

CPU 设计文档

一、 数据通路设计

(1) pc（程序计数器）

模块端口说明如下：

表 1 pc 端口说明

序号	信号名	方向	描述
1	Clk	I	时钟信号
2	Reset	I	复位信号
3	En	I	使能信号
4	NPC[31:0]	I	下一个PC值
5	PC[31:0]	O	当前的PC值

模块功能定义如下：

表 2 pc 功能定义

序号	功能名称	功能描述
1	更新PC值	当时钟上升沿到来时，将NPC写入PC
2	输出	输出当前PC的值

(2) im（指令存储器）

模块端口说明如下：

表 3 im 端口说明

序号	端口名	方向	描述
1	Addr[11:2]	I	当前PC的[11:2]位
2	Instr[31:0]	O	指令存储器中以Addr为地址的指令

模块功能定义如下：

表 4 im 功能定义

序号	功能名称	功能描述
1	初始化	将code.txt中的内容读入指令存储器中
2	输出	输出指令存储器中Addr所对应地址的指令的值

(2) grf（通用寄存器组）

模块端口说明如下：

表 5 grf 端口说明

序号	信号名	方向	描述
1	RAddr1[4:0]	I	读寄存器地址1
2	RAddr2[4:0]	I	读寄存器地址2
3	WAddr[4:0]	I	写寄存器地址
4	WData[31:0]	I	写入寄存器的数据
5	RegWrite	I	寄存器写使能信号
6	Clk	I	时钟信号
7	RData1[31:0]	O	输出地址RAddr1的寄存器中的数据
8	RData2[31:0]	O	输出地址RAddr2的寄存器中的数据

模块功能定义如下：

表 6 grf 功能定义

序号	功能名称	功能描述
1	读寄存器	输出端口RData1和RData2分别输出以输入信号RAddr1和RAddr2为地址的寄存器中的数据
2	写寄存器	当始终上升沿到来时，若写使能信号为1，则将输入信号WData中的数据写入以输入信号WAddr为地址的寄存器中

(3) alu（算术逻辑单元）

模块端口说明如下：

表 7 alu 端口说明

序号	信号名	方向	描述
1	Data1[31:0]	I	参与ALU运算的第一个值
2	Data2[31:0]	I	参与ALU运算的第二个值
3	ALUOp[2:0]	I	ALU功能的选择信号 000 : ALU进行加法运算 001 : ALU进行减法运算 010 : ALU进行或运算 011 : ALU进行与运算
4	ALUResult[31:0]	O	ALU的计算结果

模块功能定义如下：

表 8 alu 功能定义

序号	功能名称	功能描述
1	加法运算	$ALUResult = Data1 + Data2$
2	减法运算	$AUResult = Data1 - Data2$
3	与运算	$ALUResult = Data1 \& Data2$
4	或运算	$ALUResult = Data1 Data2$

(3) dm（数据存储器）

模块端口说明如下：

表 9 dm 端口说明

序号	信号名	方向	描述
1	Addr[4:0]	I	数据存储器读写的地址
2	WData[31:0]	I	将要写进数据存储器的数据
3	MemWrite	I	数据存储器的写使能端
4	Clk	I	时钟信号
5	Reset	I	复位信号
6	RData[31:0]	O	输出从数据存储器中读取的值

模块功能定义如下：

表 10 dm 功能定义

序号	功能名称	功能描述
1	读数据存储器	输出端口Data输出数据存储器在地址为MemAddr处的数据
2	写数据存储器	当时钟上升沿到来时，若MemWrite为1，且Reset信号为0，则将输入信号MemData中的数据写入数据存储器在MemAddr所对应的地址中
3	复位	当时钟上升沿到来时，若Reset信号为1，将数据存储器的内容置为0

(4) EXT（数据扩展单元）

模块端口说明如下：

表 11 ext 端口说明

序号	信号名	方向	描述
1	In[15:0]	I	扩展单元的输入信号
2	ExtOp[1:0]	I	扩展方式的选择信号 00：进行符号扩展 01：进行零扩展 10：进行低位零扩展
3	Out[31:0]	O	扩展单元的输出信号

模块功能定义如下：

表 12 ext 功能定义

序号	功能名称	功能描述
1	符号扩展	输出信号的低16位与输入信号相同，高16位为输入信号的符号位
2	零扩展	输出信号的低16位与输入信号相同，高16位为0
3	低位零扩展	输出信号的高16位与输入信号相同，低16位为0

(5) NPC

模块端口定义如下：

表 13 NPC 端口说明

序号	端口名称	方向	说明
1	PCplus4[31:0]	I	当前的PC值加4
2	I_Addr[25:0]	I	当前32位指令的低26位
3	R_Addr[31:0]	I	保存在寄存器中的地址
4	NPCSel[1:0]	I	选择下一个pc的值 00 : 选择PC+8作为下一个PC的值 01 : 选择PC + sign_extend(Addr[15:0] 02)作为下一个PC的值 10 : 选择{PC[31:28], Addr}作为下一个PC的值 11: 选择R_Addr作为下一个PC的值
5	Equal	I	beq的分支条件是否成立
6	NPC	O	下一个PC的值（若发生分支或跳转）
7	PCplus8	O	当前的PC值加8

模块功能定义如下：

表 14 NPC 功能定义

序号	功能名称	功能描述
1	输出	根据选择信号输出下一个PC的值（若发生分支或跳转）

二、 控制器设计

(1) 控制器

ctrl_D 端口说明如下：

序号	信号名	方向	描述
1	Instr[31:0]	I	F/D级流水线寄存器中的Instr值
2	EXTOp[1:0]	0	扩展单元的功能选择信号
3	NPCSel[1:0]	0	NPC的功能选择信号
4	isBranch	0	是否发生跳转或分支

ctrl_E 端口说明如下：

序号	信号名	方向	描述
1	Instr[31:0]	I	D/E级流水线寄存器的Instr值
2	ALUOp[2:0]	0	ALU功能选择信号
3	ALUSrc[1:0]	0	ALU的运算数据选择信号

ctrl_M 端口说明如下：

序号	信号名	方向	描述
1	Instr[31:0]	I	E/M级流水线寄存器的Instr值
2	MemWrite	0	数据存储器的写使能信号

ctrl_W 端口说明如下：

序号	信号名	方向	描述
1	Instr[31:0]	I	M/W级流水线寄存器中的Instr值
2	RegWrite	0	寄存器的写使能信号
3	RegDst[1:0]	0	寄存器写入地址选择信号
4	RegSrc[1:0]	0	寄存器写入数据选择信号

控制信号真值表如下：

	addu	subu	jr	ori	lw	sw	beq	lui	jal
Op	000000	000000	000000	001101	100011	101011	000100	001111	000011
Funct	100001	100011	001000						
nPC_Op[1]	0	0	1	0	0	0	0	0	1
nPC_Op[0]	0	0	1	0	0	0	1	0	0
RegWrite	1	1	0	1	1	0	0	1	1
RegDst[1]	0	0	x	0	0	x	x	0	1
RegDst[0]	1	1	x	0	0	x	x	0	0
RegSrc[1]	0	0	x	0	0	x	x	0	1
RegSrc[0]	0	0	x	0	1	x	x	0	0
ExtOp[1]	x	x	x	0	0	0	x	1	x
ExtOp[0]	x	x	x	1	0	0	x	0	x
ALUOp[1]	0	0	x	1	0	0	0	1	x
ALUOp[0]	0	1	x	0	0	0	0	0	x
ALUSrc	0	0	x	1	1	1	0	1	x
MemWrite	0	0	0	p	0	1	0	0	0

控制信号意义如下：

序号	控制信号	意义
1	NPCSel[1:0]	控制分支的信号，分支指令需要将该信号置为1
2	RegWrite	寄存器写使能信号，但需要些寄存器时将此信号置为1
3	RegDst[1:0]	选择寄存器的写入地址， 当此信号为00时，选择指令的rt字段（[20:16]）为寄存器的写入地址； 当此信号为01时，选择指令的rd字段（[15:11]）为寄存器的写入地址； 当此信号为10时，选择0x1f为寄存器的写入地址
4	RegSrc[1:0]	选择寄存器的写入数据， 当此信号为00时，选择ALU的计算结果作为寄存器堆的写入值； 当此信号为01时，选择从数据存储器中取出的信号作为寄存器堆的写入值； 当此信号为10时，选择PC+4作为寄存器堆的写入值
5	ExtOp[1:0]	Ext功能选择信号，根据指令需要进行的扩展类型来设置为相应的值
6	ALUOp[2:0]	ALU功能选择信号，根据指令需要执行的运算种类来设置相应的值
7	ALUSrc	当此信号为0时，选择指令的rt字段（[20:16]）为地址的寄存器中的数据作为ALU的第二个运算数； 当此信号为1时，选择经扩展后的立即数作为ALU的第二个运算数。
8	MemWrite	数据存储器写使能信号，当需要写数据存储器时将此信号置为1

(2) 阻塞控制器

阻塞控制器端口说明如下：

序号	信号名	方向	描述
1	FD_Instr[31:0]	I	F/D级流水线寄存器中的Instr值
2	DE_Instr[31:0]	I	D/E级流水线寄存器中的Instr值
3	EM_Instr[31:0]	I	E/M级流水线寄存器中的Instr值
4	StallPC	O	是否阻塞PC
5	StallFD	O	是否阻塞F/D流水线寄存器
6	FlushDE	O	是否清空D/E流水线寄存器中的值

阻塞发生条件如下：

IF/ID当前指令			ID/EX (Tnew)				EX/MEM (Tnew)				MEM/	
指令类型	源寄存器	Tuse	cal_r 1/rd	cal_i 1/rt	load 2/rt	jal 0/31	cal_r 0/rd	cal_i 0/rt	load 1/rt	jal 0/31	cal_r 0/rd	cal_i 0/rt
beq	rs/rt	0	暂停	暂停	暂停				暂停			
jr	rs	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							
	rt	2										

(3) 转发控制器

转发控制器端口定义如下：

序号	信号名	方向	描述
1	FD_Instr[31:0]	I	F/D级流水线寄存器中的Instr值
2	DE_Instr[31:0]	I	D/E级流水线寄存器中的Instr值
3	EM_Instr[31:0]	I	E/M级流水线寄存器中的Instr值
4	MW_Instr[31:0]	I	M/W级流水线寄存器中的Instr值
5	BypassDrs	O	转发信号1，控制D阶段相等比较的输入
6	BypassDrt	O	转发信号2，控制D阶段相等比较的输入
7	BypassErs	O	转发信号3，控制ALU的输入
8	BypassErt	O	转发信号4，控制ALU的输入
9	BypassMrt	O	转发信号5，控制DM的写入值

转发条件如下：

流水级	源寄存器	涉及指令	转发MUX	控制信号	输入0
IF/ID	rs	beq, jr	MUXB_D_rs	BypassDrs	D_GPR_RData1
	rt	beq	MUXB_D_rt	BypassDrt	D_GPR_RData2
ID/EX	rs	cal_r, cal_i, ld, st	MUXB_E_rs	BypassErs	DE_RData1
	rt	cal_r, st	MUXB_E_rt	BypassErt	DE_RData2
EX/MEM	rt	store	MUXB_M_rt	BypassMrt	EM_M_Wdata
ID/EX	EX/MEM (Tnew)				
jal 0/31	cal_r 0/rd	cal_i 0/rt	jal 0/31		
DE_PCplus8	EM_ALUOut	EM_ALUOut	EM_PCplus8		
DE_PCplus8	EM_ALUOut	EM_ALUOut	EM_PCplus8		
	EM_ALUOut	EM_ALUOut	EM_PCplus8		
	EM_ALUOut	EM_ALUOut	EM_PCplus8		
					
MEM/WB (Tnew)					
cal_r 0/rd	cal_i 0/rt	load 0/rt	jal 0/31		
MW_ALUOut	MW_ALUOut	MW_DM_Rdata	MW_PCplus8		
MW_ALUOut	MW_ALUOut	MW_DM_Rdata	MW_PCplus8		
MW_ALUOut	MW_ALUOut	MW_DM_Rdata	MW_PCplus8		
MW_ALUOut	MW_ALUOut	MW_DM_Rdata	MW_PCplus8		
MW_ALUOut	MW_ALUOut	MW_DM_Rdata	MW_PCplus8		

三、测试程序

```
#cal_r - cal_r
lui $s0, 0x5432
ori $s0, $s0, 0x9843
lui $s1, 0x5843
ori $s1, $s1, 0x6543
addu $s2, $s1, $s0
subu $s3, $s1, $s0
addu $s4, $s3, $s0
addu $s5, $s0, $s4
subu $s6, $s5, $s0
subu $s7, $s0, $s6
addu $t0, $s6, $s7
addu $t1, $t0, $s7
subu $t2, $t1, $t1
addu $t3, $t2, $t2
addu $0, $t2, $t3
subu $t4, $t3, $0
addu $t5, $0, $t4
subu $0, $t5, $0
addu $t6, $0, $0
```

```
#cal_r - cal_i
lui $s0, 0x4325
ori $s0, $s0, 0x5432
lui $s1, 0x8954
ori $s1, $s1, 0x3459
addu $s2, $s1, $s0
subu $s3, $s1, $s0
ori $s4, $s3, 0x8932
subu $s5, $s0, $s4
addu $s6, $s0, $s4
ori $s7, $s5, 0x8954
ori $t0, $s6, 0x5894
addu $t1, $s7, $t0
lui $t2, 0x8594
subu $t3, $t2, $s0
```

```
lui $t4, 0x8594
addu $t5, $t4, $s0
lui $t4, 0x3498
lui $t5, 0x8265
addu $t6, $t4, $t5
lui $0, 0x8943
addu $t7, $s0, $0
ori $0, $0, 0x8594
ori $t8, $0, $t7
addu $t9, $t8, $0
```

```
#cal_r - load
lui $s0, 0x8694
ori $s1, $s0, 0x8754
lui $s2, 0x8954
ori $s3, $s3, 0x5437
lui $s4, 0x2349
ori $s5, $0, 0x8954
lui $s6, 0x8954
ori $s7, $0, 0x7843
sw $s0, 0($0)
sw $s1, 4($0)
sw $s2, 8($0)
sw $s3, 12($0)
sw $s4, 16($0)
sw $s5, 20($0)
sw $s6, 24($0)
sw $s7, 28($0)
lw $t0, 0($0)
addu $t1, $t0, $s0
addu $t2, $t0, $t1
lw $t2, 4($0)
subu $t3, $t2, $t1
subu $t4, $t3, $t2
lw $t3, 8($0)
lw $t4, 8($0)
addu $t5, $t3, $t4
addu $t6, $t3, $t4
subu $t7, $t6, $t6
lw $t8, 12($t7)
ori $t9, $0, 0x0010
addu $t0, $t9, $0
lw $t0, 0($t0)
```

```

addu $t1, $t0, $t9
ori $t2, $0, 0x0002
addu $t3, $t2, $t9
addu $t4, $t1, $t0
lw $t4, 0($t3)
subu $t5, $t3, $t4
addu $t6, $t3, $t2
addu $t7, $t6, $t2
lw $t8, 0($t6)
lw $t8, 0($t7)
addu $t9, $t8, $t7
subu $t9, $t8, $t8

```

```

#cal_r - jal
ori $s0, $0, 0x8593
ori $ra, $0, 0x5439
jal next1
addu $s2, $s0, $ra
next1:
addu $s3, $ra, $s0
addu $ra, $s3, $s0
jal next2
next2:
subu $s4, $ra, $s0
subu $s5, $s4, $ra
addu $ra, $ra, $s0
jal next3
addu $s6, $s5, $ra
subu $s6, $s5, $ra
nop
next3:
subu $s7, $s6, $ra
addu $ra, $ra, $ra
jal next4
nop
addu $t0, $ra, $s0
next4:
addu $t1, $ra, $ra
subu $t2, $ra, $ra

```

```

#cal_i - cal_i
lui $s0, 0x5493
ori $s1, $s0, 0x8954

```

```
ori $s2, $s1, 0x6543
ori $s3, $s1, 0x2345
lui $s4, 0x8954
ori $s5, $s4, 0x3555
ori $s6, $s4, 0x8592
ori $s6, $s6, 0x3478
ori $s7, $s6, 0x8594
lui $0, 0x5439
ori $t0, $0, 0x8954
ori $0, $t0, 0x5432
lui $t1, 0x5342
ori $t2, $0, 0x2345
```

```
#cal_i - load
lui $s0, 0x8954
ori $s1, $s0, 0x7854
lui $s2, 0x2345
ori $s3, $0, 0x8954
ori $s4, $2, 0x8432
sw $s0, 0($0)
sw $s1, 4($0)
sw $s2, 8($0)
sw $s3, 12($0)
sw $s4, 16($0)
lw $t0, 0($0)
ori $t1, $t0, 0x8954
ori $t1, $t0, 0x2345
lw $t1, 4($0)
lw $t1, 8($0)
ori $t2, $t1, 0x4352
ori $t3, $t1, 0x5435
lw $0, 4($0)
ori $t4, $0, 0x8594
ori $t4, $0, 0x000c
lw $t5, 0($t4)
ori $t5, $t5, 0x5432
ori $t6, $t5, 0x3543
ori $t6, $0, 0x0010
nop
lw $t7, 0($t6)
ori $t8, $t7, 0x8943
lui $t9, 0x0000
lw $t9, 0($t9)
```

```
ori $t0, $t9, 0x8943
ori $t1, $t9, 0x7548
```

```
#cal_i - jal
lui $s0, 0x8954
ori $s0, $s0, 0x5432
ori $ra, $s0, 0x8943
jal next1
ori $s1, $ra, 0x8594
next1:
ori $s2, $ra, 0x2345
ori $ra, $ra, 0x8943
jal next2
next2:
ori $s3, $ra, 0x5483
ori $s4, $ra, 0x3456
lui $ra, 0x5843
jal next3
ori $ra, $ra, 0x5438
ori $s6, $ra, 0x8954
nop
next3:
ori $s6, $ra, 0x8965
ori $ra, $ra, 0x3453
jal next4
next4:
ori $ra, $ra, 0x1453
ori $s7, $ra, 0x5436
```

```
#load - load
ori $s0, $0, 0x0000
ori $s1, $0, 0x0004
ori $s2, $0, 0x0008
ori $s3, $0, 0x000c
sw $s0, 0($0)
sw $s1, 4($0)
sw $s2, 8($0)
sw $s3, 12($0)
lw $t0, 0($0)
lw $t1, 0($t0)
lw $t2, 4($t0)
lw $t2, 0($t2)
lw $t3, 4($t2)
```

```

lw $t4, 8($t2)
lw $0, 0($t4)
lw $t5, 8($0)
lw $t6, 0($t5)
lw $t6, 0($t6)
lw $t6, 4($t6)
lw $t6, 8($0)

#load - jal
ori $s0, $0, 0x0004
ori $s1, $0, 0x0008
la $s2, next1
sw $s0, 4($0)
sw $s1, 0($s2)
ori $ra, $0, 0x0004
jal next1
nop
next1:
lw $t0, 0($ra)
la $s3, next2
sw $s1, 0($s3)
ori $ra, $0, 0x0004
jal next2
lw $t1, 0($ra)
next2:
lw $t2, 0($ra)
la $s4, next3
sw $s1, 0($s4)
ori $ra, $0, 0x0008
jal next3
next3:
lw $t3, -4($ra)
lw $t4, -4($ra)

#store - cal_r
lui $s0, 0x8953
ori $s1, $s0, 0x2354
ori $s2, $0, 0x0004
lui $s3, 0x8955
addu $s3, $s0, $s1
sw $s3, 0($0)
sw $s3, 4($0)
lui $s4, 0x2364

```

```

subu $s4, $s3, $s1
sw $s4, 0($0)
sw $s4, 4($0)
ori $s5, $0, 0x0008
addu $s5, $s2, $0
sw $s4, 0($s5)
sw $s4, 4($s5)
lui $s6, 0x0004
subu $s6, $s4, $s4
sw $s3, 0($s6)
sw $s3, 4($s6)

```

```

#store - cal_i
lui $s0, 0x8954
ori $s1, $s0, 0x5438
lui $s1, 0x7853
sw $s1, 0($0)
sw $s1, 4($0)
ori $s1, $0, 0x5345
ori $s1, $0, 0x3465
sw $s1, 0($0)
sw $s1, 4($0)
ori $s2, $0, 0x0008
ori $s2, $0, 0x0004
sw $s2, 0($s2)
sw $s2, 4($s2)
ori $s3, $0, 0x0004
lui $s3, 0x0000
sw $s0, 0($s3)
sw $s0, 4($s3)

```

```

#store - load
ori $s0, $0, 0x0004
ori $s1, $0, 0x0008
ori $s2, $0, 0x000c
ori $s3, $0, 0x0010
sw $s0, 0($0)
sw $s1, 4($0)
sw $s2, 8($0)
sw $s3, 12($0)
ori $t0, $0, 0x8954
nop
lw $t0, 0($0)

```



```

sw $t0, 16($0)
sw $t0, 20($0)
ori $t1, $0, 0x2444
lw $t1, 0($0)
sw $t0, 0($t1)
sw $t0, 4($t1)
lw $t2, 4($0)
lw $t3, 8($0)
sw $t2, 0($t3)
sw $t3, 4($t3)
lw $t4, 12($0)
sw $t4, 4($t4)

```

```

#store - jal
lui $s0, 0x9823
ori $s0, $s0, 0x4532
ori $ra, 0x5325
jal next1
sw $s0, 0($ra)
next1:
sw $s0, 4($ra)
jal next2
next2:
sw $s0, 0($ra)
sw $s0, 8($ra)
jal next3
sw $ra, 0($0)
sw $ra, 4($0)
nop
next3:
sw $ra, 8($0)
jal next4
sw $ra, 0($ra)
next4:
sw $ra, 4($ra)

```

```

#beq - cal_r
lui $s0, 0x4583
ori $s1, $s0, 0x6542
addu $s2, $s0, $s1
beq $s2, $0, next1
nop
addu $s3, $s2, $s1

```

```

next1:
subu $s4, $s3, $s1
beq $s4, $s1, next2
nop
addu $s4, $s3, $s2
next2:
subu $s4, $s3, $s3
beq $s4, $0, next3
nop
lui $t0, 0x3543
next3:
subu $s3, $s1, $s1
nop
beq $s3, $0, next4
nop
lui $t1, 0x4835
next4:
addu $s3, $s0, $s1
addu $s4, $s0, $s1
beq $s3, $s4, next5
nop
lui $t2, 0x9543
next5:

```

```

#beq - cal_i
lui $s0, 0x5342
ori $s1, $s0, 0x5432
ori $s2, $s0, 0x5432
beq $s1, $s2, next1
nop
lui $s0, 0x4533
next1:
lui $s2, 0x5432
lui $s3, 0x5432
beq $s2, $s3, next2
nop
lui $s2, 0x5432
next2:
lui $s4, 0x9543
nop
beq $s4, $0, next3
nop
lui $s0, 0x5432

```

```

next3:

#beq - load
lui $s0, 0x8753
ori $s1, $s0, 0x8543
sw $s0, 0($0)
sw $s1, 4($0)
lw $s2, 0($0)
beq $s2, $0, next1
nop
lui $t0, 0x8594
next1:
lw $s3, 0($0)
nop
beq $s3, $0, next2
nop
lui $t2, 0x8954
next2:
lw $s4, 0($0)
lw $s5, 0($0)
beq $s4, $s5, next3
nop
lui $t3, 0x8453
next3:
lui $s7, 0x8753
lw $s6, 0($0)
lw $s7, 4($0)
beq $s6, $s7, next4
nop
lui $t4, 0x8754
next4:

#beq - jal
lui $s0, 0x8754
addu $ra, $0, $0
jal next1
nop
next1:
beq $ra, $0, next2
nop
lui $t0, 0x4532
next2:

```

```

#jr - cal_r
la $s0, next1
la $s1, next2
addu $t0, $s0, $0
jr $t0
nop
next1:
addu $t1, $s1, $0
nop
jr $t1
nop
next2:

```

```

#jr - cal_i
addu $s0, $0, $0
lui $s0, 0x3010
jr $s0
nop
addu $s1, $s0, $0
ori $s1, $0, 0x3020
jr $s1
lui $t0, 0x5432
nop

```

```

#jr - load
ori $s0, $0, 0x3020
ori $s1, $0, 0x3030
sw $s0, 0($0)
sw $s1, 4($0)
addu $s2, $0, $0
lw $s2, 0($0)
jr $s2
nop
addu $s3, $0, $0
lw $s3, 4($0)
nop
jr $s3
nop

```

```

#jr - jal
la $s3, next2
lui $s0, 0x8754
addu $ra, $0, $0

```

```
jal next1
nop
next1:
jr $s3
nop
lui $t0, 0x4532
next2:
```

思考题

1、

指令类型	前序指令	前前序指令	冲突寄存器	解决方法	测试序列
cal_r	cal_r / cal_i		rs / rt	转发	subu \$s3, \$s1, \$s0 addu \$s4, \$s3, \$s0
		cal_r / cal_i	rs / rt	转发	subu \$s7, \$s0, \$s6 addu \$t0, \$s6, \$s7 addu \$t1, \$t0, \$s7
	load		rs / rt	阻塞	lw \$t0, 0(\$0) addu \$t1, \$t0, \$s0
		load	rs / rt	转发	lw \$t0, 0(\$0) addu \$t1, \$t0, \$s0 addu \$t2, \$t0, \$t1
	jal		31	转发	jal next1 addu \$s2, \$s0, \$ra next1:
		jal	31	转发	jal next3 addu \$s6, \$s5, \$ra subu \$s6, \$s5, \$ra
cal_i	cal_r / cal_i		rs	转发	ori \$s1, \$s0, 0x8954 ori \$s2, \$s1, 0x6543
		cal_r / cal_i	rs	转发	ori \$s6, \$s4, 0x8592 ori \$s6, \$s6, 0x3478 ori \$s7, \$s6, 0x8594
	load		rs	阻塞	lw \$t1, 8(\$0) ori \$t2, \$t1, 0x4352
		load	rs	转发	lw \$t1, 8(\$0) ori \$t2, \$t1, 0x4352 ori \$t3, \$t1, 0x5435
	jal		31	转发	jal next1 ori \$s1, \$ra, 0x8594 next1:
		jal	31	转发	jal next3 ori \$ra, \$ra, 0x5438 ori \$s6, \$ra, 0x8954

load	cal_r / cal_i		rs	转发	addu \$t2, \$t0, \$t1 lw \$t2, 4(\$0)
		cal_r / cal_i	rs	转发	subu \$t3, \$t2, \$t1 subu \$t4, \$t3, \$t2 lw \$t3, 8(\$0)
	load		rs	阻塞	lw \$t0, 0(\$0) lw \$t1, 0(\$t0)
		load	rs	转发	lw \$t2, 4(\$t0) lw \$t2, 0(\$t2) lw \$t3, 4(\$t2)
	jal		31	转发	jal next2 lw \$t1, 0(\$ra)
		jal	31	转发	jal next2 lw \$t1, 0(\$ra) next2: lw \$t2, 0(\$ra)
store	cal_r / cal_i		rs / rt	转发	addu \$s3, \$s0, \$s1 sw \$s3, 0(\$0)
		cal_r / cal_i	rs / rt	转发	addu \$s3, \$s0, \$s1 sw \$s3, 0(\$0) sw \$s3, 4(\$0)
	load		rs	阻塞	lw \$t1, 0(\$0) sw \$t0, 0(\$t1)
	load		rt	转发	lw \$t0, 0(\$0) sw \$t0, 16(\$0)
		load	rs / rt	转发	lw \$t0, 0(\$0) sw \$t0, 16(\$0) sw \$t0, 20(\$0)
	jal		31	转发	jal next1 sw \$s0, 0(\$ra)
		jal	31	转发	jal next2 next2: sw \$s0, 0(\$ra) sw \$s0, 8(\$ra)

beq	cal_r / cal_i		rs / rt	阻塞	addu \$s2, \$s0, \$s1 beq \$s2, \$0, next1
		cal_r / cal_i	rs / rt	转发	subu \$s3, \$s1, \$s1 nop beq \$s3, \$0, next4
	load		rs / rt	阻塞	lw \$s2, 0(\$0) beq \$s2, \$0, next1
		load	rs / rt	阻塞	lw \$s3, 0(\$0) nop beq \$s3, \$0, next2
	jal		31	转发	
		jal	31	转发	jal next1 nop next1: beq \$ra, \$0, next2
jr	cal_r / cal_i		rs	阻塞	addu \$t0, \$s0, \$0 jr \$t0
		cal_r / cal_i	rs	转发	addu \$t1, \$s1, \$0 nop jr \$t1
	load		rs	阻塞	lw \$s2, 0(\$0) jr \$s2
		load	rs	阻塞	lw \$s3, 4(\$0) nop jr \$s3
	jal		31	转发	
		jal	31	转发	jal next1 nop next1: jr \$s3