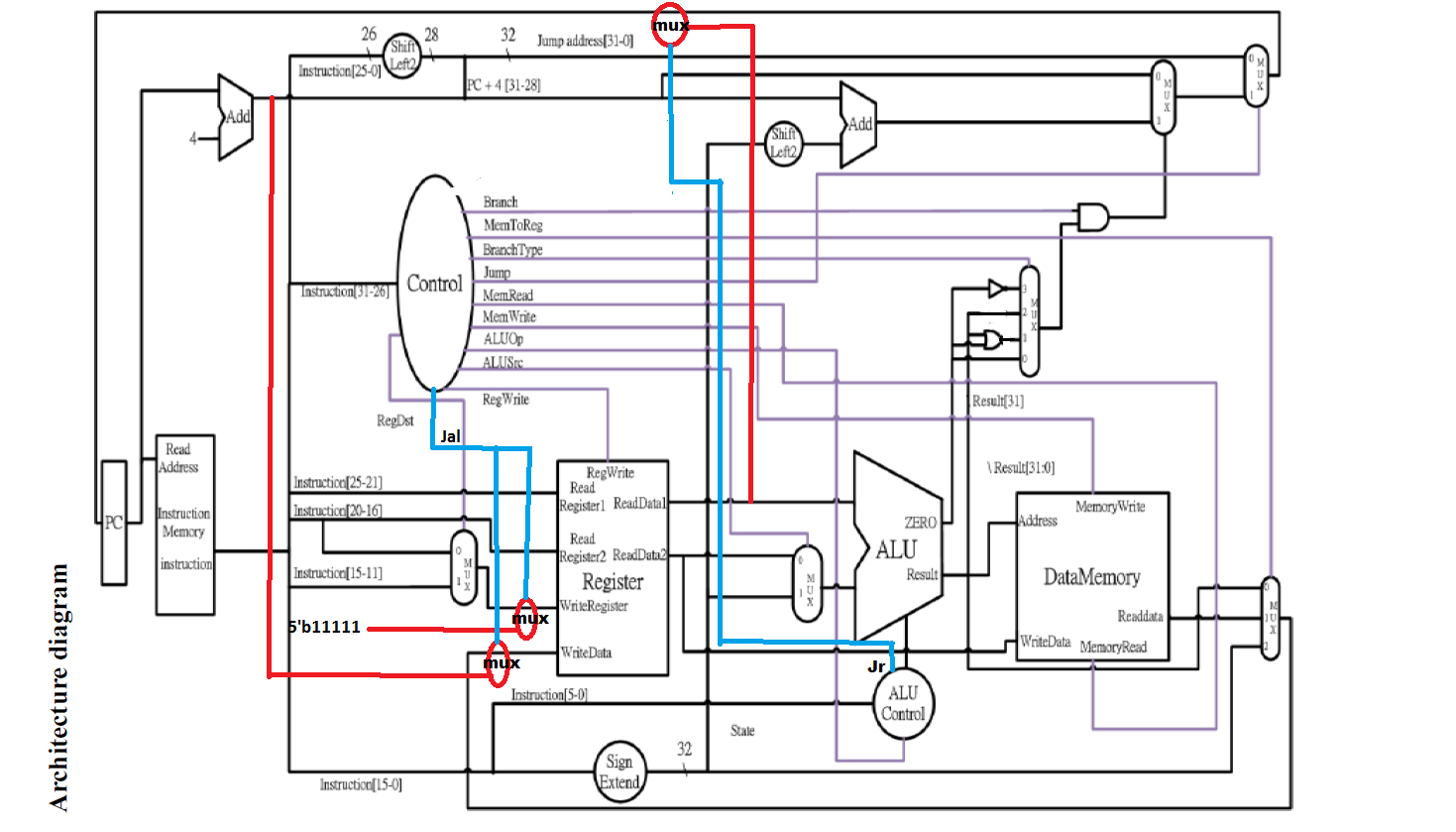
**Computer Organization**

**0416021曾香耘 0416246王彥茹**

**Architecture diagram:**



**Detailed description of the implementation:**

Done everything required:

For LW and SW, we connected data memory into the top module along with MemRead, MemWrite, MemToReg from the decoder.

For J, we added a shifter that shifts target by 2 bits and concatenated those 28 bits with instruction [31:28] from PC + 4.

For MUL, we modified the ALU so it now has more functionality.

For JAL, the jumping part is basically identical to that of J, so we only added 2 muxs, one of them for WriteRegister (to pass in 31), the other for WriteData. We chose to add new 2 to 1 muxes instead of modifying the old muxes, for them to be 3 to 1, because of simplicity on implementation. Both muxs uses the line jal that was added to the decoder as selection lines.

For JR, we added a mux between the register file and pc to select the next ingoing pc. The selection line jr is added to ALU\_Ctrl (as JR is a r-type insrruction).

For BLE, BLT, BNEZ and BEQ, we added a mux that selects from zero, zero | alu\_result[31:31], alu\_result[31:31], and ~zero, while the selection line BranchType is added to the decoder. Also, because of the poor implementation, we removed the parts for BNE that was from the last lab, now, BNE and BNEZ has the same implementation (since they have the same opcode) and both work as well. We used to have different alu\_op for BEQ and BNE (because we used to distinguish them by doing different operations in the ALU), now we only have 1 alu\_op for all four branch instructions!

For LI, we modified LUI from the last lab, only modifying the ALU so instead of “result\_o = src2\_i << 16;” it now does “result\_o = src2\_i;”. We chose to keep the operation in the ALU for implementation simplicity.

Last but not least, of course we encoded new instruction sets into alu\_op, and decoded the control signals for them.

**Problems encountered and solutions:**

At first sight we wanted to decode JR along with other signals within the decoder, then we figured out it was a r-type instruction, so instead, we had the control signal coming out from ALU\_Ctrl.

The architecture diagram provided was not specific enough, for example for JAL we had to add 2 muxes which didn’t show up on the diagram. What’s more, the mux for the branches in the diagram is for BGE and BGT, which was really confusing, so we modified some of the gates (using their complement) to compensate our design.

For test data 3, the clock cycle provided was too long that ISE couldn’t have the correct result. We thought our design had bug in it and took an hour investigating, finally we found out it was because of the clock cycle. We modified the clock cycle from 50 to 5 and everything works fine.

**Lesson learnt (if any):**

Redundant design that was once considered a feature may be taken off in the next lab.

It would be frustrating if we had to do a pipelined version of the current design.