单周期 CPU 设计说明文档

一、数据通路设计

1、数据通路具体实现

见表格文档

二、模块规格

1、PC

（1）端口说明

端口名	方向	描述
clk	I	时钟信号
reset	I	同步复位信号
npc[31:0]	I	32 位存储数据
pc[31:0]	O	输出 32 位 PC 存储的数据

（2）功能定义

序号	功能名称	功能描述
1	复位	当 reset 为高电平时 PC 同步复位为 0x0000_3000
2	写入 npc	时钟上升沿写入 npc 端口传入的值

2、NPC

（1）端口说明

端口名	方向	描述
pc[31:0]	I	当前指令 PC 值
NPCOp[1:0]	I	计算 NPC 功能选择
Imm[25:0]	I	26 位立即数
ra[31:0]	I	32 位寄存器值
npc[31:0]	O	输出 32 位 NPC 的值

pc4[31:0]	O	输出 32 位 PC+4 的值

（2）功能定义

序号	功能名称	功能描述
1	次地址计算	NPCOp=00，计算输出顺序地址（PC+4） NPCOp=01，计算输出 beq 地址 NPCOp=10，计算输出 jal 地址 NPCOp=11，计算输出 jr 地址

3、IM

（1）端口说明

端口名	方向	描述
addr[31:0]	I	输入 32 位地址
Instr[31:0]	O	输出 32 位指令

（2）功能定义

序号	功能名称	功能描述
1	取出指令	根据 addr 的值从 IM 中取出指令

4、GRF

（1）端口说明

端口名	方向	描述
A1[4:0]	I	第 1 个读出寄存器的编号
A2[4:0]	I	第 2 个读出寄存器的编号
A3[4:0]	I	写入寄存器的编号
RD1[31:0]	O	A1 指向寄存器的值
RD2[31:0]	O	A2 指向寄存器的值
WD[31:0]	I	写入寄存器的值
WE	I	写入使能
clk	I	时钟信号

reset	I	同步复位信号

（2）功能定义

序号	功能名称	功能描述
1	复位	reset=1 时在时钟上升沿所有寄存器复位为 0x0000_0000
2	读出寄存器	A1 和 A2 对应的 32 位寄存器值分别通过 RD1 和 RD2 输出
3	写入寄存器	WE=1 时，WD 的值写入 A3 所指的寄存器

5、ALU

（1）端口说明

端口名	方向	描述
A[31:0]	I	第 1 个 32 位操作数
B[31:0]	I	第 2 个 32 位操作数
C[31:0]	O	32 位计算结果
ALUOp[1:0]	I	ALU 功能选择
zero	O	A 和 B 相等比较结果

（2）功能定义

序号	功能名称	功能描述
1	ALU 数学逻辑计算	ALUOp=00，C=A+B ALUOp=01，C=A-B ALUOp=10，C=A|B ALUOp=11，保留
2	相等比较	zero=1，A=B zero=0，A≠B

6、DM

（1）端口说明

端口名	方向	描述
addr[31:0]	I	地址输入

WD[31:0]	I	写入数据
RD[31:0]	O	读出数据
WE	I	写入使能
clk	I	时钟信号
reset	I	异步复位信号

（2）功能定义

序号	功能名称	功能描述
1	复位	RST=1 时，在时钟上升沿 RAM 复位为零
2	读出数据	将 addr 所表示的内存地址中的值读出
3	写入数据	WE=1 时，将 WD 的值写入 addr 所表示的内存地址中

7、EXT

（1）端口说明

端口名	方向	描述
Imm[15:0]	I	16 位立即数输入
out[31:0]	O	32 位扩展结果输出
EXTOp[1:0]	I	扩展功能信号

（2）功能定义

序号	功能名称	功能描述
1	扩展立即数	EXTOp=00，进行零扩展 EXTOp=01，进行符号扩展 EXTOp=10，进行 LUI 扩展 EXTOp=11，保留

8、CMP

（1）端口定义

端口名	方向	描述

A[31:0]	I	32 位数据输入
B[31:0]	I	32 位数据输入
equal	O	相等信号输出

（2）功能定义

序号	功能名称	功能描述
1	相等比较	A==B, equal = 1 A!=B, equal = 0

9、multdiv

（1）端口定义

端口名	方向	描述
A[31:0]	I	32 位数据输入
B[31:0]	I	32 位数据输入
HiLoOp[2:0]	I	乘除法运算信号
C[31:0]	O	32 位数据输出
busy	O	模块忙信号

（2）功能定义

序号	功能名称	功能描述
1	multdiv 运算	`MULT_OP: 乘法运算 `MULTU_OP: 无符号乘法运算 `DIV_OP: 除法运算 `DIVU_OP: 无符号除法运算
2	HI、LO 读取	`MFHI_OP: HI 读取 `MFLO_OP: LO 读取
3	HI、LO 写入	`MTHI_OP: HI 写入 `MTLO_OP: LO 写入

10、DM_DECODE

（1）端口定义

端口名	方向	描述
Instr[31:0]	I	32 位指令输入
byte[1:0]	I	2 位地址数据输入
BE[3:0]	O	字节写使能输出

（2）功能定义

序号	功能名称	功能描述
1	DM 字节使能译码	sb: byte==00 ? BE=0001 byte==01 ? BE=0010 byte==10 ? BE=0100 byte==11 ? BE=1000 sh: byte==0X ? BE=0011 byte==1X ? BE=1100 sw: BE=1111

11、DM_EXT

（1）端口定义

端口名	方向	描述
Din[31:0]	I	32 位数据输入
DMEXTOp[2:0]	I	DM 输出扩展选择运算
byte[1:0]	I	字节输入
Dout[31:0]	O	32 位数据输出

（2）功能定义

序号	功能名称	功能描述
1	扩展选择运算	`LB_OP, `LBU_OP, `LH_OP, `LHU_OP, `LW_OP

12、Controller

（见第二部分：控制器设计）

三、控制器设计

1、端口定义及功能说明

端口名	方向	描述
opcode[5:0]	I	Instr[31:26]
funct[5:0]	I	Instr[5:0]
equal	I	相等比较信号
greater	I	大于比较信号
less	I	小于比较信号
DMWr	O	DM 写使能信号
ALUOp[3:0]	O	ALU 运算信号
HiLoOp[2:0]	O	MultDiv 运算信号
ASel	O	ALU 的 B 端 MUX 选择信号： 0：寄存器 RD2 值 1：shamt 数据
BSel	O	ALU 的 B 端 MUX 选择信号： 0：寄存器 RD2 值 1：EXT 扩展数据
EXTOp[1:0]	O	EXT 扩展信号
RFWr	O	GRF 写使能信号
WDSel[1:0]	O	寄存器的 WD 端 MUX 选择信号： 00：ALU 运算结果 01：DM 输出数据 10：PC + 4 11：保留

WRSel[1:0]	O	寄存器的 A3 端 MUX 选择信号： 00：Instr[20:16] 01：Instr[15:11] 10：0x1f 11：0x00
NPCOp[1:0]	O	IFU 中 NPC 运算信号

2、真值表

Instr	addu	subu	ori	lw	sw	beq	lui	j	jal	jr
Op	000000	000000	001101	100011	101011	000100	001111	000010	000011	000000
Func	100001	100011	NA	NA	NA	NA	NA	NA	NA	001000
DMWr	0	0	0	0	1	0	0	0	0	0
ALUOp	ADDU	SUBU	OR	ADDU	ADDU	XX	ADDU	XX	XX	XX
BSel	0	0	1	1	1	X	1	X	X	X
EXTOp	XX	XX	00	01	01	XX	10	XX	XX	XX
RFWr	1	1	1	1	0	0	1	0	1	0
WDSel	00	00	00	01	XX	XX	00	XX	10	XX
WRSel	01	01	00	00	XX	XX	00	XX	10	XX
NPCOp	00	00	00	00	00	0 Zero	00	10	10	11
Instr	bgez	bgtz	blez	bltz	bne	jalr	slt	slti	sltiu	sltu
Op	000001	000111	000110	000001	000101	000000	000000	001010	001011	000000
Func	NA	NA	NA	NA	NA	001001	101010	NA	NA	101011
DMWr	0	0	0	0	0	0	0	0	0	0
ALUOp	XX	XX	XX	XX	XX	XX	SLT	SLT	SLTU	SLTU
BSel	X	X	X	X	X	X	0	1	1	0
EXTOp	XX	XX	XX	XX	XX	XX	XX	01	01	XX
RFWr	0	0	0	0	0	1	1	1	1	1
WDSel	XX	XX	XX	XX	XX	XX	00	00	00	00
WRSel	XX	XX	XX	XX	XX	XX	01	00	00	01

NPCOp	0 Zero	0 Zero	0 Zero	0 Zero	0 Zero	11	00	00	00	00
Instr	addi	addiu	andi	xori	sll	sllv	sra	srav	srl	srlv
Op	001000	001001	001100	001110	000000	000000	000000	000000	000000	000000
Func	100001	100011	NA	NA	000000	000100	000011	000111	000010	000110
DMWr	0	0	0	0	0	0	0	0	0	0
ALUOp	ADD	ADDU	AND	XOR	SLL	SLL	SRA	SRA	SRL	SRL
BSel	1	1	1	1	0	0	0	0	0	0
EXTOp	01	01	00	00	XX	XX	XX	XX	XX	XX
RFWr	1	1	1	1	1	1	1	1	1	1
WDSel	00	00	00	00	00	00	00	00	00	00
WRSel	00	00	00	00	01	01	01	01	01	01
NPCOp	00	00	00	00	00	00	00	00	00	00
Instr	lh	lhu	lb	lbu	sh	sb	and	nor	or	xor
Op	100001	100101	100000	100100	101001	101000	000000	000000	000000	000000
Func	NA	NA	NA	NA	NA	NA	100100	100111	100101	100110
DMWr	0	0	0	0	1	1	0	0	0	0
ALUOp	ADDU	ADDU	ADDU	ADDU	ADDU	ADDU	AND	NOR	OR	XOR
BSel	1	1	1	1	1	1	0	0	0	0
EXTOp	01	01	01	01	01	01	XX	XX	XX	XX
RFWr	1	1	1	1	0	0	1	1	1	1
WDSel	01	01	01	01	XX	XX	00	00	00	00
WRSel	00	00	00	00	XX	XX	01	01	01	01
NPCOp	00	00	00	00	00	00	00	00	00	00
Instr	add	sub	mult	multu	div	divu	mfhi	mflo	mthi	mtlo
Op	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
Func	100000	100010	011000	011001	011010	011011	010000	101010	010001	010011
DMWr	0	0	0	0	0	0	0	0	0	0
ALUOp	ADD	SUB	MULT	MULTU	DIV	DIVU	MFHI	MFLO	MTHI	MTLO
BSel	0	0	0	0	0	0	0	0	0	0

EXTOp	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
RFWr	1	1	1	1	1	1	1	1	1	1
WDSel	00	00	00	00	00	00	00	00	00	00
WRSel	01	01	01	01	01	01	01	01	01	01
NPCOp	00	00	00	00	00	00	00	00	00	00

3、暂停控制设计表

				ID/EX 级				EX/MEM 级				
			指令类型	Calc_R	Calc_I	Load	JAL	Calc_R	Calc_I	Load	JAL	busy
			目标寄存器	rd	rt	rt	$31	rd	rt	rt	$31	
			Tnew	1	1	2	0	0	0	1	0	
指令类型	源寄存器	Tuse										
Calc_R	rs/rt	1				Stall						
Calc_I	rs	1				Stall						
Load	rs	1				Stall						
Beq	rs/rt	0		Stall	Stall	Stall				Stall		
Store	rs	1				Stall						
	rt	2										
JR	rs	0		Stall	Stall	Stall				Stall		
MDuse												stall

4、转发控制设计表

流水级	源寄存器	涉及指令	转发 MUX	控制信号	输入 0
IF/ID	rs	branch, jreg	MUX_ForwardRS_ID	ForwardRS_ID	GRF.RD1
	rt	branch	MUX_ForwardRT_ID	ForwardRT_ID	GRF.RD2
ID/EX	rs	calc_r, calc_i,load,store	MUX_ForwardRS_EX	ForwardRS_EX	RegData1_ID_EX
	rt	calc_r	MUX_ForwardRT_EX	ForwardRT_EX	RegData2_ID_EX
EX/MEM	rt	store	MUX_ForwardRT_MEM	ForwardRT_MEM	DM_WD_EX_MEM

			ID/EX 级			
		指令类型	Calc_R	Calc_I	Load	JAL
		目标寄存器	rd	rt	rt	$31
		Tnew	1	1	2	0
流水级	源寄存器					
IF/ID	rs		/	/	/	PC8_EX
	rt		/	/	/	PC8_EX

ID/EX	rs		/	/	/	/
	rt		/	/	/	/
EX/MEM	rt		/	/	/	/

		EX/MEM 级			
		Calc_R	Calc_I	Load	JAL
		rd	rt	rt	$31
		0	0	1	0
流水级	源寄存器				
IF/ID	rs	ALUout_MEM	ALUout_MEM	/	PC8_MEM
	rt	ALUout_MEM	ALUout_MEM	/	PC8_MEM
ID/EX	rs	ALUout_MEM	ALUout_MEM	/	PC8_MEM
	rt	ALUout_MEM	ALUout_MEM	/	PC8_MEM
EX/MEM	rt	/	/	/	/

		MEM/WB 级			
		Calc_R	Calc_I	Load	JAL
		rd	rt	rt	$31
		0	0	0	0
流水级	源寄存器				
IF/ID	rs	/	/	/	/
	rt	/	/	/	/
ID/EX	rs	RegWriteData_WB	RegWriteData_WB	RegWriteData_WB	RegWriteData_WB
	rt	RegWriteData_WB	RegWriteData_WB	RegWriteData_WB	RegWriteData_WB
EX/MEM	rt	RegWriteData_WB	RegWriteData_WB	RegWriteData_WB	RegWriteData_WB

四、测试模块

1、测试 A

测试程序：

```
li $s0, 0xfedcba87
li $s1, 0x6543210f
sw $s0, 0($0)
li $t0, 8
sh $s1, -4($t0)
sb $s1, -2($t0)
sb $s1, -1($t0)
lw $v0, -4($t0)
addiu $t0, $t0, -4
```

```
lh $v1, -2($t0)
lhu $v1, -2($t0)
lb $v1, -3($t0)
lbu $v1, -4($t0)
```

期望结果：

```
@00003000: $ 1 <= fedc0000
@00003004: $16 <= fedcba87
@00003008: $ 1 <= 65430000
@0000300c: $17 <= 6543210f
@00003010: *00000000 <= fedcba87
@00003014: $ 8 <= 00000008
@00003018: *00000004 <= 0000210f
@0000301c: *00000004 <= 000f210f
@00003020: *00000004 <= 0f0f210f
@00003024: $ 2 <= 0f0f210f
@00003028: $ 8 <= 00000004
@0000302c: $ 3 <= ffffedc
@00003030: $ 3 <= 0000fedc
@00003034: $ 3 <= ffffffba
@00003038: $ 3 <= 00000087
```

2、测试 B

测试程序：

```
li $s0, 0xcccc3333
li $s1, 0x88884444
mult $s0, $s1
mfhi $t0
```

```
mflo $t1
multu $s0, $s1
mfhi $t0
mflo $t1
div $s0, $s1
addiu $v0, $v0, 1
addiu $v0, $v0, 1
addiu $v0, $v0, 1
mfhi $t0
mflo $t1
divu $s0, $s1
jal go
sll $s0, $s0, 16
go:
slt $a0, $s0, $s1
mfhi $t0
mflo $t1
mthi $s0
mfhi $v0
mtlo $s1
mflo $v1
```

期望结果：

```
@00003000: $ 1 <= cccc0000
@00003004: $16 <= cccc3333
@00003008: $ 1 <= 88880000
@0000300c: $17 <= 88884444
@00003014: $ 8 <= 17e506d3
@00003018: $ 9 <= eeef258c
@00003020: $ 8 <= 6d397e4a
@00003024: $ 9 <= eeef258c
```

```
@0000302c: $ 2 <= 00000001
@00003030: $ 2 <= 00000002
@00003034: $ 2 <= 00000003
@00003038: $ 8 <= cccc3333
@0000303c: $ 9 <= 00000000
@00003044: $31 <= 0000304c
@00003048: $16 <= 33330000
@0000304c: $ 4 <= 00000000
@00003050: $ 8 <= 4443eeef
@00003054: $ 9 <= 00000001
@0000305c: $ 2 <= 33330000
@00003064: $ 3 <= 88884444
```

3、测试 C

测试程序：

```
li $s0, 0xfedcba87
sll $v0, $s0, 16
srl $v0, $s0, 16
sra $v0, $s0, 16
li $s1, 30
sllv $v1, $s0, $s1
srlv $v1, $s0, $s1
srav $v1, $s0, $s1
li $t0, 1
li $t1, -1
slt $a0, $t0, $t1
sltu $a0, $t0, $t1
slti $a0, $t0, -1
slti $a0, $t1, 1
```

```
sltiu $a0, $t0, -1
sltiu $a0, $t1, 1
andi $a1, $s0, 0xfefe
xori $a1, $s0, 0xfefe
```

期望结果：

```
@@00003000: $ 1 <= fedc0000
@00003004: $16 <= fedcba87
@00003008: $ 2 <= ba870000
@0000300c: $ 2 <= 0000fedc
@00003010: $ 2 <= ffffedc
@00003014: $17 <= 0000001e
@00003018: $ 3 <= c0000000
@0000301c: $ 3 <= 00000003
@00003020: $ 3 <= ffffffff
@00003024: $ 8 <= 00000001
@00003028: $ 9 <= ffffffff
@0000302c: $ 4 <= 00000000
@00003030: $ 4 <= 00000001
@00003034: $ 4 <= 00000000
@00003038: $ 4 <= 00000001
@0000303c: $ 4 <= 00000001
@00003040: $ 4 <= 00000000
@00003044: $ 5 <= 0000ba86
@00003048: $ 5 <= fedc4479
```

4、测试 D（Calc_R）

编号	测试类型	测试序列	预先赋值	期望结果
1	R-M-RS	subu $2, $3, $4	$3 = 5	$2 <= 3

		addu $1, $2, $4	$4 = 2	$1 <= 1
2	R-M-RT	subu $2, $3, $4 addu $1, $3, $2	$3 = 5 $4 = 2	$2 <= 3 $1 <= 2

3	R-W-RS	subu $2, $3, $4
		nop
		addu $1, $2, $4
			$3 = 5
			$4 = 2
				$2 <= 3
				$1 <= 1

4	R-W-RT	subu $2, $3, $4
		nop
		addu $1, $3, $2
			$3 = 5
			$4 = 2
				$2 <= 3
				$1 <= 2

5	I-M-RS	ori $2, $3, 8 addu $1, $2, $3	$3 = 3	$2 <= 11 $1 <= 14
6	I-M-RT	ori $2, $3, 8 addu $1, $3, $2	$3 = 3	$2 <= 11 $1 <= 14
7	I-W-RS	ori $2, $3, 8 nop addu $1, $2, $3	$3 = 3	$2 <= 11 $1 <= 14
8	I-W-RT	ori $2, $3, 8 nop addu $1, $3, $2	$3 = 3	$2 <= 11 $1 <= 14
9	LW-M-RS	lw $2, 0($3) addu $1, $2, $3	$3 = 4 *4 = 10	$2 <= 10 stall $1 <= 14
10	LW-M-RT	lw $2, 0($3) addu $1, $3, $2	$3 = 4 *4 = 10	$2 <= 10 stall $1 <= 14

11	LW-W-RS	lw $2, 0($3)
		nop
		addu $1, $2, $3
			$3 = 4
			*4 = 10
				$2 <= 10
				$1 <= 14

12	LW-W-RT	lw $2, 0($3)	$3 = 4	$2 <= 10

		nop addu $1, $3, $2	*4 = 10	$1 <= 14
13	JAL-M-RS	jal go addu $1, $31, $0 go:…		$31 <= PC(jal) + 8 $1 <= PC(jal) + 8
14	JAL-M-RT	jal go addu $1, $0, $31 go:…		$31 <= PC(jal) + 8 $1 <= PC(jal) + 8
15	JAL-W-RS	jal go nop go: addu $1, $31, $0		$31 <= PC(jal) + 8 $1 <= PC(jal) + 8
16	JAL-W-RT	jal go nop go: addu $1, $0, $31		$31 <= PC(jal) + 8 $1 <= PC(jal) + 8

5、测试 E(Calc_I)

编号	测试类型	测试序列	预先赋值	期望结果
1	R-M-RS	subu $2, $3, $4 ori $1, $2, 1	$3 = 8 $4 = 2	$2 <= 6 $1 <= 7
2	R-W-RS	subu $2, $3, $4 nop ori $1, $2, 1	$3 = 8 $4 = 2	$2 <= 6 $1 <= 7
3	I-M-RS	ori $2, $3, 8 ori $1, $2, 1	$3 = 2	$2 <= 10 $1 <= 11
4	I-W-RS	ori $2, $3, 8 nop ori $1, $2, 1	$3 = 2	$2 <= 10 $1 <= 11

5	LW-M-RS	lw $2, 0($3) ori $1, $2, 1	$3 = 4 *4 = 10	$2 <= 10 stall $1 <= 11
6	LW-W-RS	lw $2, 0($3) nop ori $1, $2, 1	$3 = 4 *4 = 10	$2 <= 10 $1 <= 11
7	JAL-M-RS	jal go ori $1, $31, 1 go:…		$31 <= PC(jal) + 8 $1 <= PC(jal) + 8 + 1
8	JAL-W-RS	jal go nop go: ori $1, $31, 1		$31 <= PC(jal) + 8 $1 <= PC(jal) + 8 + 1

6、测试 F (Load)

编号	测试类型	测试序列	预先赋值	期望结果
1	R-M-RS	subu $2, $3, $4 lw $1, 0($2)	$3 = 8 $4 = 4 *4 = 127	$2 <= 4 $1 <= 127

1	R-M-RS	subu $2, $3, $4
		nop
		lw $1, 0($2)
			$3 = 8
			$4 = 4
			*4 = 127
				$2 <= 4
				$1 <= 127

3	I-M-RS	ori $2, $3, 8 lw $1, 0($2)	$3 = 0 *8 = 127	$2 <= 8 $1 <= 127
4	I-W-RS	ori $2, $3, 8 nop lw $1, 0($2)	$3 = 0 *8 = 127	$2 <= 8 $1 <= 127

5	LW-M-RS	lw $2, 0($3) lw $1, 0($2)	$3 = 4 *4 = 10	$2 <= 10 stall $1 <= 11

6	LW-W-RS	lw $2, 0($3)
		nop
		lw $1, 0($2)
			$3 = 4
			*4 = 10
				$2 <= 10
				$1 <= 11

7	JAL-M-RS	jal go lw $1, 0($31) go:…	*[PC(jal) + 8] = 333	$31 <= PC(jal) + 8 $1 <= 333
8	JAL-W-RS	jal go nop go: lw $1, 0($2)	*[PC(jal) + 8] = 333	$31 <= PC(jal) + 8 $1 <= 333

7、测试 G (Store)

编号	测试类型	测试序列	预先赋值	期望结果
1	R-M-RS	subu $2, $3, $4 sw $4, -4($2)	$3 = 10 $4 = 2	$2 <= 8 *4 <= 2
2	R-M-RT	subu $2, $3, $4 sw $2, -4($4)	$3 = 6 $4 = 4	$2 <= 2 *0 <= 2

3	R-W-RS	subu $2, $3, $4
		nop
		sw $4, -4($2)
			$3 = 10
			$4 = 2
				$2 <= 8
				*4 <= 2

4	R-W-RT	subu $2, $3, $4
		nop
		sw $2, -4($4)
			$3 = 6
			$4 = 4
				$2 <= 2
				*0 <= 2

5	I-M-RS	ori $2, $3, 8 sw $4, -4($2)	$3 = 4 $4 = 11	$2 <= 12 *8 <= 11
6	I-M-RT	ori $2, $3, 8	$3 = 3	$2 <= 11

		sw $2, -4($4)	$4 = 24	*20 <= 11
7	I-W-RS	ori $2, $3, 8 nop sw $4, -4($2)	$3 = 4 $4 = 11	$2 <= 12 *8 <= 11
8	I-W-RT	ori $2, $3, 8 nop sw $2, -4($4)	$3 = 3 $4 = 24	$2 <= 11 *20 <= 11
9	LW-M-RS	lw $2, 0($3) sw $4, -4($2)	$3 = 4 *4 = 12 $4 = 11	$2 <= 12 stall *8 <= 11
10	LW-M-RT	lw $2, 0($3) sw $2, -4($4)	$3 = 4 *4 = 10 $4 = 8	$2 <= 10 *4 <= 10

11	LW-W-RS	lw $2, 0($3)
		nop
		sw $4, -4($2)
			$3 = 4
			*4 = 12
			$4 = 11
				$2 <= 12
				*8 <= 11

12	LW-W-RT	lw $2, 0($3)
		nop
		sw $2, -4($4)
			$3 = 4
			*4 = 10
			$4 = 8
				$2 <= 10
				*4 <= 10

13	JAL-M-RS	jal go sw $1, 0($31) go:…	$1 = 0	$31 <= PC(jal) + 8 *[PC(jal) + 8] <= 0 (可写入.text)
14	JAL-M-RT	jal go sw $31, 0($0) go:…		$31 <= PC(jal) + 8 *0 <= PC(jal) + 8
15	JAL-W-RS	jal go nop go: sw $1, 0($31)	$1 = 0	$31 <= PC(jal) + 8 *[PC(jal) + 8] <= 0 (可写入.text)

16	JAL-W-RT	jal go nop go: sw $31, 0($0)		$31 <= PC(jal) + 8 *0 <= PC(jal) + 8

8、测试 H (Branch)

编号	测试类型	测试序列	预先赋值	期望结果
1	R-E-RS	subu $2, $3, $4 beq $2, $5, go nop lui $1, 0xffff go: lui $1, 0x1111	$3 = 10 $4 = 2 $5 = 8	$2 <= 8 stall jump to go $1 <= 0x11110000
2	R-E-RT	subu $2, $3, $4 beq $5, $2, go nop lui $1, 0xffff go: lui $1, 0x1111	$3 = 10 $4 = 2 $5 = 8	$2 <= 8 stall jump to go $1 <= 0x11110000
3	R-M-RS	subu $2, $3, $4 nop beq $2, $5, go nop lui $1, 0xffff go: lui $1, 0x1111	$3 = 10 $4 = 2 $5 = 8	$2 <= 8 jump to go $1 <= 0x11110000
4	R-M-RT	subu $2, $3, $4 nop beq $5, $2, go	$3 = 10 $4 = 2 $5 = 8	$2 <= 8 jump to go $1 <= 0x11110000

		nop lui $1, 0xffff go: lui $1, 0x1111		
5	I-E-RS	ori $2, $0, 8 beq $2, $5, go nop lui $1, 0xffff go: lui $1, 0x1111	$5 = 8	$2 <= 8 stall jump to go $1 <= 0x11110000
6	I-E-RT	ori $2, $0, 8 beq $5, $2, go nop lui $1, 0xffff go: lui $1, 0x1111	$5 = 8	$2 <= 8 stall jump to go $1 <= 0x11110000
7	I-M-RS	ori $2, $0, 8 nop beq $2, $5, go nop lui $1, 0xffff go: lui $1, 0x1111	$5 = 8	$2 <= 8 jump to go $1 <= 0x11110000
8	I-M-RT	ori $2, $0, 8 nop beq $5, $2, go nop lui $1, 0xffff go:	$5 = 8	$2 <= 8 jump to go $1 <= 0x11110000

		lui $1, 0x1111		
9	LW-E-RS	lw $2, 0($3) beq $2, $5, go nop lui $1, 0xffff go: lui $1, 0x1111	$3 = 4 *4 = 12 $5 = 12	$2 <= 12 stall stall jump to go $1 <= 0x11110000
10	LW-E-RT	lw $2, 0($3) beq $5, $2, go nop lui $1, 0xffff go: lui $1, 0x1111	$3 = 4 *4 = 12 $5 = 12	$2 <= 12 stall stall jump to go $1 <= 0x11110000
11	LW-M-RS	lw $2, 0($3) nop beq $2, $5, go nop lui $1, 0xffff go: lui $1, 0x1111	$3 = 4 *4 = 12 $5 = 12	$2 <= 12 stall jump to go $1 <= 0x11110000
12	LW-M-RT	lw $2, 0($3) nop beq $5, $2, go nop lui $1, 0xffff go: lui $1, 0x1111	$3 = 4 *4 = 12 $5 = 12	$2 <= 12 stall jump to go $1 <= 0x11110000
13	LW-W-RS	lw $2, 0($3) nop	$3 = 4 *4 = 12	$2 <= 12 jump to go

		nop beq $2, $5, go nop lui $1, 0xffff go: lui $1, 0x1111	$5 = 12	$1 <= 0x11110000
14	LW-W-RT	lw $2, 0($3) nop nop beq $5, $2, go nop lui $1, 0xffff go: lui $1, 0x1111	$3 = 4 *4 = 12 $5 = 12	$2 <= 12 jump to go $1 <= 0x11110000
15	JAL-E-RS	未定义行为		
16	JAL-E-RT	未定义行为		
17	JAL-M-RS	jal pass nop pass: beq $ra, $2, go nop lui $1, 0xffff go: lui $1, 0x1111	$2 = 2	$2 <= 12 stall NOT jump to go $1 <= 0xffff0000 $1 <= 0x11110000
18	JAL-M-RT	jal pass nop pass: beq $2, $ra, go nop	$2 = 2	$2 <= 12 stall NOT jump to go $1 <= 0xffff0000 $1 <= 0x11110000

		lui $1, 0xffff go: lui $1, 0x1111		
19	JAL-W-RS	jal pass nop pass: nop beq $ra, $2, go nop lui $1, 0xffff go: lui $1, 0x1111	$2 = 2	$2 <= 12 stall NOT jump to go $1 <= 0xffff0000 $1 <= 0x11110000
20	JAL-W-RT	jal pass nop pass: nop beq $2, $ra, go nop lui $1, 0xffff go: lui $1, 0x1111	$2 = 2	$2 <= 12 stall NOT jump to go $1 <= 0xffff0000 $1 <= 0x11110000

9、测试 I （JumpWithReg）

编号	测试类型	测试序列	预先赋值	期望结果
1	R-E-RS	addu $2, $3, $4 jr $2	$3 = 0x3000 $4 = 8	$2 <= 0x000030008 stall jump to 0x000030008
2	R-M-RS	addu $2, $3, $4 nop	$3 = 0x3000 $4 = 8	$2 <= 0x000030008 jump to 0x000030008

		jr $2		
3	R-W-RS	addu $2, $3, $4 nop nop jr $2	$3 = 0x3000 $4 = 8	$2 <= 0x000030008 jump to 0x000030008
4	I-E-RS	ori $2, $0, 0x3008 jr $2		$2 <= 0x000030008 stall jump to 0x000030008
5	I-M-RS	ori $2, $0, 0x3008 nop jr $2		$2 <= 0x000030008 jump to 0x000030008
6	I-W-RS	ori $2, $0, 0x3008 nop nop jr $2		$2 <= 0x000030008 jump to 0x000030008
7	LW-E-RS	lw $2, 0($0) jr $2	*0 = 0x3008	$2 <= 0x000030008 stall stall jump to 0x000030008
8	LW=M-RS	lw $2, 0($0) nop jr $2	*0 = 0x3008	$2 <= 0x000030008 stall jump to 0x000030008
9	LW-W-RS	lw $2, 0($0) nop nop jr $2	*0 = 0x3008	$2 <= 0x000030008 jump to 0x000030008
10	JAL-E-RS	未定义行为		
11	JAL-M-RS	jal go nop		$ra <= PC(jal)+8 forever jump

		go: jr $ra		
12	JAL-W-RS	jal go nop go: nop jr $ra		$ra <= PC(jal)+8 forever jump

五、思考题

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

乘除法的计算一般需要耗费较多的时间，并且，它的逻辑运算单元也比较庞大，将 ALU 与其分离开来，有利于对芯片模块化的设计，让两者产生更少的干扰，也便于乘除法模块对流水线暂停的控制。

将 HI、LO 独立出来，是对乘除指令与访存 HI、LO 指令需求的体现，同时，也让乘除法模块更具有整体性，是一种高内聚的设计方法。

2、参照你对延迟槽的理解，试解释“乘除槽”。

分支延迟槽是针对跳转指令在执行跳转中，为避免控制冒险而浪费一个时钟周期，的一种解决方案，将跳转指令前的不影响程序功能的指令放入延迟槽，以此节省下来控制冲突带来的浪费。

乘除法因其所需大量的计算时间，因此，允许其后不需要乘除法模块的指令继续通过流水线，而阻塞需要乘除法模块的 mult、mflo 等指令，以此优化乘除法的延迟。

3、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）

在对存储单元小于一个字长的数据结构，或存储地址为非按字对齐的数据进行访问时，按字节访问更有优势。

4、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？

在设计我的 CPU 中，我的风格应该是“侦测者”型。

将指令根据是否产生新的数据、产生新数据的位置、需求数据的个数和获取位置，将它们进行分类和抽象。这样，在扩增指令时，只需考虑对应指令的相应类型，就可以精确实现转发和暂停的控制。并且，采用分布式译码，对访问寄存器的地址和数据进行提前计算，模块化设计，都是对 CPU 设计的规范手段。

5、在本实验中你遇到了哪些不同指令类型组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？

为手动构造特殊的指令进行冲突测试，根据暂停和转发的表格，对指令顺序进行有计划的排列，如上所写测试。