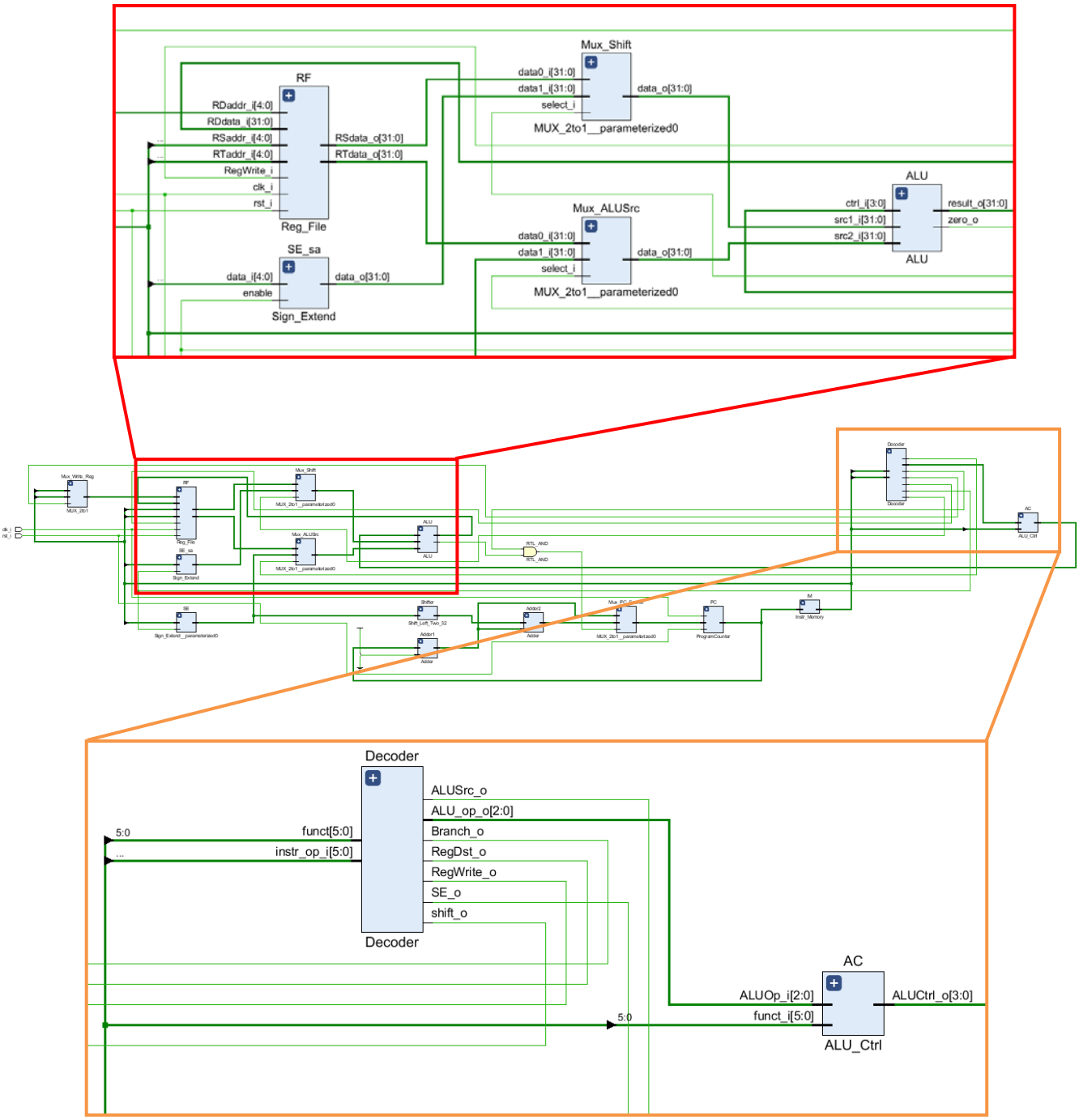
Computer Organization

Architecture diagram:

The architecture is basically same as the picture given in CO_Lab_2.pdf. The only different is we add a Mux_Shift to determine ALU input should be rsdata_o or shamt when handling the sra instruction.

Moreover, we add two output, se_o and shift_o, for Decoder, which represent the signal control for sign_extend and Mux_Shift respectively.



Detailed description of the implementation:

Decoder Design

	Op	Func	ALUSrc	Branch	RegWrite	RegDst	Signed	Shift	ALUOp
addu	000000	100001	0	0	1	1	x	0	010
addi	001000	-	1	0	1	0	1	0	010
subu	000000	100011	0	0	1	1	x	0	010
and	000000	100100	0	0	1	1	x	0	010
or	000000	100101	0	0	1	1	x	0	010
slt	000000	101010	0	0	1	1	x	0	010
sltiu	001011	-	1	0	1	1	0	0	110
beq	000100	-	0	1	0	x	1	0	001
sra	000000	000011	0	0	1	1	x	1	010
srav	000000	000111	0	0	1	1	x	0	010
lui	001111	-	1	0	1	0	1	0	011
ori	001101	-	1	0	1	0	0	0	100
bne	000101	-	0	1	0	x	1	0	101

ALUCtrl Design

ALUOp	Func	ALUCtrl
000	-	0010
001	-	0110
010	100001	0010
	100011	0110
	100100	0000
	100101	0001
	101010	0111
	000011	1000
	000111	1000
011	-	1001
100	-	0001
101	-	1010

Problems encountered and solutions:

- **Problem: Failed when implementing sra (result_o = $signed(src2_i) >>> src1_i)**
- **Solutions: It turned out to be that not only src2_i should be signed but also the result_o itself, so we declared result_o as signed then fixed.**

Lesson learnt (if any):

- **Understand how to distribute control code to functions through decoder and ALUcontrol.**
- **Understand how a simple single cpu works**