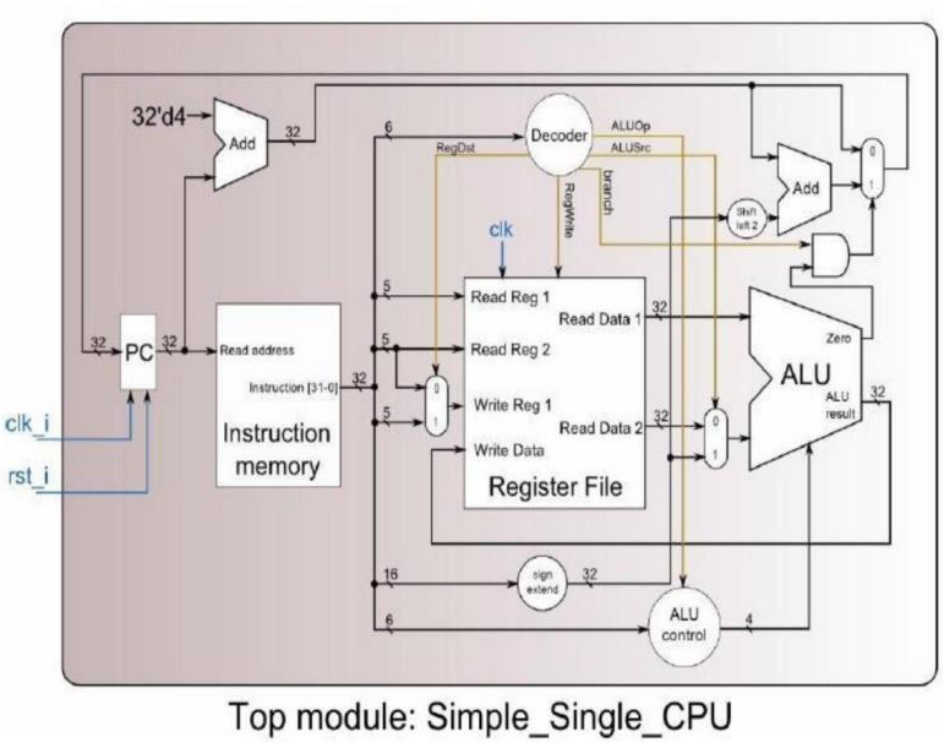
Computer Organization

Architecture diagrams:



Top module: Simple_Single_CPU

Hardware module analysis:

Decoder:

Every instruction starts with 6-bit opcode. By the table below, Decoder use the six bits to specifies the operation to be performed. In this lab, we can divide into 4 categories.

If these six bits are all zero(R-format), we should use the last six bits to determine which operation need to do.

Reg_Write=1, if the result need to write back.

ALUSrc_o=1,if the second operand comes from immediate

Op (31:26)								
28-26 31-29	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	R-format	Bltz/gez	Jump	Jal	Beq	Bne	Blez	Bgtz
1(001)	Addi	Addiu	Slti	Sltiu	Andi	Ori	Xori	Lui
2(010)	TLB	FIPt						
3(011)								
4(100)	Lb	Lh	Lwl	Lw	Lbu	Lhu	lwr	
5(101)	Sb	Sh	Swl	Sw			Swr	
6(110)	Lwc0	Lwc1						
7(111)	Swc0	swc1						

ALU:

Source1 from Register File, Source2 depends on the MUX

With the output of ALUCtrl, ALU knows what operation should do.

ALUCtrl:

Use the output of Decoder(ALUOp_i), if the ALUOp_i=000, we need to take a closer look at which operation need to be performed.

MUX_2to1:

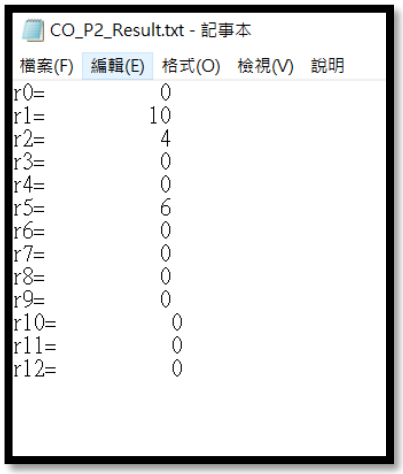
Choose which input should be selected as output.

Sign_Extend:

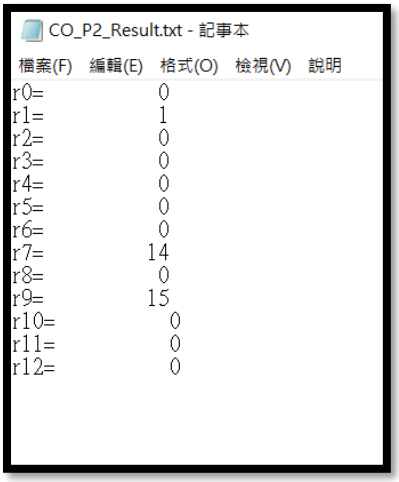
According to the 15th bit, replicate the sign bit to the left.

Finished part:

>>data1



>>data2



Problems you met and solutions:

1.The input and output of each submodule are east to confuse, so I did it step by step according to the architecture diagram, and took the easily recognizable variable

name.
2.Almost every submodule needs the output of decoder to do their part, so I finished decoder before others.

Summary:

Seemingly complicated structure, it becomes much easier to complete each part separately. As mentioned in lab1, Implementing the design by connecting multiple modules together will reduce the complexity.