2020

和 P3 差不多，仔细看 RTL。总的来说感觉最好课下好好练练 Verilog，课上写的时候感觉有点手生，或许是因为不如电路那么直观。但是在写复杂逻辑的时候不用连电路了确实很好。

- 第一题：`bsoal`

  感觉这个指令是全场最难（溜）。需要熟悉组合电路的 `always @(*)` 写法，或者直接用 `function`。

  当寄存器值有奇数个 1 的时候跳转。

$$
\begin{aligned}
&\text{if}\ \ \text{has\_odd\_one\_bits}(GRF[rs])\ \ \text{then}\\
&\qquad PC = PC + 4 + \text{sign\_ext}(offset||0^2)\\
&\qquad GRF[31] = PC + 4\\
&\text{else}\\
&\qquad PC = PC + 4
\end{aligned}
$$

- 第二题：`xor`

  就是需要加一个高老板 PDF 的 `xor`，居然出了原题，这应该是全场最简单的了。

- 第三题：`swrr`

  向右循环位移，有点类似 `sw`。注意好好看题！题目要求 `$display()` 要字对齐。

$$
\begin{aligned}
&Addr \leftarrow GPR[base] + \texttt{sign\_ext}(offset)\\
&temp \leftarrow Addr_{1..0}\\
&\text{if}\ \ temp == 0\ \ \text{then}\\
&\qquad mem_{addr} \leftarrow GRF[rt]\\
&\text{else}\\
&\qquad mem_{addr} \leftarrow GRF[rt]_{8*temp-1\cdots 0}||GRF[rt]_{31\cdots 8*temp}
\end{aligned}
$$

课下一定要模块化，课上不要慌！（我课上因为失误，第三题拖了好长时间）

最重要的是拿到指令先去和写过的找共同点，分析要用哪些部件。然后按照套路添加即可。

```
// 在ALU里加一个always-for
integer i = 0;
always @(*) begin
    for (i = 0; i < 32; i = i + 1) begin
        if (SrcA[i] == 1'b1) cnt1 = cnt1 + 1;
    end
end
// 在条件判断时，
if (cnt1 % 2) ; // operation
// or write like this
if (^SrcA == 1'b1) ; // operation
```

2021

P4_L1_rlb

编码：

31 26	25 21	20 16	15 0
rlb 101010	rs	rt	imm
6	5	5	16

格式：

```
rlb rs, rt, imm
```

描述：

```
指定位数取反
```

操作：

```
temp <- GPR[rs]imm-1...0
GPR[rt] <- ( GPR[rs]31...imm || ~temp )
```

实例：

```
rlb $t1, $t2, 5
```

其他：

```
保证 imm 的范围在 0 ~ 31 之间
```

注意：这个下标是变量不能直接写，需要用for循环

P4_L2_bnezalc

编码：

31 26	25 21	20 16	15 0
regimm 000001	rs	bnezalc 11001	offset
6	5	5	16

格式：

```
bnezalc rs, offset
```

描述：

```
若 GPR[rs] != 0 跳转并 Link
```

操作：

```
I:
    target_offset <- sign_extend(offset||00)
    condition <- GPR[rs] != 0
    If Condition:
        GPR[31] <- PC + 8
    endif
I+1:
    If Condition:
        PC ← PC + target_offset
    endif
```

实例：

```
bnezalc $t1, -1
```

其他：

```
不跳转则不Link，视为nop
```

注意：c 说明是若跳转才链接，al应该是unconditionally的

P4_L3_lwrr

编码：

31 26	25 21	20 16	15 0
lwrr 000011	base	rt	offset
6	5	5	16

格式：

```
lwrr rt, offset(base)
```

描述：

```
根据内存后两位，循环右移后存入对应寄存器中
```

操作：

addr <- GPR[base] + sign_extend(offset)
Paddr <- Memory[addr]
temp <- Paddr1...0
if temp != 0:
    Vaddr <- (Paddr(temp-1)...0 || Paddr31...temp)
else:
    Vaddr <- Paddr
GPR[rt] <- Vaddr

实例:

```
lwrr $t5, 1($0)
```

其他:

```
无
```

```
对load的数据直接操作即可
要写rt寄存器
另写一个output为Vaddr
在DM里建模组合逻辑就行
```

```
output reg [31:0] Vaddr;

wire temp[1:0];
wire Paddr[31:0];
integer i = 0;

assign Paddr = MemReadData;
assign temp = Paddr[1:0];

always @(*) begin
    case (temp)
    /*if (temp != 2'b00) begin
        for (i = 31; i >= 0; i = i - 1) begin
            if (i >= 32 - temp) Vaddr[i] = Paddr[temp + i - 32];
            else Vaddr[i] = Padd[];
        end
        case
    end*/
        2'b00 : begin
            Vaddr = Paddr;
        end
        2'b01 : begin
            Vaddr = {Paddr[0], Paddr[31:1]};
        end
        2'b10 : begin
            Vaddr = {Paddr[1:0], Paddr[31:2]};
        end
        2'b11 : begin
            Vaddr = {Paddr[2:0], Paddr[31:3]};
```

```
        end
    endcase
end
```

2022

第一次

L1_bal

GPR[rs]中如果1的个数不为零且被0的个数整除则bal

```
reg bal;

always @(*) begin
    bal = 0; // clear first
    integer cnt1 = 0;
    integer cnt0 = 0;
    integer i = 0;
    for (i = 0; i < 32; i = i + 1) begin
        if (SrcA[i] == 1'b1) cnt1 = cnt1 + 1;
    end
    cnt0 = 32 - cnt1;
    if (cnt1 % cnt0 == 0) bal = 1;
    else bal = 0;
end
```

L2

GPR[rt]和GPR[rs]如果后缀0个数相同则GPR[rd]置1 否则GPR[rd]置0

```
always @(*) begin
    cnt1 = 0;
    cnt2 = 0; // clear first
    for (i = 0; i < 32 && SrcA[i] == 1'b0; i = i + 1) begin
        cnt1 = cnt1 + 1;
    end
    for (i = 0; i < 32 && SrcB[i] == 1'b0; i = i + 1) begin
        cnt2 = cnt2 + 1;
    end
    if (cnt1 == cnt2) Result = 32'b1;
    else Result = 32'b0;
end
// RegDst = 1;
```

共第三次

P4_L1_RLB_2022

题目编号 937-982

编码	**31 26**	**25 21**	**20 16**	**15 0**
RLB 111111	rs	rt	immediate	
6	5	5	16	
格式	rlb rt, rs, immediate			
描述	将GPR[rs]低immediate位按位取反后，存入GPR[rt]			
操作	def :bitwise_not(x): 将x按位取反if imm = 0 :GPR[rt] ← GPR[rs]else :GPR[rt] ← GPR[rs]31...imm||bitwise_not(GPR[rs]imm-1...0)			
示例	rlb $t0, $t1, 16			
其他	数据保证immediate在[0,31]闭区间内			

P4_L5_BNEZALC_2022

题目编号 937-986

编码	**31 26**	**25 21**	**20 16**	**15 0**
regimm 000001	rs	BNEZALC 10011	offset	
6	5	5	16	
格式	bnezalc rs, offset			
描述	如果 GPR[rs]不等于0，则跳转到 label，并将 PC + 4 存入 31 号寄存器。			
操作	if (GPR[rs] ≠ 0) thenPC ← PC + 4 + sign_extend(offset||02)GPR[31] ← PC + 4elsePC ← PC + 4endif			
示例	bnezalc $t0, label			
其他	不考虑延迟槽			

P4_L6_LBOEZ_2022

题目编号 937-987

编码	**31 26**	**25 21**	**20 16**	**15 0**
LBOEZ 111110	base	rt	offset	
6	5	5	16	
格式	lboez rt, offset(base)			
描述	Addr ← GPR[base] + sign_extend(offset)，若memory[Addr]的对应**字节**数据的二进制1的个数等于0的个数，则将该**字节**符号扩展存入GPR[rt]，否则将0存入GPR[rt]。			
操作	Addr ← GPR[base] + sign_extend(offset)memword ← Memory[Addr]byte ← Addr1..0temp ← memword7+8*byte + memword7+8*byte-1 + … + memword8*byteif temp = 4 thenGPR[rt] ← sign_extend(memword7+8*byte..8*byte)elseGPR[rt] ← 0endif			
示例	lboez $t0, 1($t1)			
其他	注意：需要取出字中相应的字节，而非对整个字进行操作。			