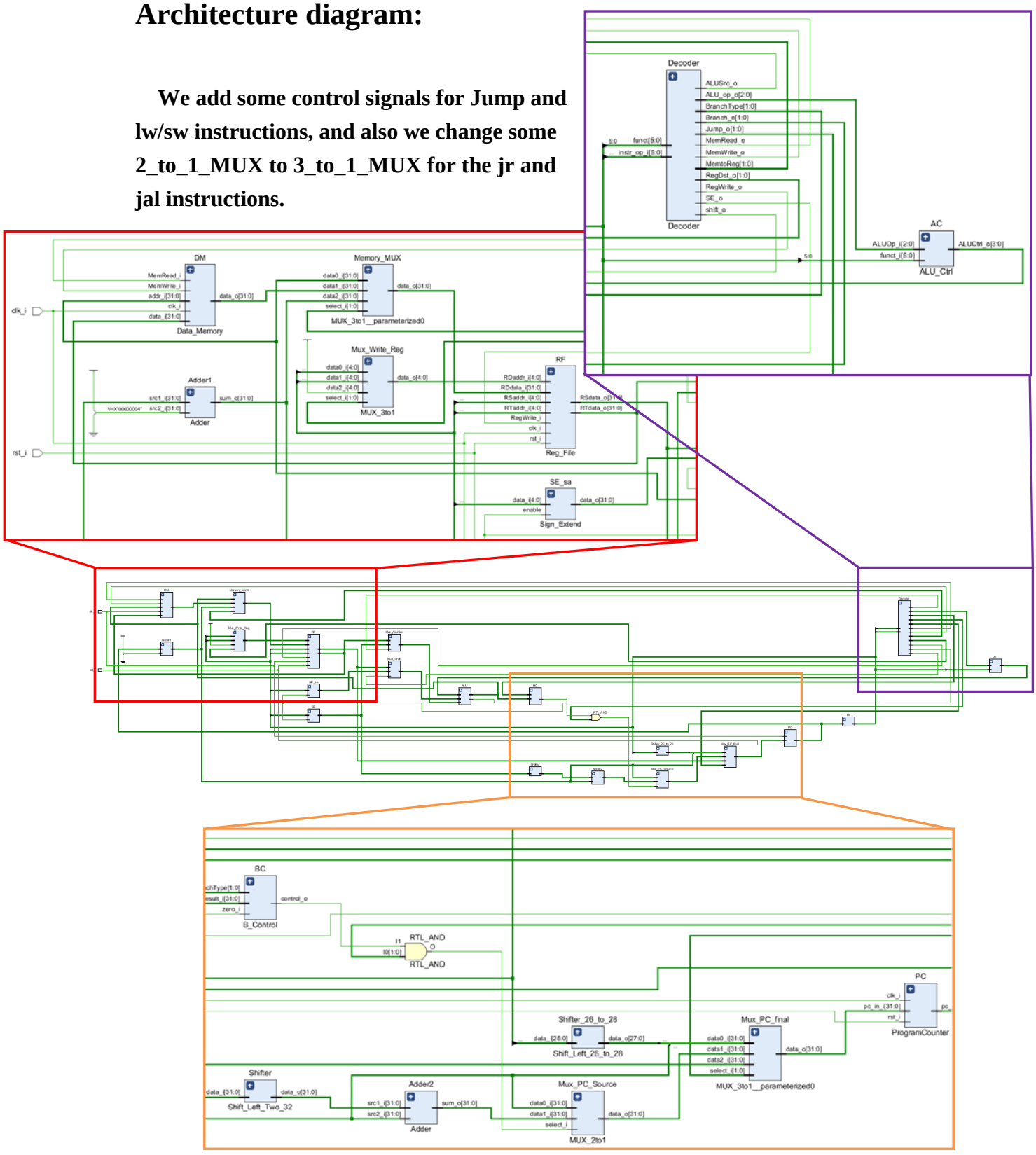
Computer Organization

Architecture diagram:

We add some control signals for Jump and lw/sw instructions, and also we change some 2_to_1_MUX to 3_to_1_MUX for the jr and jal instructions.



Detailed description of the implementation:

Decoder Design (some addition and modification from lab2)

	Op	Func	RegDst	Mem toReg	Mem Read	Mem Write	Branch Type	Jump	ALUOp
lw	100011	-	0	1	1	0	x	1	000
sw	101011	-	x	x	0	1	x	1	000
mul	000000	011000	1	0	0	0	x	1	010
j	000010	-	x	0	0	0	x	0	x
jal	000011	-	2	2	0	0	x	0	x
jr	000000	001000	x	x	x	x	x	2	010
ble	000110	-	x	x	0	0	2	1	111
bnez	000101	-	x	x	0	0	1	1	001
bltz	000001	-	x	x	0	0	3	1	100
li	001111	-	0	0	0	0	x	1	000

Problems encountered and solutions:

- **Problem 1: Having no idea how to implement jal and jr through the original diagram (from lab2)**
- **Solution to Problem 1: Turn some 2to1Muxs into 3to1Muxs and give proper control signals then fix.**
- **Problem 2: Having some compiling errors and not understanding the log.**
- **Solution to Problem 2: We use iVerilog and Vivado at the meantime, they both give log help us to debug.**

Lesson learnt (if any):

- **Knowing how to store and load data through data memory.**
- **Understanding how to implement jump functions, including how to store current PC into register by using correct 3to1Mux and correct decoding.**
- **Understanding how to translate MIPS into machine code.**
- **Knowing how to implement different types of branch functions.**