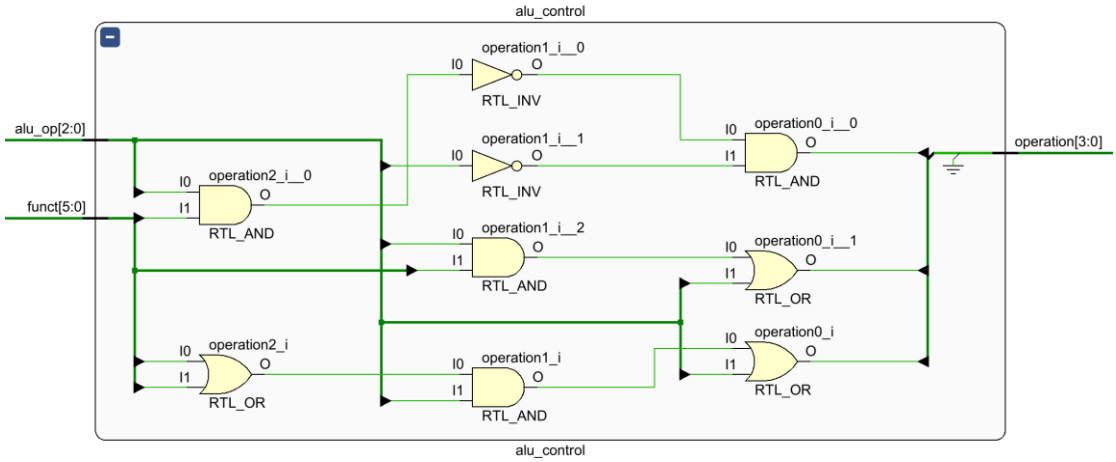
Lab2: Single-Cycle Processor

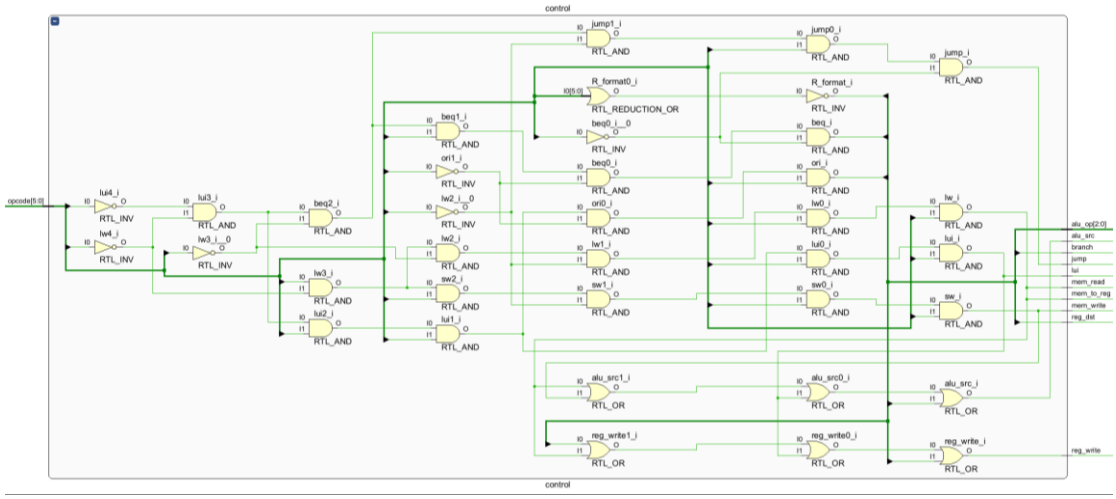
111550142 尤瑋辰

1. Architecture Diagrams
   a. ALU control



Most of the design follows Appendix D – 6. Additionally, I expanded the alu_op from 2 bits to 3 bits, with alu_op[2] specifically designated for ori. 100(ori) => 0011 (or)

   b. Main control



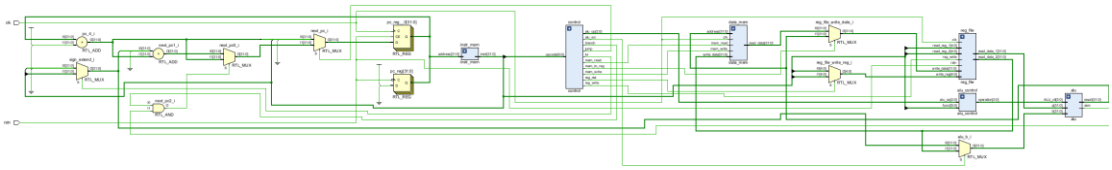
Most of design follows Appendix D – 8. Additionally, I added jump and lui signal for jump and load Immediate, and expanded the alu_op as mentioned in part 1-a.

Here are the detailed controls for the added signals:

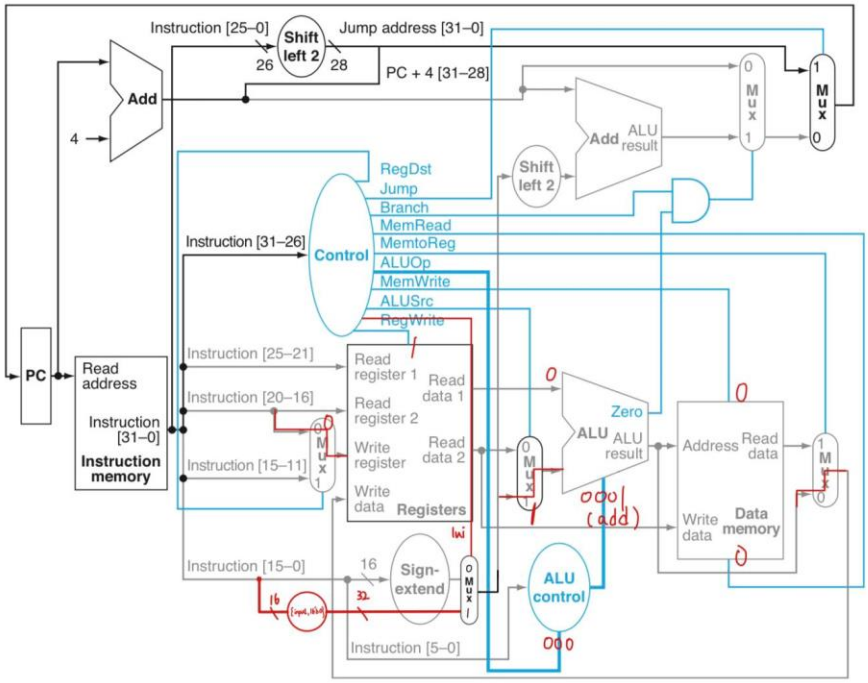
jump = op: 000010, lui = op: 001111, ori = op: 001101

	reg_dst	alu_src	mem_to_reg	reg_write	mem_read	mem_write	branch	alu_op
jump	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*xxx*
lui	*0*	*1*	*0*	*1*	*0*	*0*	*0*	*000 (add)*
ori	*0*	*1*	*0*	*1*	*0*	*0*	*0*	*100 (or)*

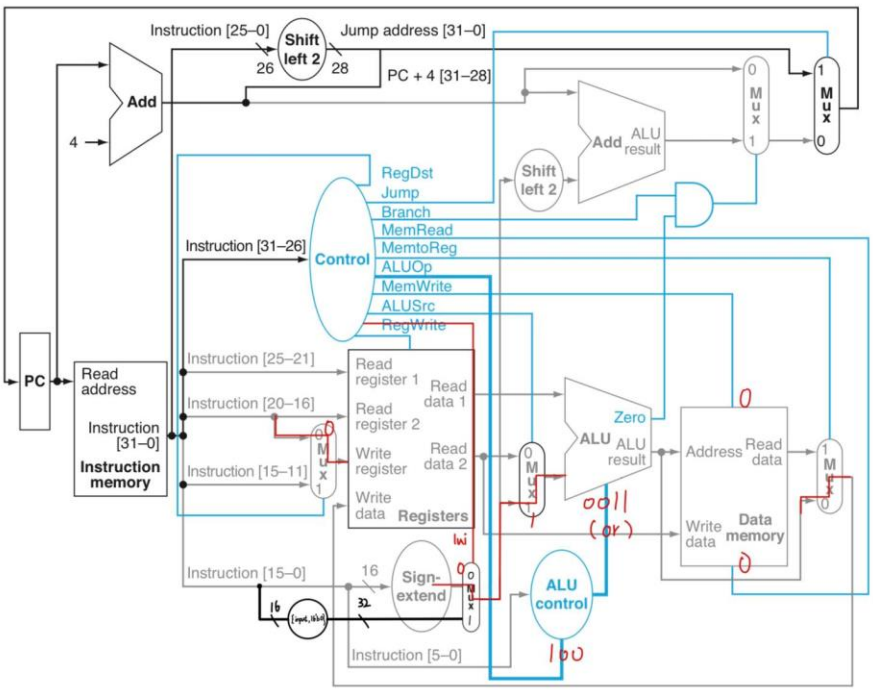
c. Single-cycle processor



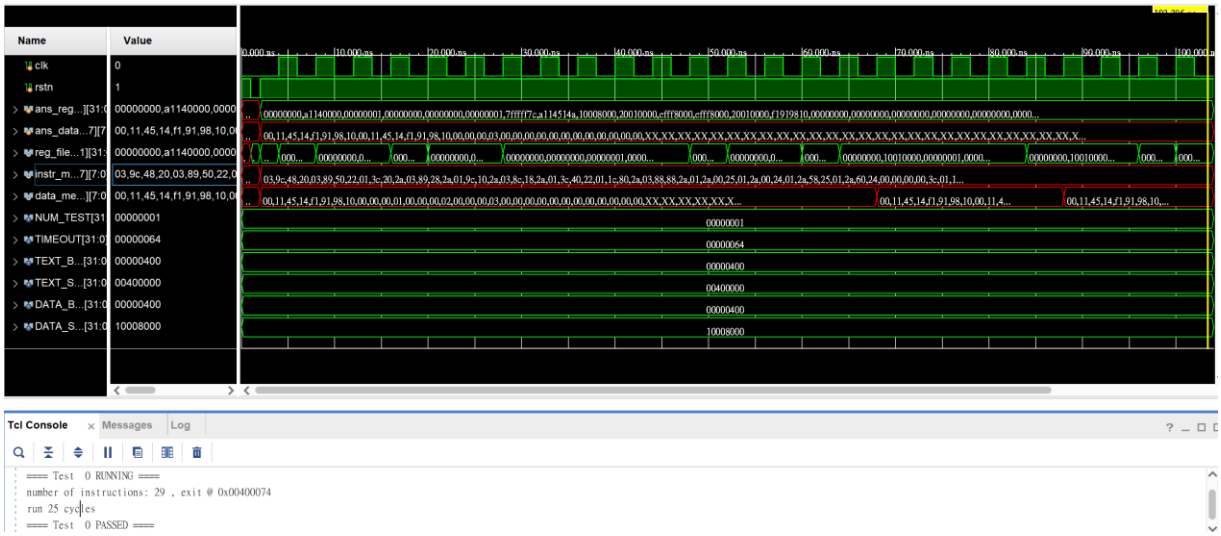
Most of design follows textbook (P.329). Additionally, I added a MUX for lui instruction. Here is the lui instruction:



Here is the ori instruction:



2. Experimental Result



3. Answer the following Questions:
   1. When does write to register/memory happen during the clock cycle? How about read?

      Writing to register/memory occurs during the Write Back stage, which is the final stage in the positive cycle.

      Reading from register/memory occurs during the Read stage, which is the initial stage in the positive cycle.
   2. Translate the "branch" pseudo instructions (blt , bgt , ble , bge) in the Green Card into real instructions. Only at register can be modified, and other common registers should not be modified.

      blt $s1, $s2, label (<)

      slt $t0, $s1, $s2 # set boolean temp $t0 to $s1 < $s2

      bne $t0, $0, label # branch on boolean temp is "true"

      bgt $s1, $s2, label (>)

      slt $t0, $s2, $s1 # set boolean temp $t0 to $s2 < $s1

      bne $t0, $0, label # branch on boolean temp is "true"

      ble $s1, $s2, label (>=)

      slt $t0, $s2, $s1 # set boolean temp $t0 to $s2 < $s1

      beq $t0, $0, label # branch on boolean temp is "false"

      bge $s1, $s2, label (>=)

      slt $t0, $s1, $s2 # set boolean temp $t0 to $s1 < $s2

      beq $t0, $0, label # branch on boolean temp is "false"

3. Give a single beq assembly instruction that causes infinite loop. (consider that there's no delay slot)

   beq $0, $0, -4

   Since 0 equals 0, the next PC will be PC + 4 – 4. In other words, it will execute the same instruction in the next clock cycle..
4. The j instruction can only jump to instructions within the "block" defined by "(PC+ 4) [31:28]". Design a method to allow j to jump to the next block (block number + 1) using another j.

   Define the new jump instruction as "jb" and add an additional MUX after the original jump MUX. If the instruction is "jb", select {(PC+4) [31:28] + 1, address, 2'b00}; otherwise, choose the original jump.
5. Why a Single-Cycle Implementation Is Not Used Today?

   The primary reason is performance limitation. In a single-cycle implementation, each instruction takes the same amount of time to execute, regardless of its complexity. Hence, the duration of the clock cycle is always determined by the slowest operation, such as load (lw) or store (sw), even in cases where the data memory remains unused. Practical implementations use multiple cycles per instruction, which fixes some shortcomings of the 1-cycle implementation.

4. Problems Encountered & Solution

   Q: It is hard to debug.

   A: To verify the correctness of each signal, I add a register named "display" and connect it to the signal I want to check. In the testbench, I use $display to show the signal after each clock cycle.