无效 1：beq
bne	0	bne 指示信号 0：无效 1：bne
Bgez	0	Bgez 指示信号 0：无效 1：Bgez
Blez	0	Blez 指示信号 0：无效 1：Blez
Bgtz	0	Bgtz 指示信号 0：无效 1：Bgtz
blez	0	blez 指示信号 0：无效 1：blez
Muldivop[2:0]	0	乘除模块控制信号 0：无效 1：mult 2：multu 3：div 4：divu 5：mthi 6：mtlo
WDsel[1:0]	0	写入寄存器堆数据选择信号 0：DR_W 1：PC8_W 2：AO_W
Dmexto[2:0]	0	Dm 扩展器控制信号 0：无效 1：lbu 2：lb 3：lhu 4：lh

2. 真值表

func	100001	100011	100000	100010	001000	001001							
op	000000	000000	000000	000000	000000	000000	001101	100011	101011	000100	001111	001110	101001
	addu	subu	add	sub	jr	jalr	ori	lw	sw	beq	lui	xori	sh
RegDst	1	1	1	1	0	1	0	0	0	0	0	0	0
Asel	00	00	00	00	00	00	00	00	00	00	00	00	00
Bsel	0	0	0	0	0	0	1	1	1	0	1	1	1
MemtoReg	0	0	0	0	0	0	0	1	0	0	0	0	0
RegWrite	1	1	1	1	0	1	1	1	0	0	1	1	0
MemWrite	0	0	0	0	0	0	0	0	1	0	0	0	1
EXTop	00	00	00	00	00	00	00	10	10	11	01	00	10
ALUctr	0000	0001	0000	0001	0000	0000	0010	0000	0000	0000	0011	0100	0000
dmextop	000	000	000	000	000	000	000	000	000	000	000	000	000
func	000000	000000	000000	000000	000000	000000							
op	000000	000010	000011	000100	000110	000111	000011	000010	100000	100100	100001	100101	101000
	sll	srl	sra	sllv	srlv	srav	jal	j	lb	lbu	lh	lhu	sb
RegDst	1	1	1	1	1	1	0	0	0	0	0	0	0
Asel	01	01	01	10	10	10	00	00	00	00	00	00	00
Bsel	0	0	0	0	0	0	0	0	1	1	1	1	1
MemtoReg	0	0	0	0	0	0	0	0	1	1	1	1	0
RegWrite	1	1	1	1	1	1	1	0	1	1	1	1	0
MemWrite	0	0	0	0	0	0	0	0	0	0	0	0	1
EXTop	00	00	00	00	00	00	00	00	10	10	10	10	10
ALUctr	0101	0110	0111	0101	0110	0111	0000	0000	0000	0000	0000	0000	0000
dmextop	000	000	000	000	000	000	000	000	010	001	100	011	000

表格 30ctr 真值表

（五） 测试

（一） 测试程序

功能测试：

```
#功能测试
ori $t0, $0, 520
ori $t1, $t0, 233

lui $t0, 520
lui $t1, 0xffff
ori $t1, $t1, 0xffff

#addu, subu
addu $t2, $t0, $t0#++
addu $t3, $t0, $t1#+-
addu $t4, $t1, $t1#--
subu $t5, $t0, $t2#++
subu $t6, $t0, $t1#+-
subu $t7, $t1, $t4#--

#add, sub
ori $s0, $0, 1
ori $s1, $0, 4
ori $s2, $0, 15
lui $t0, 0xfda3
ori $t0, $t0, 0x34f5 #-
ori $t1, $0, 0x234 #+
lui $t2, 0x424
ori $t2, $t2, 32853
add $a0, $t0, $t1 #-+
add $a1, $t0, $t0 #--
add $a2, $t1, $t1 #++
subu $a0, $t0, $t1 #-+
subu $a1, $t1, $t0 #+-
subu $a2, $t1, $t2 #++
subu $a3, $t2, $t1 #++
lui $t3, 0xf39f
subu $a1, $t0, $t3 #--
subu $a2, $t3, $t0

#and, or, xor, nor
ori $s0, $0, 1
ori $s1, $0, 4
ori $s2, $0, 15
lui $t0, 0xfda3
ori $t0, $t0, 0x34f5 #-
ori $t1, $0, 0x234 #+
lui $t2, 0x424
ori $t2, $t2, 32853
and $a0, $t0, $t1 #-+
and $a1, $t0, $t0 #--
and $a2, $t1, $t1 #++
or $a0, $t0, $t1 #-+
or $a1, $t1, $t0 #+-
or $a2, $t1, $t2 #++
xor $a3, $t2, $t1 #++
xor $a0, $t0, $t1 #-+
xor $a1, $t1, $t0 #+-
lui $t3, 0xf39f
or $a1, $t0, $t3 #--
xor $a2, $t3, $t0
nor $a0, $t0, $t1 #-+
nor $a1, $t1, $t0 #+-
nor $a2, $t1, $t2 #++

#mult
lui $t0, 0xfeaf
ori $t0, $t0, 0x5254
lui $t1, 0x0243
ori $t1, $t1, 0x323f
mult $t0, $t0 #--
mfhi $a0
mflo $a1
mult $t1, $t0 #+-
mfhi $a0
mflo $a1
mult $t1, $t1 #-+
mfhi $a1
mflo $a0

#multu
lui $t0, 0xfeaf
ori $t0, $t0, 0x5254
lui $t1, 0x0243
ori $t1, $t1, 0x323f
```

```
mflo $a1
multu $t1,$t0
mfhi $a0
mflo $a1
multu $t1,$t1
mfhi $a0
mflo $a1

#div,divu
lui $t0,0xfeaf
ori $t0,$t0,0x5254
lui $t1,0x0243
ori $t1,$t1,0x323f
lui $t2,0xe120
ori $t2,$t2,0x300
ori $t3,$0,0x99
div $t0,$t1 #-+
mfhi $a0
mflo $a1
div $t1,$t0
mfhi $a0
mflo $a1
div $t0,$t2 #--
mfhi $a0
mflo $a1
div $t2,$t0
mfhi $a0
```

```
mflo $a1
div $t1,$t3 #++
mfhi $a0
mflo $a1
div $t0,$t1 #-+
mfhi $a0
mflo $a1
div $t1,$t0
mfhi $a0
mflo $a1
div $t0,$t2 #--
mfhi $a0
mflo $a1
div $t2,$t0
mfhi $a0
mflo $a1
div $t1,$t3 #++
mfhi $a0
mflo $a1

#mthi,mtlo
lui $t0,0xfeaf
ori $t0,$t0,0x5254
lui $t1,0x0243
ori $t1,$t1,0x323f
mult $t0,$t0
mthi $t0
```

```
mfhi $a0
mflo $a1
mult $t1,$t0
mthi $t1
mfhi $a0
mult $t1,$t1
mtlo $t0
mfhi $a0
mflo $a1

#sll,srl,sra
ori $t0,$t0,123
sll $t0,$t0,4
srl $t0,$t0,4
ori $t1,$t1,0xffff
sll $t1,$t1,2
srl $t1,$t1,2
sra $t0,$t0,3  #+
lui $t0,0xf234
sra $t1,$t0,3 #-
lui $t0,0x23
sra $t0,$t0,4 #+

#sllv,srlv,srav
ori $s0,$0,1
ori $s1,$0,4
ori $s2,$0,15
```

```
lui $t0, 0xfda3
ori $t0, $t0, 0x34f5
ori $t1, $0, 0x234
lui $t2, 0x424
ori $t2, $t2, 32853
sllv $a0, $t0, $s0 #-
sllv $a1, $t1, $s1 #+
sllv $a3, $t2, $s2 #+
srlv $a0, $t0, $s0
srlv $a1, $t1, $s1
srlv $a3, $t2, $s2
srav $a0, $t0, $s0
srav $a0, $t0, $s1
srav $a1, $t1, $s1
srav $a3, $t2, $s2

#slt, sltu
ori $t0, $0, 2
ori $t1, $0, 1
ori $t2, $0, 2
lui $t3, 0xffff
slt $a0, $t2, $t1#++
slt $a1, $t3, $t1#-+
ori $t4, $t3, 0x1234
slt $a2, $t3, $t4 #--
sltu $a0, $t2, $t1 #++
sltu $a1, $t3, $t1 #-+
sltu $a2, $t3, $t4
sltu $a3, $t4, $t3

#jalr
ori $t0, $0, 1
jal jalr_loop
ori $t0, $ra, 0
ori $t2, $0, 2
addu $a1, $a1, $a0
j jalr_end
ori $t6, $t6, 6
jalr_loop:
addu $a0, $0, $ra
jalr $a0, $ra
ori $t4, $a0, 0
ori $t5, $t5, 5
jalr_end:

#addi, addiu, andi, xori
ori $s0, $0, 1
ori $s1, $0, 4
ori $s2, $0, 15
lui $t0, 0xfda3
ori $t0, $t0, 0x34f5 #-
ori $t1, $0, 0x234 #+
lui $t2, 0x424
ori $t2, $t2, 32853
addi $a0, $t0, 0xea4
addi $a1, $t0, -134
addi $a2, $t1, 0xf53f
addi $a3, $t1, 533
addiu $a0, $t0, 0xea4
addiu $a1, $t0, -134
addiu $a2, $t1, 0xf53f
addiu $a3, $t1, 533
andi $a0, $t0, 0xea4
andi $a1, $t0, -134
andi $a2, $t1, 0xf53f
andi $a3, $t1, 533
xori $a0, $t0, 0xea4
xori $a1, $t0, -134
xori $a2, $t1, 0xf53f
xori $a3, $t1, 533

#slti, sltiu
ori $t0, $0, 2
ori $t1, $0, 1
lui $t3, 0xffff
slti $a0, $t1, 2 #+
slti $a1, $t0, 1
slti $a2, $t0, -100
slti $a3, $t3, 2390
sltiu $a0, $t1, 2 #+
sltiu $a1, $t0, 1
```

```
sltiu $a2, $t0, -100 #-
sltiu $a3, $t3, 2390

#sw, sh, sb, lh. lhu, lb, lbu
ori  $20, $0, 1
lui  $21, 0xffff
ori  $21, $21, 0xffff
lui  $3, 0xf3f4
ori  $3, $3, 0x71f2
sw  $3, 0($0)
sh  $3, 8($0)
sh  $3, 10($0)
sb  $3, 4($0)
sb  $3, 5($0)
sb  $3, 6($0)
sb  $3, 7($0)
lh  $2, 0($0)
lh  $2, 2($0)
lhu  $2, 0($0)
lhu  $2, 2($0)
lb  $2, 0($0)
lb  $2, 1($0)
lb  $2, 2($0)
lb  $2, 3($0)
lbu  $2, 0($0)
lbu  $2, 1($0)
lbu  $2, 2($0)
lbu  $2, 3($0)

#bne
ori  $t0, $0, 1
bne  $t0, $t0, bne_label1
ori  $t1, $0, 1
ori  $t2, $0, 2
bne  $t0, $t2, bne_label1
ori  $t3, $0, 3
ori  $t4, $0, 4
bne_label1:ori  $t3, $t3, 3

#blez, bgtz
ori  $a1, $0, 1
lui  $a2, 0xf433
blez  $a1, blez_label1
ori  $t0, $t0, 1
ori  $t1, $t1, 2
ori  $a0, $0, 0
blez  $a0, blez_label1
ori  $t2, $t2, 2
ori  $t3, $t3, 3
blez_label1:ori  $t4, $t4, 4
blez  $a2, bgez_label2
ori  $t5, $t5, 5
ori  $t6, $t6, 6
blez_label2:ori  $t7, $t7, 7

bgtz  $a2, bgtz_label3
ori  $t0, $t0, 1
ori  $t1, $t1, 2
bgtz  $a0, bgtz_label3
ori  $t2, $t2, 2
ori  $t3, $t3, 3
bgtz  $a1, bgtz_label3
ori  $t5, $t5, 5
ori  $t6, $t6, 6
bgtz_label3:ori  $t4, $t4, 4

#bltz, bgez
ori  $a1, $0, 1
lui  $a2, 0xf433
bgez  $a2, bgez_label1
ori  $t0, $t0, 1
ori  $t1, $t1, 2
ori  $a0, $0, 0
bgez  $a0, bgez_label1
ori  $t2, $t2, 2
ori  $t3, $t3, 3
bgez_label1:ori  $t4, $t4, 4
bgez  $a1, bgez_label2
ori  $t5, $t5, 5
ori  $t6, $t6, 6
bgez_label2:ori  $t7, $t7, 7
bltz  $a1, bgez_label3
```

```
ori $t0,$t0,1
ori $t1,$t1,2
bltz $a0,bgez_label3
ori $t2,$t2,2
ori $t3,$t3,3
bltz $a2,bgez_label3
ori $t5,$t5,5
ori $t6,$t6,6
bgez_label3:ori $t4,$t4,4


ori $t0,$0,8
sw $t1,-8($t0)
sw $t2,0($t0)
sw $t3,8($t0)
lw $t4,-8($t0)
lw $t5,0($t0)
lw $t6,8($t0)
```

转发暂停测试：

```
 #测试跳转及延迟槽
jal loop1
ori $t0, $0, 233
nop
j test
ori $t0, $0, 555
ori $t0, $0, 342
test:
ori $t0, 0
ori $t1, $0, 1
beq $t0, $t1, test1
ori $t2, $0, 1
nop
beq $t1, $t2, test1
ori $t3, 3
ori $t4, 4
test1:
ori $t0, $0, 0

#冲突测试
#需要初始化内存单元
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, 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 $t2,$ra,$t0
#JAL-W-RT
jal loop2
nop
addu $t2,$t0,$ra
#MU-M-RS
mult $t0,$t0
mfhi $a0
addu $t2,$a0,$t0
#MU-M-RT
mflo $a0
addu $t2,$t1,$a0
#MU-W-RS
mult $t1,$t1
mfhi $a1
```

```
nop
addu $t2,$a1,$t0
#MU-W-RT
mflo $a2
nop
addu $t0,$t0,$a2
#MU-W-RT
mfhi $a1
sw $a1,444($0)
#R-M-RS
addu $a3,$a1,$a2
mtlo $a3
mflo $a2
#R-W-RS
addu $a3,$a1,$a2
nop
mtlo $a3
mflo $a2
#I-M-RS
ori $a2,$a1,1234
mtlo $a2
mflo $a1
#I-W-RS
xori $a3,$a2,1234
nop
mtlo $a3
mflo $a1
```

```
#LD-W-RT
lw $a0,0($0)
mthi $a0
mfhi $a0
#LD-W-RT
lw $a1,4($0)
nop
mthi $a1
mfhi $a1
#LD-W-RT
lw $a2,8($0)
nop
nop
mthi $a2
mfhi $a2
#MDZZ-MDZZ
mult $a1,$a2
div $a0,$a2
mfhi $1
mflo $2
#
mult $a0,$a1
mthi $a3
mtlo $a0
mfhi $3
mflo $4
#mdzz-R-mthi-RS
```

```
mult $a1, $a2
addu $a3, $1, $2
mthi $a3
mfhi $4
mflo $5
#mdzz-R-X-mthi-RS
multu $4, $5
subu $3, $2, $1
nop
mtlo $3
mfhi $5
mflo $6
#mdzz-R-mdzz-rs
mult $1, $4
srav $8, $4, $3
multu $8, $4
mfhi $1
mflo $2
#mdzz-R-mdzz-RT
mult $2, $6
srav $9, $5, $3
mult $5, $9
mfhi $1
mflo $2
#mdzz-r-x-mdzz-rs
mult $2, $4
srav $8, $4, $2
```

```
multu $8, $5
mfhi $1
mflo $2
#mdzz-r-x-mdzz-rt
mult $4, $6
srav $9, $7, $3
nop
mult $1, $9
mfhi $1
mflo $2
#ori
#ori RSuse在E
#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
#MU-M-RS
mult $t0, $t0
mfhi $a0
ori $t0, $a0, 3415
#MU-W-RS
mult $t1, $t1
mflo $a1
nop
ori $t0, $a1, 328
#mu-w-rt
mflo $1
sw $1, 888($0)
mflo $2
nop
```

```
sw $2, 884($0)
mflo $3
nop
nop
sw $3, 880($0)

#lw
#lwRSuse在E
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在D
#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
```

```
#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
```

```
#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
nop
beq $t0, $ra, wrong
```

```
#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
#JAL-M-RS
jal label1
mthi $31
nop
label1:mfhi $14
jal label2
nop
nop
label2:mthi $31
mfhi $15
jal label3
nop
nop
label3:nop
```

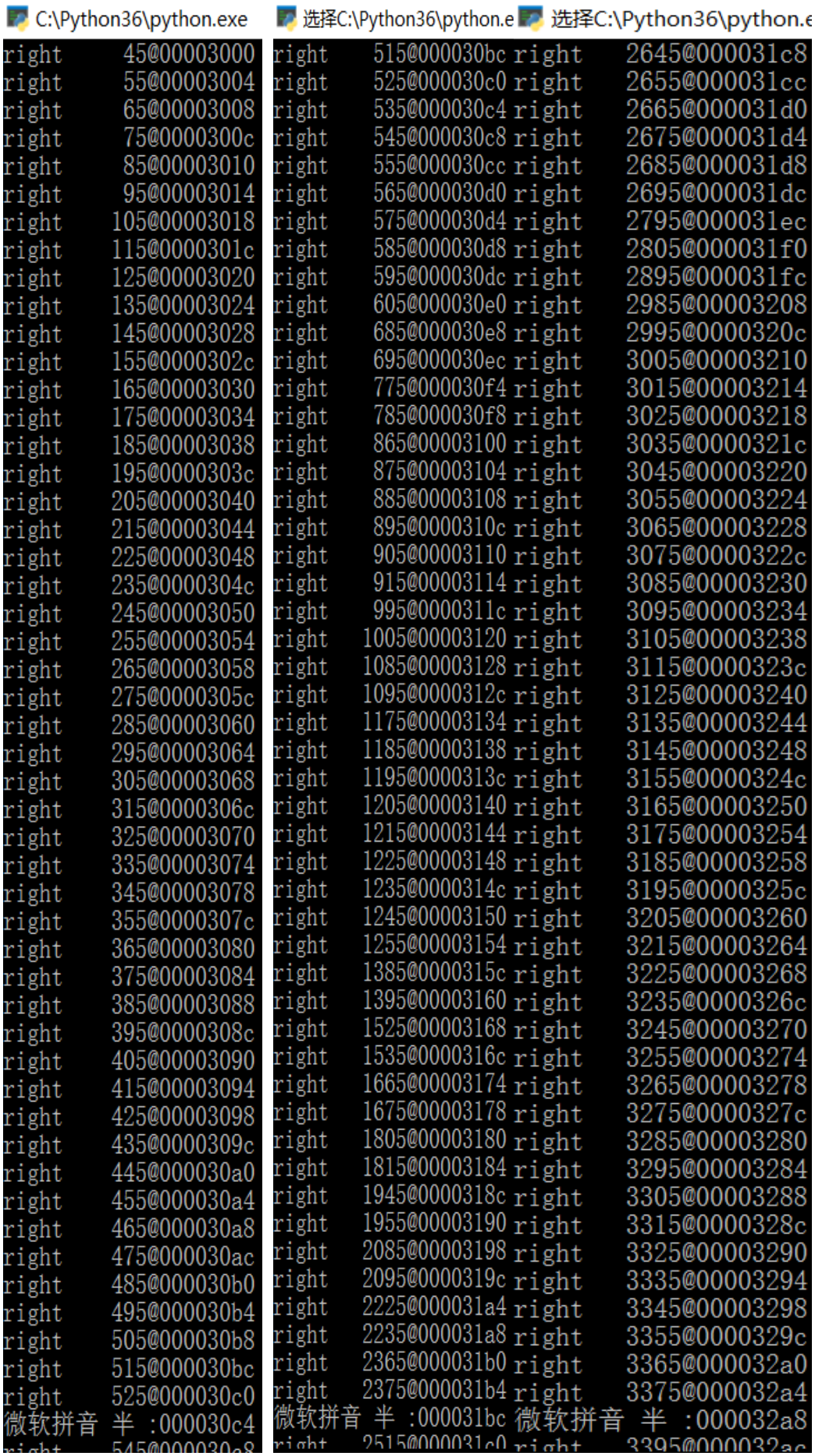
```
mthi $31
mfhi $16

#MU-E-RS
lui $t0, 0xfe23
ori $t0, $t0, 0x24f5
lui $t1, 0x948
ori $t1, $t1, 0x8840
ori $a0, $0, 0
mult $t1, $t1
mfhi $a0
beq $a0, $0, wrong
nop
#MU-E-RT
ori $a1, $0, 0
mflo $a1
beq $0, $a1, wrong
nop
#MU-M-RS
mult $t0, $t0
ori $a0, $0, 0
mfhi $a0
nop
beq $a0, $0, wrong
nop
#MU-M-RT
ori $a1, $0, 0
```

```
mflo $a1
nop
beq $0, $a1, wrong
nop
#MU-W-RS
mult $t0, $t0
ori $a0, $0, 0
mfhi $a0
nop
nop
beq $a0, $0, wrong
nop
#MU-W-RT
ori $a1, $0, 0
mflo $a1
nop
nop
beq $0, $a1, wrong
nop


#sw
#sw, RSuse在E, RTuse在M
#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)
#MU-W-RT
lui $t1, 0x948
ori $t1, $t1, 0x8840
ori $a0, $0, 0
mult $t1, $t1
mfhi $a0
sw $a0, 0($0)
```

```
#mult
#mult,RSuse,RTuse在EE
lui $t1,0xf284
ori $t1,$t1,516
lui $t2,0x344
ori $t2,$t2,233
#R-M-RS
addu $a0,$t1,$t2
mult $a0,$t1
mfhi $a0
mflo $a2
#R-M-RT
addu $a1,$t1,$t1
mult $t2,$a1
mfhi $a1
mflo $a2
#R-W-RS
addu $a0,$t2,$t2
nop
mult $a0,$t1
mfhi $a1
mflo $a2
#R-W-RT
addu $t1,$t1,$t2
nop
mult $a0,$t1
mfhi $a0
mflo $a2
#I-M-RS
ori $a0,$0,0x324
mult $a0,$t0
mfhi $a0
mflo $a2
#I-M-RT
ori $a1,$0,0x4242
mult $t0,$a1
mflo $a1
#I-W-RS
ori $a0,$0,0x3343
nop
mult $a0,$t0
nop
mfhi $a0
mflo $a2
#I-W-RT
ori $a1,$0,0xf242
nop
mult $t0,$a1
nop
mflo $a1
#LD-M-RS
lw $t0,0($0)
mult $t0,$a1
mfhi $a0
#LD-M-RT
lw $t0,4($0)
mult $a0,$t0
mflo $a2
#LD-W-RS
lw $t0,8($0)
nop
mult $t0,$a1
mfhi $a1
mflo $a2
#LD-W-RT
lw $t0,12($0)
nop
mult $a1,$t0
mfhi $a0
mflo $a1
#JAL-M-RS
ori $a0,0x4248
jal loop1
mult $ra,$a0
mfhi $a0
mflo $a1
#JAL-M-RT
jal loop2
mult $a1,$ra
mfhi $a2
mflo $a1
```

```
#JAL-W-RS
jal loop1
nop
mult $ra, $a2
mfhi $a2
mflo $a3
#JAL-W-RT
jal loop2
nop
mult $a3, $ra
mfhi $t0
mflo $a2

j end
nop

wrong:
ori $s0, $0, 2333

loop1:
jr $ra
nop
nop

loop2:
jr $ra
nop
```

（二） 期望输出与实际输出比对


C:\Python36\python.exe
选择C:\Python36\python.e
选择C:\Python36\python.e
right 45@00003000
right 55@00003004
right 65@00003008
right 75@0000300c
right 85@00003010
right 95@00003014
right 105@00003018
right 115@0000301c
right 125@00003020
right 135@00003024
right 145@00003028
right 155@0000302c
right 165@00003030
right 175@00003034
right 185@00003038
right 195@0000303c
right 205@00003040
right 215@00003044
right 225@00003048
right 235@0000304c
right 245@00003050
right 255@00003054
right 265@00003058
right 275@0000305c
right 285@00003060
right 295@00003064
right 305@00003068
right 315@0000306c
right 325@00003070
right 335@00003074
right 345@00003078
right 355@0000307c
right 365@00003080
right 375@00003084
right 385@00003088
right 395@0000308c
right 405@00003090
right 415@00003094
right 425@00003098
right 435@0000309c
right 445@000030a0
right 455@000030a4
right 465@000030a8
right 475@000030ac
right 485@000030b0
right 495@000030b4
right 505@000030b8
right 515@000030bc
right 525@000030c0
微软拼音 半 :000030c4
right 515@000030bc
right 525@000030c0
right 535@000030c4
right 545@000030c8
right 555@000030cc
right 565@000030d0
right 575@000030d4
right 585@000030d8
right 595@000030dc
right 605@000030e0
right 685@000030e8
right 695@000030ec
right 775@000030f4
right 785@000030f8
right 865@00003100
right 875@00003104
right 885@00003108
right 895@0000310c
right 905@00003110
right 915@00003114
right 995@0000311c
right 1005@00003120
right 1085@00003128
right 1095@0000312c
right 1175@00003134
right 1185@00003138
right 1195@0000313c
right 1205@00003140
right 1215@00003144
right 1225@00003148
right 1235@0000314c
right 1245@00003150
right 1255@00003154
right 1385@0000315c
right 1395@00003160
right 1525@00003168
right 1535@0000316c
right 1665@00003174
right 1675@00003178
right 1805@00003180
right 1815@00003184
right 1945@0000318c
right 1955@00003190
right 2085@00003198
right 2095@0000319c
right 2225@000031a4
right 2235@000031a8
right 2365@000031b0
right 2375@000031b4
微软拼音 半 :000031bc
right 2515@000031c0
right 2645@000031c8
right 2655@000031cc
right 2665@000031d0
right 2675@000031d4
right 2685@000031d8
right 2695@000031dc
right 2795@000031ec
right 2805@000031f0
right 2895@000031fc
right 2985@00003208
right 2995@0000320c
right 3005@00003210
right 3015@00003214
right 3025@00003218
right 3035@0000321c
right 3045@00003220
right 3055@00003224
right 3065@00003228
right 3075@0000322c
right 3085@00003230
right 3095@00003234
right 3105@00003238
right 3115@0000323c
right 3125@00003240
right 3135@00003244
right 3145@00003248
right 3155@0000324c
right 3165@00003250
right 3175@00003254
right 3185@00003258
right 3195@0000325c
right 3205@00003260
right 3215@00003264
right 3225@00003268
right 3235@0000326c
right 3245@00003270
right 3255@00003274
right 3265@00003278
right 3275@0000327c
right 3285@00003280
right 3295@00003284
right 3305@00003288
right 3315@0000328c
right 3325@00003290
right 3335@00003294
right 3345@00003298
right 3355@0000329c
right 3365@000032a0
right 3375@000032a4
微软拼音 半 :000032a8
right 3395@000032ac

选择C:\Python36\python.e 选择C:\Python36\python.exe

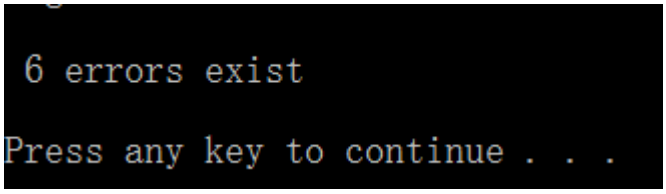
```
right   3395@000032ac right   3895@00003378
right   3405@000032b0 right   3905@0000337c
right   3415@000032b4 right   3915@00003380
right   3425@000032b8 right   3925@00003384
right   3435@000032bc right   3935@00003388
right   3445@000032d0 right   3945@0000338c
right   3455@000032d4 right   3955@00003390
right   3465@000032d8 right   3965@00003394
right   3475@000032c0 right   3975@00003398
right   3485@000032c4 right   3985@0000339c
right   3505@000032cc right   3985@000033a0
right   3515@000032e0 ERROR 1 3995@000033a4: *00000008 <= 000071f2 should be *00000008 <= 71f2
right   3525@000032e4 ERROR 2 4005@000033a8: *00000008 <= 71f271f2 should be *0000000a <= 71f2
right   3535@000032e8 ERROR 3 4015@000033ac: *00000004 <= 000000f2 should be *00000004 <= f2
right   3545@000032ec ERROR 4 4025@000033b0: *00000004 <= 0000f2f2 should be *00000005 <= f2
right   3555@000032f0 ERROR 5 4035@000033b4: *00000004 <= 00f2f2f2 should be *00000006 <= f2
right   3565@000032f4 ERROR 6 4045@000033b8: *00000004 <= f2f2f2f2 should be *00000007 <= f2
right   3575@000032f8 right   4065@000033bc
right   3585@000032fc right   4075@000033c0
right   3595@00003300 right   4085@000033c4
right   3605@00003304 right   4095@000033c8
right   3615@00003308 right   4105@000033cc
right   3625@0000330c right   4115@000033d0
right   3635@00003310 right   4125@000033d4
right   3645@00003314 right   4135@000033d8
right   3655@00003318 right   4145@000033dc
right   3665@0000331c right   4155@000033e0
right   3675@00003320 right   4165@000033e4
right   3685@00003324 right   4175@000033e8
right   3695@00003328 right   4185@000033ec
right   3705@0000332c right   4215@000033f4
right   3715@00003330 right   4225@000033f8
right   3725@00003334 right   4255@00003400
right   3735@00003338 right   4265@00003408
right   3745@0000333c right   4275@0000340c
right   3755@00003340 right   4285@00003410
right   3765@00003344 right   4305@00003418
right   3775@00003348 right   4315@0000341c
right   3785@0000334c right   4325@00003420
right   3795@00003350 right   4355@00003428
right   3805@00003354 right   4365@00003430
right   3815@00003358 right   4385@00003438
right   3825@0000335c right   4395@000034a0
right   3835@00003360 right   4415@000034a8
right   3845@00003364 right   4425@000034ac
right   3855@00003368 right   4445@000034b4
right   3865@0000336c right   4455@000034b8
right   3875@00003370 right   4475@000034c0
right   3885@00003374 right   4485@000034c8
微软拼音 半 :00003378 微软拼音 半 :000034cc
right   3905@0000337c right   4495@000034d0
```

选择C:\Python36\python.e 选择C:\Python36\python.e 选择C:\Python36\python.e

```
right   4505@000034d4 right   5145@000035c8 right   6625@00003714
right   4515@000034d8 right   5155@000035cc right   6645@0000371c
right   4535@000034dc right   5175@000035d0 right   6775@00003724
right   4545@000034e0 right   5185@000035d4 right   6785@00003728
right   4555@000034e4 right   5205@000035dc right   6805@00003730
right   4565@000034e8 right   5215@000035e0 right   6935@0000373c
right   4575@000034ec right   5235@000035e8 right   6945@00003740
right   4625@000034f8 right   5245@000035ec right   6965@00003748
right   4635@00003500 right   5255@000035f0 right   7095@00003754
right   4645@00003504 right   5285@000035f4 right   7105@00003758
right   4675@0000350c right   5295@000035f8 right   7115@0000375c
right   4705@00003518 right   5325@000035fc right   7125@00003760
right   4715@00003520 right   5365@00003604 right   7135@00003764
right   4725@00003524 right   5375@00003608 right   7155@0000376c
right   4725@00003528 right   5415@00003610 right   7165@00003770
right   4745@0000352c right   5495@00003618 right   7175@00003774
right   4745@00003530 right   5505@0000361c right   7185@00003778
right   4765@00003534 right   5515@00003620 right   7205@00003780
right   4765@00003538 right   5525@00003624 right   7215@00003784
right   4785@0000353c right   5605@0000362c right   7235@00003788
right   4785@00003540 right   5625@00003634 right   7245@0000378c
right   4805@00003544 right   5635@00003638 right   7265@00003794
right   4805@00003548 right   5655@00003640 right   7275@00003798
right   4825@0000354c right   5665@00003644 right   7285@0000379c
right   4825@00003550 right   5665@00003648 right   7315@000037a0
right   4845@00003554 right   5685@0000364c right   7355@000037a8
right   4845@00003558 right   5705@00003654 right   7435@000037b0
right   4865@0000355c right   5715@00003658 right   7445@000037b4
right   4865@00003560 right   5745@00003664 right   7525@000037bc
right   4885@00003564 right   5755@00003668 right   7545@000037c4
right   4885@00003568 right   5775@00003670 right   7555@000037c8
right   4905@0000356c right   5785@00003674 right   7555@000037cc
right   4915@00003570 right   5815@00003680 right   7575@000037d0
right   4925@00003574 right   5825@00003684 right   7585@000037d8
right   4935@00003578 right   5855@0000368c right   7605@000037dc
right   4945@0000357c right   5865@00003690 right   7625@000037e8
right   4955@00003580 right   5895@0000369c right   7645@000037ec
right   4965@00003584 right   5905@000036a0 right   7655@000037f0
right   4985@0000358c right   5945@000036b0 right   7665@000037f4
right   4995@00003590 right   6145@000036bc right   7675@000037f8
right   5015@00003598 right   6155@000036c0 right   7685@000037fc
right   5025@0000359c right   6255@000036d0 right   7705@00003804
right   5035@000035a0 right   6265@000036d4 right   7715@00003808
right   5045@000035a4 right   6285@000036dc right   7725@0000380c
right   5055@000035a8 right   6355@000036e4 right   7735@00003810
right   5065@000035ac right   6365@000036e8 right   7755@00003818
right   5085@000035b4 right   6385@000036f0 right   7765@0000381c
right   5095@000035b8 right   6455@000036fc right   7785@00003820
right   5115@000035c0 right   6465@00003700 right   7795@00003824
微软拼音 半 :000035c4 微软拼音 半 :00003708 微软拼音 半 :0000382c
right   5145@000035e8 right   6615@00003710 right   7815@00003830
```

选择C:\Python36\python.e 选择C:\Python36\python.e: 选择C:\Python36\python.e

```
right   7815@00003830 right   8925@000039bc right   9875@00003ae8
right   7835@00003834 right   8935@000039c0 right   9875@00003aec
right   7835@00003838 right   8985@000039d0 right   9895@00003af0
right   7865@00003840 right   9045@000039d4 right   9905@00003af4
right   7875@00003844 right   9085@000039e4 right   9915@00003af8
right   7885@00003848 right   9095@000039e8 right   9925@00003afc
right   7895@0000384c right   9145@000039fc right   9935@00003b00
right   7905@00003850 right   9205@00003a00 right   0015@00003b08
right   7915@00003854 right   9255@00003a14 right   0025@00003b0c
right   7955@00003860 right   9265@00003a18 right   0035@00003b10
right   7965@00003864 right   9315@00003a2c right   0115@00003b18
right   8005@00003870 right   9325@00003a30 right   0125@00003b1c
right   8015@00003874 right   9335@00003a34 right   0135@00003b20
right   8025@00003878 right   9335@00003a38 right   0225@00003b2c
right   8065@00003888 right   9355@00003a3c right   0235@00003b30
right   8075@0000388c right   9365@00003a40 right   0245@00003b34
right   8115@0000389c right   9375@00003a44 right   0335@00003b40
right   8125@000038a0 right   9375@00003a48 right   0345@00003b44
right   8135@000038a4 right   9395@00003a4c right   0355@00003b48
right   8185@000038b8 right   9405@00003a50 right   0435@00003b50
right   8195@000038bc right   9415@00003a54 right   0445@00003b54
right   8245@000038d0 right   9425@00003a5c right   0455@00003b58
right   8255@000038d4 right   9445@00003a60 right   0535@00003b60
right   8265@000038d8 right   9455@00003a64 right   0545@00003b64
right   8315@000038e4 right   9465@00003a68 right   0635@00003b74
right   8325@000038e8 right   9475@00003a70 right   0645@00003b78
right   8335@000038ec right   9495@00003a74 right   0655@00003b7c
right   8385@000038fc right   9495@00003a78 right   0745@00003b8c
right   8395@00003900 right   9515@00003a7c right   0755@00003b90
right   8405@00003904 right   9515@00003a80 right   0845@00003b98
right   8455@00003918 right   9535@00003a84 right   0855@00003b9c
right   8465@0000391c right   9545@00003a8c right   0945@00003ba4
right   8475@00003920 right   9565@00003a90 right   0955@00003ba8
right   8535@00003930 right   9575@00003a98 right   1045@00003bb4
right   8545@00003934 right   9595@00003a9c right   1055@00003bb8
right   8585@00003944 right   9605@00003aa0 right   1065@00003bbc
right   8595@00003948 right   9625@00003aa4 right   1155@00003bc8
right   8645@0000395c right   9625@00003aa8 right   1165@00003bcc
right   8665@00003968 right   9645@00003aac right   1175@00003bd0
right   8675@0000396c right   9655@00003ab4 right   1185@00003bd4
right   8705@0000397c right   9675@00003ab8 right   1265@00003bdc
right   8715@00003980 right   9685@00003ac0 right   1275@00003be0
right   8755@00003994 right   9705@00003ac4 right   1285@00003be4
right   8765@00003998 right   9705@00003ac8 right   1365@00003bec
right   8775@0000399c right   9745@00003acc right   1375@00003bf0
right   8785@000039a0 right   9755@00003ad4 right   1385@00003bf4
right   8795@000039a4 right   9775@00003ad8 right   1495@00003c00
right   8805@000039a8 right   9785@00003adc right   1505@00003c04
right   8885@000039b0 right   9795@00003ae0 right   1515@00003c08
微软拼音 半 :000039bc 微软拼音 半 :00003ae8 微软拼音 半 :00003c14
right   8935@000039c0 right   9875@00003aec
```



（三） 结论

期望输出与实际输出相同。

二、 思考题

（一） 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？

乘除指令中我们需要将其计算结果分别存入 HI 寄存器和 LO 寄存器，所以需要独立的特殊寄存器 HI 寄存器和 LO 寄存器。乘除法在实际实现中是比其他的运算需要更多的时间，倘若整合到 ALU，势必会大大增加流水线的周期，对于 cpu 的执行效率来说是不利的，所以需要有单独的乘除法部件。

独立的 LO 和 HI 我认为一是为了配合乘除相关指令，二是由于在汇编指令中乘除相关指令一半会配合取 HI 寄存器和 LO 寄存器的操作，为了转发方便节省周期数，需要有独立的 HI 和 LO 寄存器。

（二） 参照你对延迟槽的理解，试解释“乘除槽”。

跳转指令需要经过一个周期的判断和地址输送才能改变下一个地址来读取下一条指令，所以势必需要一个周期进行暂停，所以在一个周期去处理一条不相关指令时可行的，可以提高 cpu 的执行效率。我们称这个跳转的空档为延迟槽，原因来源于跳转指令的延迟。

而对于乘除指令，其在进入 E 级部件后也会产生 5-10 个周期的延迟，其相关指令势必要进行暂停，但是在这个空档中是可以处理一些不相关指令的，也可以提高 cpu 的执行效率。所以相比较延迟槽，这个进行乘除法操作的延迟，可以被称为乘除槽。

（三） 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?

MEM 部件的延迟最高，而流水线的执行周期是根据整条流水线中延迟最高的部件决定的，倘若把数据扩展模块放在 DM 之后的话，其势必会增加 MEM 部件的延迟，同时增加了整条流水线的执行周期，对于 cpu 的运行效率有很大影响。

但是一定会转发（无论转发过去是否被选择），数据的经过通路是相同的，所以对于时

钟周期实际上没有影响，放在 DM 之后也可以。

如果考虑了异常和中断，比如还要几个周期才能产生数据，那么是让 load 指令停在 E/M 还是 M/W，可能会有影响？（猜测）

（四） 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）

当进行 sb 指令时，若 DM 按字来访问，需要提出该字地址的整个字后，修改相应字节的内容，再存回去。而若把 DM 分成四个字节，在存储字节操作上我们可以直接将相应字节存入相应地址，而不需要把数据读出再操作。虽然在我们实现中看不出来时间的差异，但是在实际中按字节访问在效率和时间利用上是比按字访问内存更有优势的。

（五） 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？

我的 CPU 设计风格是规划者型，对于每个控制信号以不同指令来判断信号。

以转发、暂停为例，分析新指令是否需要 rs 和 rt 和是否写寄存器来分析冲突，然后构建相应的 Tuse 和 Tnew 信号，就可以完成一条新增指令的冲突分析。在最初的冲突构建中已经考虑到了所有的冲突情况，把所有条件抽象出来。所以在新增指令到来时，只需要将指令添加到相对应的抽象信号中即可。

（六） 你对流水线 CPU 设计风格有何见解？

1. 设计中善于使用宏定义是很有优势的，用简洁明了的名称来代替二进制编码虽然在代码开头需要更多处理，但是在后面的编写过程，尤其是条件判断和信号生成上会更加的直观明显。
2. 在刚开始设计的时候，最好能够考虑到它的可扩展性，体现在控制信号的位数上，或者是否可能整合某几个信号，比如 cmp 单元我用多个信号分别表示大于 0、小等 0、大等 0、小于 0、等于 0、不等于 0、大于、小等、大等、小于、等于、不等，其实完全可以用两个多位信号，分别表示和 0 比较结果和两个数比较结果。

（七） 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(add,addu,sub,subu,sllv,srlv,srav,and,or,xor,nor,slt,sltu)

编号	类型	前序指令	冲突时钱前序指令位置	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
					Addu $4,$1,$5
						从 W 级将 AO_W 转发到 E 级部件 alua

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	从 W 级将 AO_W 转发到 E 级部件 alub

					Addu $3,$4,$1	

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(addi,addiu,andi,ori,xori,lui,slti,sltiu)

编号	类型	前序指令	冲突时钱前序指令位置	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 Ori $3,$1,1	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 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	从 W 级将 PC8_W 转发到 E 级部件 alua

					Ori $2,$31,1	

Load 型(lw,lh,lhu,lb,lbu)

编号	类型	前序指令	冲突时钱前序指令位置	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	从 W 级将 PC8_W 转发到 E 级部件 alua

					nop Lw $2,0($31)	

Store 型(sw,sh,sb)

编号	类型	前序指令	冲突时钱前序指令位置	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-	load	WB	rt	Lw $1,0($2)	从 W 级将 DR_W 转发

	RT				Sw $1,0($3)	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

B(Beq,bne,bgez,bgtz,blez,bltz)

编号	类型	前序指令	冲突时钱前序指令位置	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-	load	WB	rs	Lw $1,0($2)	暂停

	RS				nop Beq $1,$3,loop	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-RT	jal	EX	rt	Jal loop Beq $2,$31,loop	从 E 级将 PC8_E 转发到 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

J(Jr,jalr)

编号	类型	前序指令	冲突时钱前序指令位置	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	从 M 级将 AO_M 转发到 D 级部件 cmpD1

					Beq $1,$4,loop	
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) Beq $1,$3,loop	暂停 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

s(sll,srl,sra)

编号	类型	前序指令	冲突时钱前序指令位置	冲突寄存器	测试样例	解决方法
1	R-M-RS	cal_r	MEM	rt	Addu $1,$2,$3 sll $4,$1,1	从 M 级将 AO_M 转发到 E 级部件 alub

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