CA hw4 report:

Modules: Control.v, Adder.v, MUX32.v, Sign_Extend.v, ALU.v, ALU_Control.v

1. Control.v:

   Control module read opcode of the instruction as input, and output ALUop, ALUSrc and RegWrite. ALUop will be passed to ALU_control, ALUSrc will be passed to MUX32 and in this homework, the final result always need to be written in RD, so we assign RegWrite always be 1.

   Opcode convert to ALUop, ALUSrc:

fuction	opcode	ALUop	ALUsrc
and	0110011	10	0
xor	0110011	10	0
sll	0110011	10	0
add	0110011	10	0
sub	0110011	10	0
mul	0110011	10	0
addi	0010011	00	1
srai	0010011	00	1

2. Adder.v

   Adder module read PC and a constant 4 as input, and output next PC

   data_o = data1_in+ data2_in

3. MUX32 module read data of source register 2, extended immediate of instruction and ALUSrc as input, and output the selected data.

   If ALUSrc == 0, select data of source register 2(data1_in)

   Else select immediate(data1_in)

   The result will be passed to ALU

4. Sign_Extend module read immediate of instruction as input and output the extended(32 bit) data.

   we need to check whether the immediate is positive or negative by its last bit(data_i[11]), and padding data_i[11] in 12 to 31 bit of the output

5. ALU_Control module read fuct7 +fuct3 of instruction and ALUop as input, and output ALUCtrl and pass to ALU.

   Fuct7, fuct3 and ALUop convert to ALUCtrl

fuction	fuct7	fuct3	ALUop	ALUCtrl
and	0000000	111	10	001
xor	0000000	100	10	011
sll	0000000	001	10	010
add	0000000	000	10	000
sub	0000000	000	10	100

mul	0000000	000	10	101
addi	x	000	00	000
srai	0000000	101	00	110

6. ALU module read data in register 1 , data in register 2(or immediate) and ALUCtrl as input, and output the result after doing some arithmetic operation, use ALU_Ctrl to determine which operation should do. The other output Zero always be 0 in the assignment(we don’t consider branch operation)

   ALU_Ctrl convert to operation:

ALU_Ctrl	
001	and(data1 & data2)
011	xor(data1^data2)
010	sll(data1 << data2)
000	add(data1+data2)
100	sub(data1-data2)
101	mul(data1*data2)
000	addi(data1*data2(immediate))
110	srai(data1 >> data2(immediate))