实验报告（控制信号设计）

信号设计

我设计了以下9个控制信号：

1. Z：零标志信号，为ALU输出，对Pcsrc有影响。
2. Regrt：控制寄存器的写入端口。。
3. Se：控制扩展模块，包括零扩展与符号扩展。
4. Wreg：控制寄存器端的写使能信号。
5. Aluqb：控制ALU的Y端口的输入值。
6. Aluc：控制ALU的计算种类。
7. Wmem：控制数据存储器的写使能信号。
8. Pcsrc：控制目标指令地址。
9. Reg2reg：控制REHFILE更新值的来源。

之后我设计了一个表格辅助完成ControlUnit模块：

输入端口	**输入端口**	**输入端口**	**输入端口**	**输出端口**							
Op	Func	备注	Z	Regrt[1:0]	Se	Wreg	Aluqb	Aluc[1:0]	Wmem	Pcsrc[1:0]	Reg2reg[1:0]
000000	100001	addu	X	0	0	1	1	0	0	0	1
000000	100011	subu	X	0	0	1	1	1	0	0	1
001101	X	ori	X	1	0	1	0	11	0	0	1
100011	X	lw	X	1	1	1	0	0	0	0	0
101011	X	sw	X	1	1	0	0	0	1	0	1
000100	X	beq	0	1	1	0	1	1	0	0	1
000100	X	beq	1	1	1	0	1	1	0	10	1
001111	X	lui	X	1	0	1	0	10	0	0	1
000011	X	jal	X	10	0	1	1	0	0	11	10
000000	001000	jr	X	1	0	0	1	0	0	1	1
000010	X	j	X	1	0	0	1	0	0	11	1

模块实现

我利用2组合逻辑硬布线的方式实现ControlUnit模块，首先先用与门阵列获取输入指令的类型，之后再用或门阵列获取具体每个信号对应的值。

代码实现

```
module ControllerUnit(
    input [5:0] Op,
    input [5:0] Func,
    input Z,
    output [1:0] Regrt,
```

```
    output Se,
    output Wreg,
    output Aluqb,
    output [1:0] Aluc,
    output Wmem,
    output [1:0] Pcsrc,
    output [1:0] Reg2reg
    );

wire R_type = ~|Op;
wire I_addu = R_type & Func[5] & ~Func[4] & ~Func[3] & ~Func[2] & ~Func[1] &
Func[0];
wire I_subu = R_type & Func[5] & ~Func[4] & ~Func[3] & ~Func[2] & Func[1] &
Func[0];
wire I_ori = ~Op[5] & ~Op[4] & Op[3] & Op[2] & ~Op[1] & Op[0];
wire I_lw = Op[5] & ~Op[4] & ~Op[3] & ~Op[2] & Op[1] & Op[0];
wire I_sw = Op[5] & ~Op[4] & Op[3] & ~Op[2] & Op[1] & Op[0];
wire I_beq = ~Op[5] & ~Op[4] & ~Op[3] & Op[2] & ~Op[1] & ~Op[0];
wire I_lui = ~Op[5] & ~Op[4] & Op[3] & Op[2] & Op[1] & Op[0];
wire I_jal = ~Op[5] & ~Op[4] & ~Op[3] & ~Op[2] & Op[1] & Op[0];
wire I_jr = R_type & ~Func[5] & ~Func[4] & Func[3] & ~Func[2] & ~Func[1] &
~Func[0];
wire I_j = ~Op[5] & ~Op[4] & ~Op[3] & ~Op[2] & Op[1] & ~Op[0];

assign Regrt[0] = I_ori | I_lw | I_sw | I_beq | I_lui | I_jr | I_j;
assign Regrt[1] = I_jal;
assign Se = I_lw | I_sw | I_beq;
assign Wreg = I_addu | I_subu | I_ori | I_lw | I_lui | I_jal;
assign Aluqb = I_addu | I_subu | I_beq | I_jal | I_jr | I_j;
assign Aluc[1] = I_ori | I_lui;
assign Aluc[0] = I_subu | I_ori | I_beq;
assign Wmem = I_sw;
assign Pcsrc[1] = (I_beq & Z) | I_jal | I_j;
assign Pcsrc[0] = I_jr | I_jal | I_j;
assign Reg2reg[1] = I_jal;
assign Reg2reg[0] = I_addu | I_subu | I_ori | I_sw | I_beq | I_lui | I_jr | I_j;
assign IsSyscall = R_type & ~Func[5] & ~Func[4] & Func[3] & Func[2] & ~Func[1] &
~Func[0];

endmodule
```