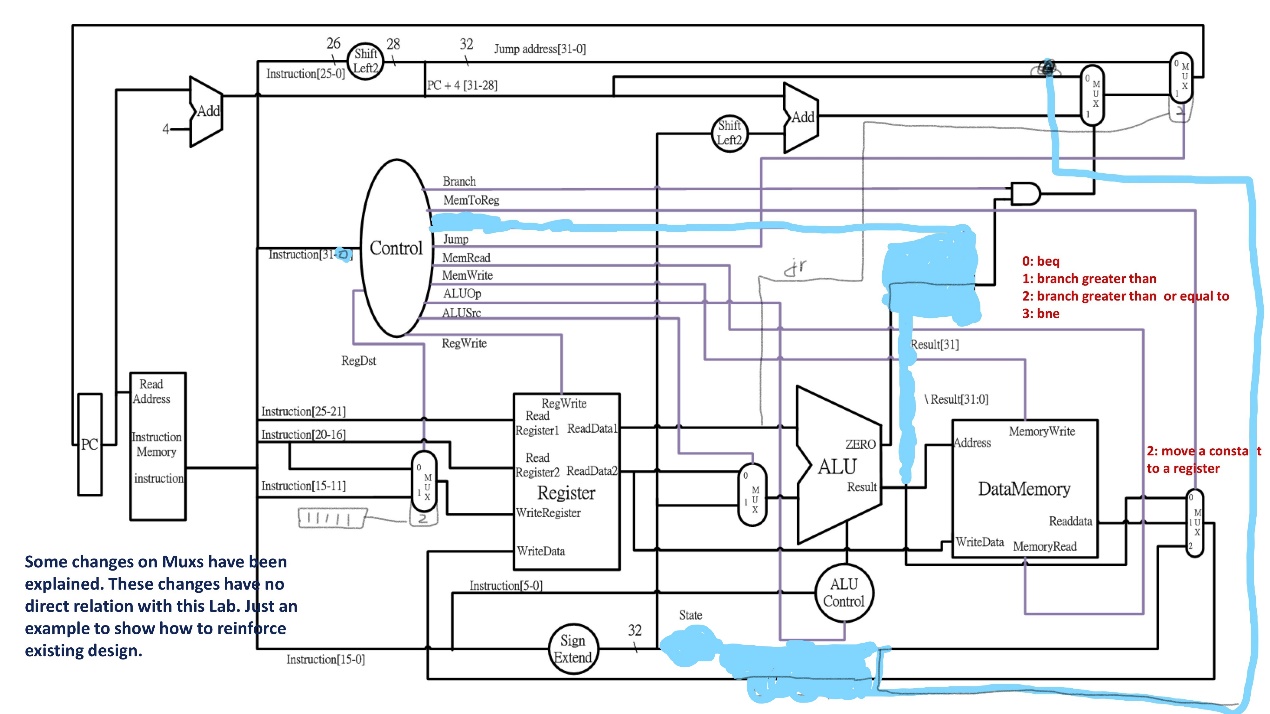
**Computer Organization Lab3**

**Name:陳星宇**

**ID:109550060**

**Architecture diagrams:**

****

**Referenced by TA’s elaborate diagram :)**

**Hardware module analysis:**

**Combined with the lab finished last time, this time we need to implement lw/sw function as well as jump related functions. Lw/sw function is performed in the DataMemory module and operated by the decoder module. Jump related functions are controlled by jump\_o in the decoder. Jal function further requires to save to register[31], so I change Regdst into 2 bit controller of the mux. Jr function requires loading the register into PC, so I wire the rddata\_o and use jump\_o to serve as a mux controller. Another issue is that the opcode of Jr function and R-format are the same, that’s why I choose to use the whole 32bit instruction as the decoder input.**

**Finished part:**

**Test\_data1**

****

**Test\_data2**

****

**Instruction are changed when facing posedge clock signals, indicating that every time the clock triggers the posedge, PC will give the new instruction and +4 for fetching another one, except for receiving instructions like beq, j, jal or jr, which will change the PC address, after all the instructions are done, PC will still increase while other signals remain still the same.**

**Problems you met and solutions:**

**The template module given is identical to the last lab; however, this time we need to implement more functions. So the first problem I’ve met is the lack of existing module, while some requires extended feature like shift\_left26\_to\_28 and a 3to1\_mux. The second problem I’ve encountered is I forgot that the return address of $ra in the jal function requires only PC+4, I thought that was PC+8, so when I first run my program, it stuck in the infinite loop as it will not trigger the jr function in the test\_data2. After fixing the bug, it runs perfectly.**

**Summary:**

**It is quite interesting to implement almost every MIPS instruction on our own, and it also makes me understand more about those binary instruction as what they actually do and change the state of memory and registers, and also, my thought toward those seemingly identical zeroes and ones.**