思考题

L0. T2. 1

30 位 PC：接指令存储器时方便，无需使用分线器取前 30 位;在执行需要构造下一指令地址的指令时（如 j 指令）方便，无需再向左移位。

32 位 PC：在执行需要使用 PC+4 的值的指令时（如 jal）方便，不需要重新补齐两个零

两者各自的优点分别是对方的缺点

L0. T2. 2

合理，理由如下：

1. DM 必须用 RAM 因为 logisim 中只有 RAM 可以设为 separate load and store 模式，符合本设计的需求；用寄存器来做的话需要手画大量寄存器，代价太高
2. IM 用 ROM 做更合适，因为 ROM 无法通过端口写入，防止代码段数据被篡改。
3. 寄存器堆必须用寄存器来手动构建，否则实现两路输出十分困难.

L0. T3. 1

1. 按照题目中给出的真值表，若记 op 自低到高分别为 o0,o1,o2,o3,o4,o5

func 自低到高为 f0,f1,f2,f3,f4,f5.则

RegDst= !o1 !o2 !o3 !o4 !o5 !o0(f5 !f4 !f3 !f2 !f1 !f0 + f5 !f4 !f3 !f2f1 !f0)

ALUSrc= (!o5 !o4 o3 o2 !o1 o0) + (o5 !o4 !o3 !o2 o1 o0) + (o5 !o4 o3 !o2 o1 o0)

Memtoreg= (o5 !o4 !o3 !o2 o1 o0)

RegWrite= !o1 !o2 !o3 !o4 !o5 !o0(f5 !f4 !f3 !f2 !f1 !f0 + f5 !f4 !f3 !f2f1 !f0)

+ !o5 !o4 o3 o2 !o1 o0 + o5 !o4 !o3 !o2 o1 o0

npc_sel= (!o5 !o4 !o3 o2 !o1 !o0)

extop= (o5 !o4 !o3 !o2 o1 o0) + (o5 !o4 o3 !o2 o1 o0)

L0.T3.2

RegDst= !o1 !o2 !o3 !o4 !o5 !o0f5 !f4 !f3 !f2 !f0

AluSrc= !o5 !o4 o3 o2 !o1 o0 + o5 !o4 !o2 o1 o0

Memtoreg = o5 !o4 !o2 o1 o0

Regwrite=!o1 !o2 !o3 !o4 !o5 !o0(f5 !f4 !f3 !f2 !f1 !f0 + f5 !f4 !f3 !f2f1 !f0)+ !o5 !o4 o3 o2 !o1 o0 + o5 !o4 !o3 !o2 o1 o0

npc_sel= (!o5 !o4 !o3 o2 !o1 !o0)

extop= (o5 !o4 o3 !o2 o1 o0)

L0. T3. 1

不需要 有两种解释

1. 在网站图中所示的方案里未要求实现 sll 指令，所有的写使能信号都给出了明确的 01 值，且 1 为写入，在这种情况下，我们可以将 nop 时的所有信号置为 0，由于所有写使能均为 0，不会造成任何影响。而这样的信号在在由最小项构成的表达式中不会有任何体现。因此没有影响
2. Nop 指令可以被解析为 shamt 字段为 00000 的 sll 指令，向左移 0 位不会造成任何改变。因此就算实现 sll 指令也不需要将 nop 加入真值表。

L0. T4. 1

如果片选信号是为了确定所要访问的内存地址在哪一部分里，那就对所需地址的高位进行比大小，用比较结果作为片选信号就好了。比如说，如

果数据段从0x00003000开始，那就将前20位提出来与0x00003做比较，比较结果作为复选器的选择信号，例如，若比较结果为大于，选择信号为1，复选器选择 DM 存储器。

L0. T4. 2

1. 形式验证的优点：
    a. 形式验证 100%可靠，覆盖了所有可能，不存在漏掉某些情况的可能
    b. 形式验证比编写程序验证快，可以缩短实现所用时间 1
2. 形式验证缺点:
    a. 形式验证只能比较你的实现和设计是否相符，对设计上出现固有缺陷导致的问题很难查出。

CPU 文档

一：功能模块规格

1. IFU

a.接口定义

接口名	类型	描述
clk	I	时钟信号
reset	I	异步复位信号
branch	I	是否选择跳转至 npc 接口数据指向的地址，1 为有效
npc[31:0]	I	跳转地址输入
Instr[31:0]	O	输出所取出的指令
pc4	O	输出 PC+4 的值

b.功能描述

序号	功能名称	功能描述
1	复位	当 clr 信号有效时 pc 被置为 0x0000000
2.	取指令	取出 pc 对应地址里的指令
3	生成下一指令地址	若 branch 有效，下一指令地址为 npc 否则为 pc+4

2. GPR

a. 端口定义

接口名	类型	描述
clk	I	时钟信号
reset	I	异步复位信号，1 为有效
we	I	写使能信号，1 为有效
A1[4:0]	I	RD1 端口输出的寄存器的编号
A2[4:0]	I	RD2 端口输出的寄存器的编号
A3[4:0]	I	要写入的寄存器的编号
WD	I	要写入选定寄存器的数据

RD1	O	输出 A1 端口选中寄存器的值
RD2	O	输出 A2 端口选中寄存器的值

b．功能描述

序号	功能名称	功能描述
1	复位	reset 信号有效时，所有寄存器的值置为 0
2	读寄存器	RD1,RD2 输出 A1,A2 选中寄存器的值
3	写寄存器	we 信号有效时 A3 选中的寄存器的值被写为 WD 的值

3. ALU

a. 接口定义

接口名	类型	描述
a1[31:0]	I	输入 alu 的第一个操作数
a2[31:0]	I	输入 alu 的第二个操作数
aluop[1:0]	I	alu 功能选择 00 为与运算，01 为加运算，10 为减运算，11 为或运算，
aluout[31:0]	O	输出运算结果
zero	O	若 a1==a2 则输出 1 否则为 0

b.功能描述

序号	功能名称	功能描述
1	与运算	输出 a1&a2
2	加运算	输出 a1+a2
3	减运算	输出 a1-a2
4	或运算	输出 a1|a2
5	判断是否相等	输出 a1==a2

4.DM

接口名	类型	描述
A[4:0]	I	DM 输入的地址
INPUT[31:0]	I	DM 要写入的数据

clk	I	时钟信号
clr	I	异步复位信号
str	I	写使能信号，1 为有效，将 A 端口选中的字写为 INPUT 中数据
D [31:0]	O	输出 A 端口选中的字的值

b. 功能描述

序号	功能名称	功能描述
1	复位	clr 信号有效时，所有主存的值置为 0
2	读主存	输出 A 端口选中的字的值
3	写主存	Str 有效时将 A 端口选中的字写为 INPUT 中数据

5.NPC

a.端口定义

接口名	类型	描述
npcop[1:0]	I	功能选择 00 依据 Imm 字段计算下一指令地址 01 依据指令后 26 位与 pc+4 高四位计算地址 02 输出 rd1 中的地址
pc4	I	输入 pc+4
extout	I	输入 imm 进行 16 位扩展后的信号
rd1	I	输入 gpr 中 a1 选定寄存器的值
intsr	I	输入当前正在执行的指令
npc	O	输出计算出的下一指令

b 功能描述

序号	功能名称	功能描述
1	实现立即数跳转	依据 Imm 字段计算下一指令地址
2	实现 j 类型跳转	依据指令后 26 位与 pc+4 高四位计算地址
3	实现寄存器跳转	输出 rd1 中的地址

6.EXT

a.端口定义

接口名	类型	描述
in[15:0]	I	输入 16 位待扩展立即数
out[31:0]	O	输出扩展后 32 位数据
extop[1:0]	I	功能选择 00 无符号扩展，01 有符号扩展，02Lui 扩展

b 功能描述

序号	功能名称	功能描述
1	无符号扩展	高位补 16 个 0
2	有符号扩展	高位补 16 个符号位
3	Lui 扩展	低位补 16 个 0

7.shift

a.端口定义

接口名	类型	描述
shamt[4:0]	I	输入 shamt 字段移位位数
data[31:0]	I	输入待移位数据
shiftop[1:0]	I	功能选择 00 逻辑左移 01 逻辑右移 10 算数右移
shiftdata[31:0]	O	输出移位后数据

b 功能描述

序号	功能名称	功能描述
1	逻辑左移	左移 shamt 位，低位补 0
2	逻辑右移	右移 shamt 位，高位补 0
3	算数右移	右移 shamt 位，高位补符号位

8.byte

a.端口定义

接口名	类型	描述
addr[1:0]	I	输入要写入地址的低两位
dmout[31:0]	I	输入 dm 中读取的字数据
data[31:0]	I	输入装有待写入字节的数据
byteout[31:0]	O	输出装有待输出字节的数据

fordmdata[31:0]	O	输出更改了指定字节，写回 dm 的数据

b 功能描述

序号	功能名称	功能描述
1	读指定字节	输出包含地址选定字节的数据
2	写指定字节	生成修改了指定字节，用来写回 dm 的数据

9.halfword

a.端口定义

接口名	类型	描述
addr[1:0]	I	输入要写入地址的低两位
dmout[31:0]	I	输入 dm 中读取的字数据
data[31:0]	I	输入装有待写入半字的数据
halfwoord[31:0]	O	输出装有待输出半字的数据
fordmdata[31:0]	O	输出更改了指定半子，写回 dm 的数据

b 功能描述

序号	功能名称	功能描述
1	读指定字节	输出包含地址选定半字的数据
2	写指定字节	生成修改了指定半字，用来写回 dm 的数据

二:控制器设计

1. 控制逻辑真值表：(未填写部分全部为 x)

	addu	subu	ori	Lw	Sw	Lui	Beq	J	Jal	Jr	sll
branch	0	0	0	0	0	0	1	1	1	1	0
regdst	00	00	01	01		01			10		00
regwrite	1	1	1	1	0	1	0	0	1	0	1
npcop	x						00	01	01	10	
alusrc	0	0	1	1	1						
aluop	01	10	11	01	01						
memtoreg	000	000	000	001		010			011		100

memwrite	0	0	0	0	1	0	0	0	0	0	0
extop			00	01	01	10					
shiftop											00
dmoutctrl	0	0	0	0	0	0	0	0	0	0	0
dmdatactrl	0	0	0	0	0	0	0	0	0	0	0
isbeq	0	0	0	0	0	0	1	0	0	0	0

续表

	Srl	Sra	Lhu	Sh	Lbu	Sb					
branch	0	0	0	0	0	0					
regdst	00	00	01		01						
regwrite	1	1	1	0	1	0					
npcop											
alusrc			1	1	1	1					
aluop			01	01	01	01					
memtoreg	100	100	001		001						
memwrite	0	0	0	1	0	1					
extop			01	01	01	01					
shiftop	01	10									
dmoutctrl	0	0	01	00	10	00					
dmdatactrl	0	0	00	01	00	10					
isbeq	0	0	0	0	0	0					

相关控制信号含义

控制信号	解释
branch	1 时有效，控制 IM 模块取用 npc 端口输入的指令地址
regdst	00：rd 域 01：rt 域 10：常数 31
regwrite	寄存器写使能信号
npcop	00：imm 扩展地址 01 低 26 位生成地址 10：rs 域寄存器生成地址
alusrc	0：rd2 域寄存器值 1：ext 扩展出的值
aluop	00 且 01 加 10 减 11 或

memtoreg	000alu 计算结果 001 主存返回值 010ext 返回值 010pc4 返回值 100 移位器返回值
memwrite	主存写使能信号
extop	00 无符号扩展 01 有符号扩展 10lui 扩展
shiftop	00 左移 01 逻辑右移 10 算数右移
dmoutctrl	00 主存输出结果 01 半字模块输出结果 10 字节模块输出结果
dmdatactrl	00rd2 寄存器输出 01 半字模块写回结果 10 字节模块写回结果
isbeq	是否是 beq 指令，1 时有效

2. 测试程序

测试程序甲：

```
lui $s0,0xffff
ori $s0,0xffff
ori $s1,$0,1
ori $s2,$0,2
addu $s3,$s0,$s2
subu $s4,$s2,$s1
beq $s1,$s2,tar1
 ori $a0,$0,1
tar1:
beq $s4,$s1,tar2
 ori $a1,$0,1
tar2:
lui $t0,0xabcd
ori $t0,0x1234
sw $t0,4($0)
lw $t1,4($0)
lhu $t2,6($0)
lbu $t3,7($0)
sw $t0,0($0)
```

```
    sh $t2,10($0)
  sb $t3,9($0)
```

预期结果：$a0=1 $t0=0xabcd1234 $t1=0xabcd1234 $t2=0000abcd $t3=000000ab $s0=-1(0xffffffff),$s1=1,$s2=2,$s3=1,$s4=1

测试程序乙：

```
    ori $a0,$0,24
    j tar1
    ori $s0,$0,1
    tar1:
    ori $s1,1
    jal tar2
    ori $s2,$0,1
    ori $s3,1
    tar2:
    addu $s3,$s3,$s3
    addu $ra,$ra,$a0
    jr $ra
    ori $s4,$0,1
    ori $s5,$0,1
```

预期结果：$a0=0x18,$s0=0,$s1=1,$s2=0,$s3=0 $s4=0 $s5=1 $ra=0x2c