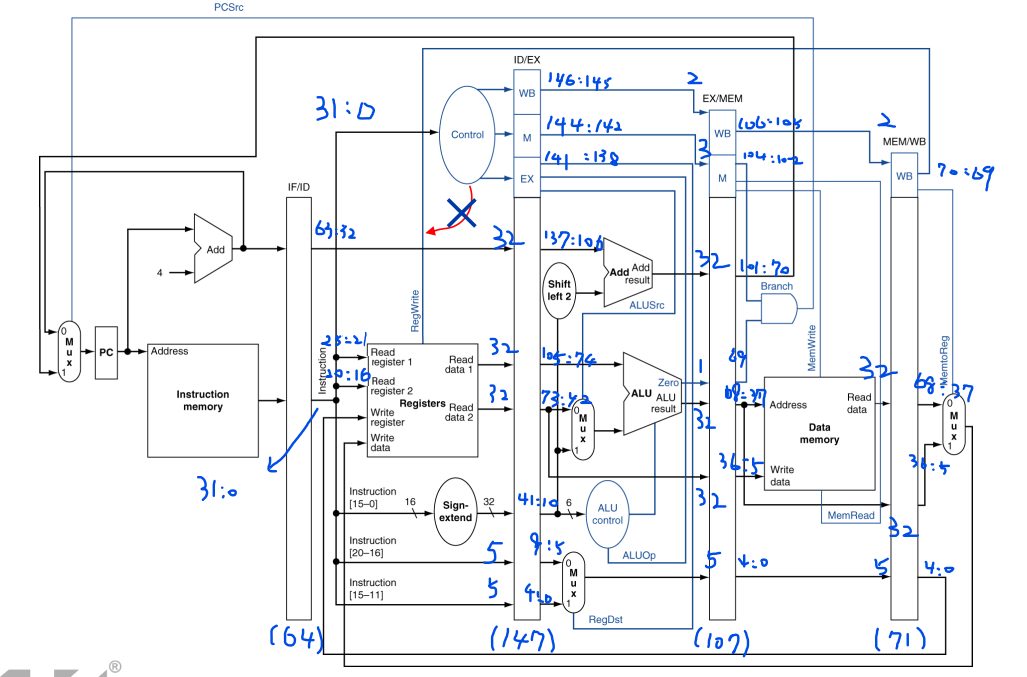
**Computer Organization Lab4**

**Name:陳星宇**

**ID:109550060**

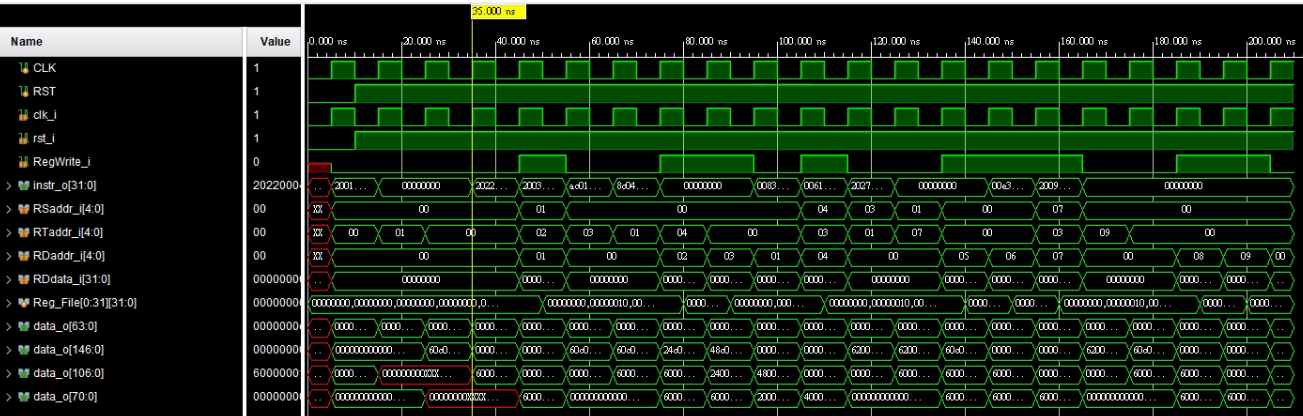
**Architecture diagrams:**

****

**Hardware module analysis:**

**Pipeline module allows to increase the throughput of instructions at the same time using pipeline register to hold information from the last part; however, it suffers from different hazards such as data hazard and branch hazard. If the module is not configured to detect the hazards itself, it requires some reorganize or insertion of NOPs.**

**Finished part:**

****

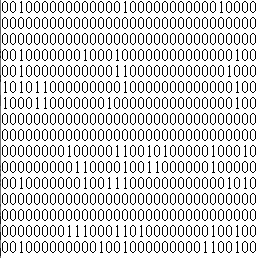
**As it shows, the first time the register outputs change are 4 clock cycles after the instruction was fetched, that’s because the pipeline strategy has 5 stages to perform a instruction.**

**Problems you met and solutions:**

**The part I’ve stuck is before the first instruction is passed into the IF/ID register, it has 64’b0 by default and therefore it will pass 32’b0 into the decoder. I didn’t notice that and I didn’t write operation for NOP instructions to deal with the problem. So before the first instruction reached, the decoder already sent the R-type operation as opcode equals to zero, which causes RegDst will be 1 before the first instruction reached WB part, leading to Xs in some register. After writing operations for the NOP instruction, the problem is solved correctly.**

**Bonus (optional):**

**Solution:**

****

**Explanation:**

**Add two NOPs between every two instructions that leads to data hazard.**

**Summary:**

**It’s interesting to do the same instructions in several different ways from Lab2 to Lab4, each has its own strengths and weaknesses. I’ve learned a lot from these labs of understanding more and more that how to deal with the MIPs instructions efficiently. Despite the module is more and more complicated, it still has a lot of fun when building them.**