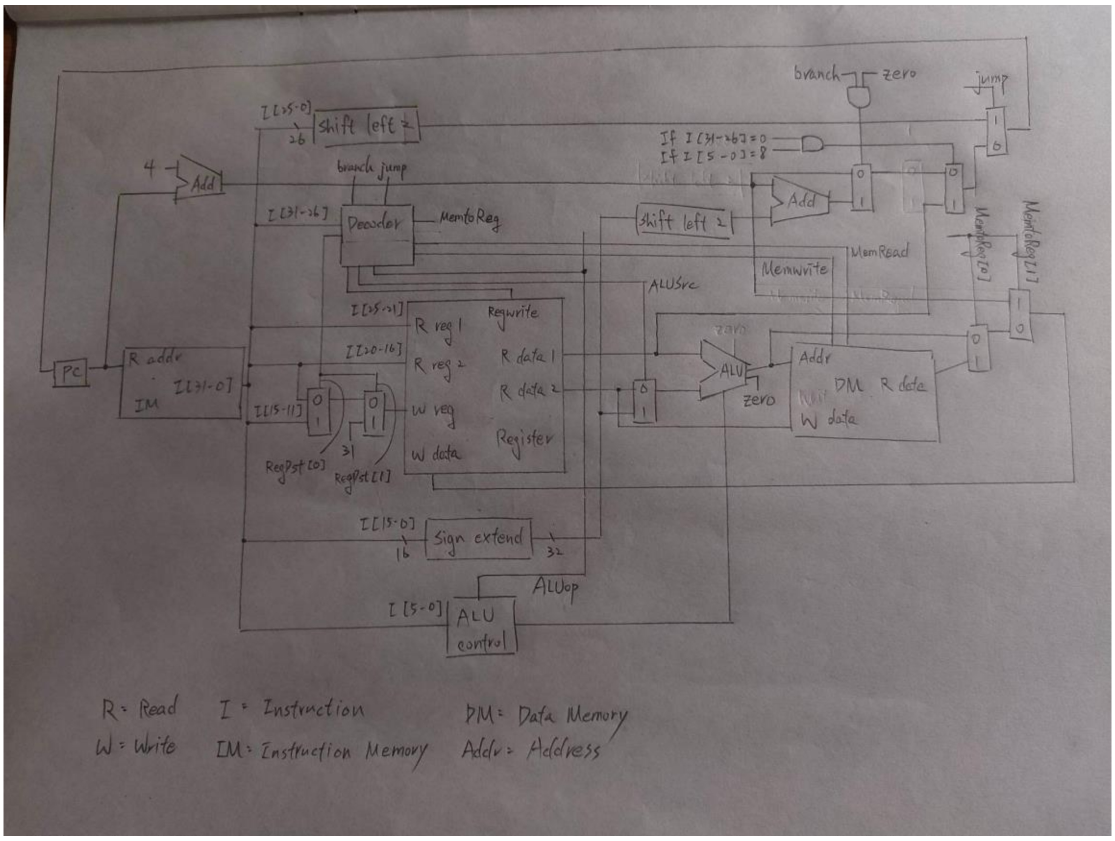
Computer Organization Lab3

Name: 劉秉驊

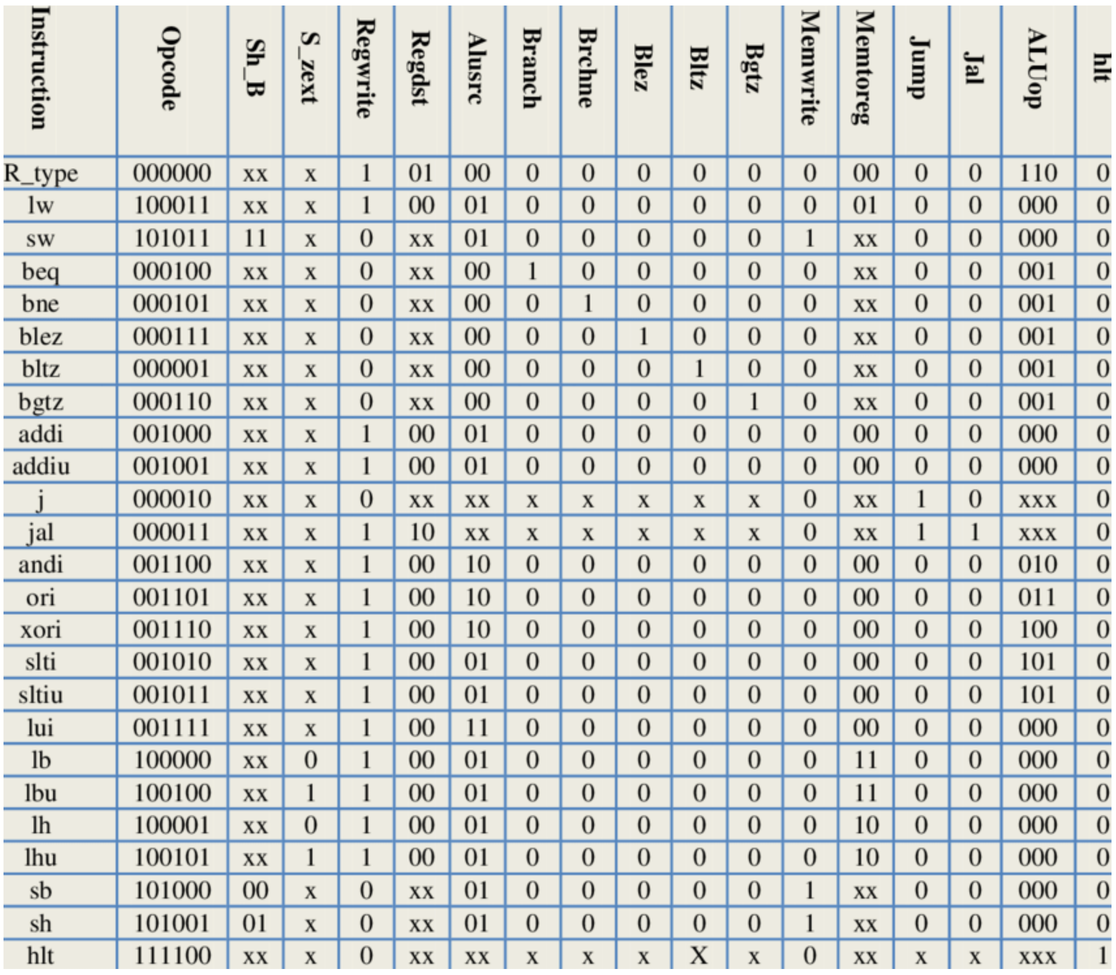
ID: 110550130

Architecture diagrams:



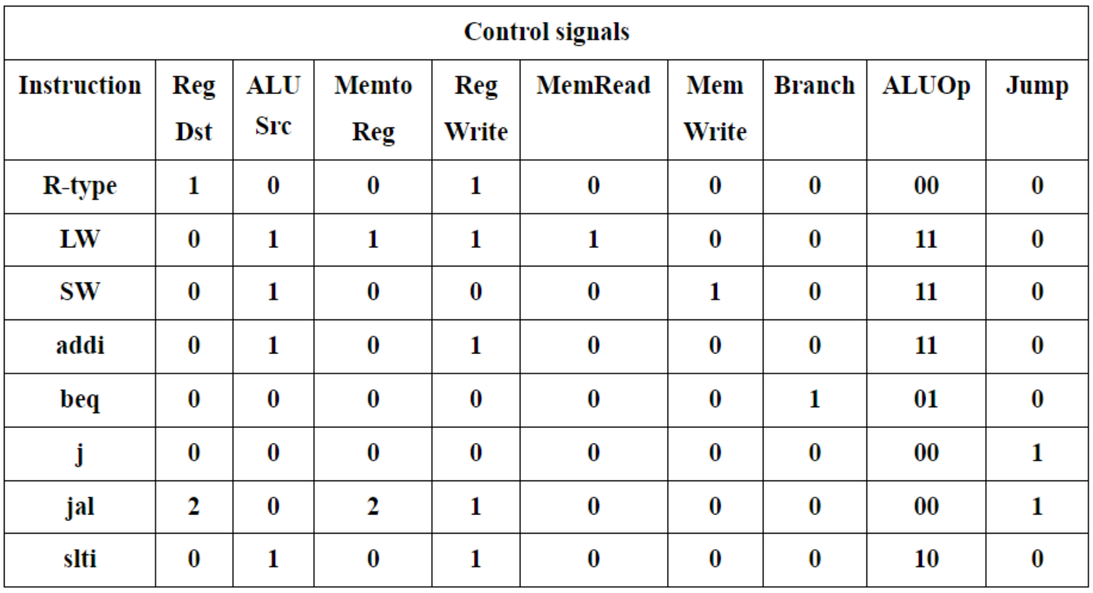
Hardware module analysis:

The first picture is from https://www.researchgate.net/figure/main-control-truth-table_tbl2_317490993



Instruction	Opcode	Sh_B	S_zext	Regwrite	Regdst	Alusrc	Branch	Brchne	Blez	Bltz	Bgtz	Memwrite	Memtoreg	Jump	Jal	ALUop	hlt
R_type	000000	xx	x	1	01	00	0	0	0	0	0	0	00	0	0	110	0
lw	100011	xx	x	1	00	01	0	0	0	0	0	0	01	0	0	000	0
sw	101011	11	x	0	xx	01	0	0	0	0	0	1	xx	0	0	000	0
beq	000100	xx	x	0	xx	00	1	0	0	0	0	0	xx	0	0	001	0
bne	000101	xx	x	0	xx	00	0	1	0	0	0	0	xx	0	0	001	0
blez	000111	xx	x	0	xx	00	0	0	1	0	0	0	xx	0	0	001	0
bltz	000001	xx	x	0	xx	00	0	0	0	1	0	0	xx	0	0	001	0
bgtz	000110	xx	x	0	xx	00	0	0	0	0	1	0	xx	0	0	001	0
addi	001000	xx	x	1	00	01	0	0	0	0	0	0	00	0	0	000	0
addiu	001001	xx	x	1	00	01	0	0	0	0	0	0	00	0	0	000	0
j	000010	xx	x	0	xx	xx	x	x	x	x	x	0	xx	1	0	xxx	0
jal	000011	xx	x	1	10	xx	x	x	x	x	x	0	xx	1	1	xxx	0
andi	001100	xx	x	1	00	10	0	0	0	0	0	0	00	0	0	010	0
ori	001101	xx	x	1	00	10	0	0	0	0	0	0	00	0	0	011	0
xori	001110	xx	x	1	00	10	0	0	0	0	0	0	00	0	0	100	0
slti	001010	xx	x	1	00	01	0	0	0	0	0	0	00	0	0	101	0
sltiu	001011	xx	x	1	00	01	0	0	0	0	0	0	00	0	0	101	0
lui	001111	xx	x	1	00	11	0	0	0	0	0	0	00	0	0	000	0
lb	100000	xx	0	1	00	01	0	0	0	0	0	0	11	0	0	000	0
lbu	100100	xx	1	1	00	01	0	0	0	0	0	0	11	0	0	000	0
lh	100001	xx	0	1	00	01	0	0	0	0	0	0	10	0	0	000	0
lhu	100101	xx	1	1	00	01	0	0	0	0	0	0	10	0	0	000	0
sb	101000	00	x	0	xx	01	0	0	0	0	0	1	xx	0	0	000	0
sh	101001	01	x	0	xx	01	0	0	0	0	0	1	xx	0	0	000	0
hlt	111100	xx	x	0	xx	xx	x	x	x	X	x	0	xx	x	x	xxx	1

The second picture is from https://www.fpga4student.com/2017/01/verilog-code-for-single-cycle-MIPS-processor.html



Control signals								
Instruction	Reg Dst	ALU Src	Memto Reg	Reg Write	MemRead	Mem Write	Branch	ALUOp	Jump
R-type	1	0	0	1	0	0	0	00	0
LW	0	1	1	1	1	0	0	11	0
SW	0	1	0	0	0	1	0	11	0
addi	0	1	0	1	0	0	0	11	0
beq	0	0	0	0	0	0	1	01	0
j	0	0	0	0	0	0	0	00	1
jal	2	0	2	1	0	0	0	00	1
slti	0	1	0	1	0	0	0	10	0

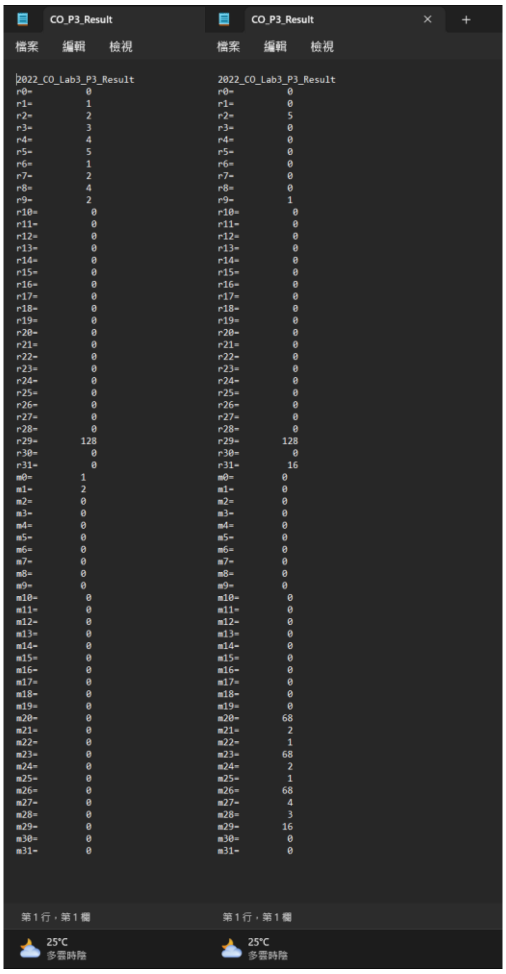
Basically, it’s mostly the same as Lab 2. But there are some differences in Decoder, besides more outputs in Lab 3.

1. jal neels the second digit, so RegDst becomes 2 digits.
2. Since the hardware Architecture, we need MemtoReg to distinguish jal. Thus, MemtoReg of jal is 2’b10 instead of 2’bxx in the first picture same in Lab 2.
3. I search for MemRead code in the second picture.

Finally, I constructed Decoder with code in 2 pictures above.

Finished part:

The left one below is data 1, and the right one below is data 2.
All reaches the requirements.



```
2022_CO_Lab3_P3_Result
r0=        0
r1=        1
r2=        2
r3=        3
r4=        4
r5=        5
r6=        1
r7=        2
r8=        4
r9=        2
r10=        0
r11=        0
r12=        0
r13=        0
r14=        0
r15=        0
r16=        0
r17=        0
r18=        0
r19=        0
r20=        0
r21=        0
r22=        0
r23=        0
r24=        0
r25=        0
r26=        0
r27=        0
r28=        0
r29=      128
r30=        0
r31=        0
m0=       1
m1=       2
m2=       0
m3=       0
m4=       0
m5=       0
m6=       0
m7=       0
m8=       0
m9=       0
m10=       0
m11=       0
m12=       0
m13=       0
m14=       0
m15=       0
m16=       0
m17=       0
m18=       0
m19=       0
m20=       0
m21=       0
m22=       0
m23=       0
m24=       0
m25=       0
m26=       0
m27=       0
m28=       0
m29=       0
m30=       0
m31=       0
```

```
2022_CO_Lab3_P3_Result
r0=        0
r1=        0
r2=        5
r3=        0
r4=        0
r5=        0
r6=        0
r7=        0
r8=        0
r9=        1
r10=        0
r11=        0
r12=        0
r13=        0
r14=        0
r15=        0
r16=        0
r17=        0
r18=        0
r19=        0
r20=        0
r21=        0
r22=        0
r23=        0
r24=        0
r25=        0
r26=        0
r27=        0
r28=        0
r29=      128
r30=        0
r31=       16
m0=       0
m1=       0
m2=       0
m3=       0
m4=       0
m5=       0
m6=       0
m7=       0
m8=       0
m9=       0
m10=       0
m11=       0
m12=       0
m13=       0
m14=       0
m15=       0
m16=       0
m17=       0
m18=       0
m19=       0
m20=      68
m21=       2
m22=       1
m23=      68
m24=       2
m25=       1
m26=      68
m27=       4
m28=       3
m29=      16
m30=       0
m31=       0
```

Problems you met and solutions:

The first thing is that because of some required functions, we need to know 2 digits code such as RegDst. Thus, there'll be 3 cases to output the result in that condition. Without MUX 3 to 1 function which is not provided, I made 1-level 3 cases into 2-level 2 cases that required more and more wires to connect. The second is that I was confused with j and jal and jr then. What's the difference of the ways of connection among them? At the end, I found out j and jr were similar, and jal was much different that needed more information from PC and address. And I used a MUX 2 to 1 to represent jr when its opcode and function code emerged.

Summary:

To me, this is totally a new thing I've never touched. I spent lots of time handling the basic knowledge and searching the net. I'm quite happy to construct the simple cpu that makes me more understand how computer works. I think I will be willing to delve into the world of hardware.