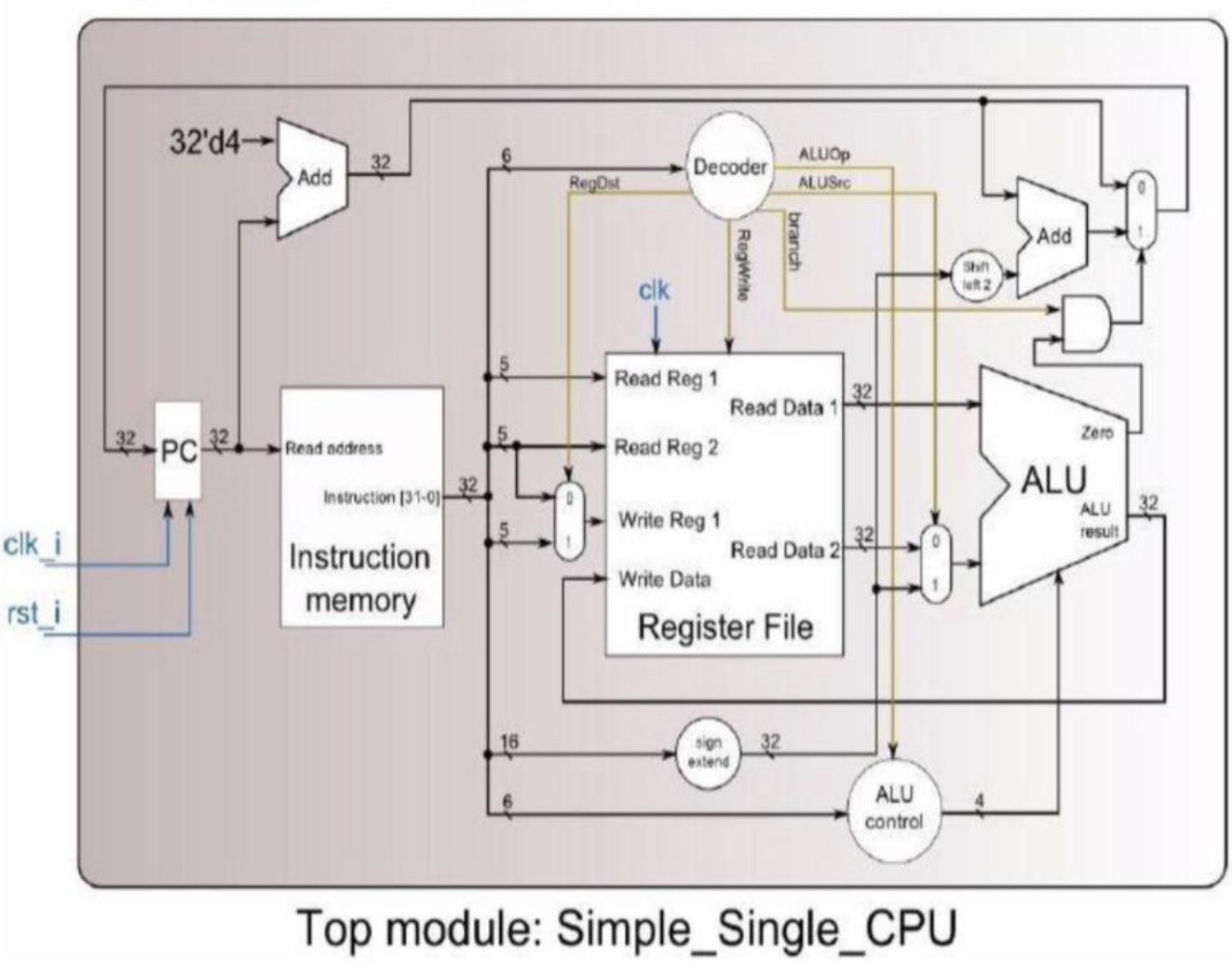
Computer Organization

Lab 2: Single Cycle CPU – Simple Edition

110550108 施柏江

Architecture diagrams:



Top module: Simple_Single_CPU

Hardware module analysis:

Adder.v: 將兩個 32bit 的 input 相加。

ALU_Ctrl.v: 根據 instruction field 的 opcode 和 decoder 的 ALU_op 來決定要讓 ALU 執行何種運算。

ALU.v: 根據 ALU_Ctrl 的結果來決定要對兩個 32bit 的 input 做何種運算。

Decoder.v: 根據 instruction field 的 opcode 來決定是否有 rd 欄位(RegDst)、是否要寫入 register file(RegWrite)、是否要 branch out(branch)、是否需要做 sign extension(ALUSrc)，以及決定 ALU_Ctrl 要做哪種指令(ALUOp)。

Instr_Memory.v: 根據 PC 傳入的 address，輸出對應的 instruction。

MUX_2to1.v: 根據傳入的 selection control 來決定要輸出哪一筆資料。

ProgramCounter.v: 指向要執行指令的 address。

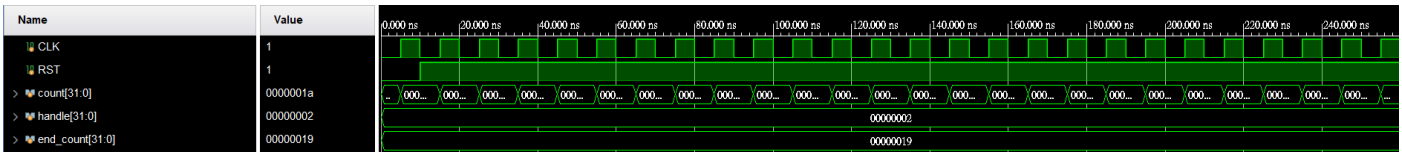
Reg_File.v: 將資料暫存，負責讀入和寫出資料。

Shift_Left_Two_32.v: 將輸入的值左移 2 位並輸出。

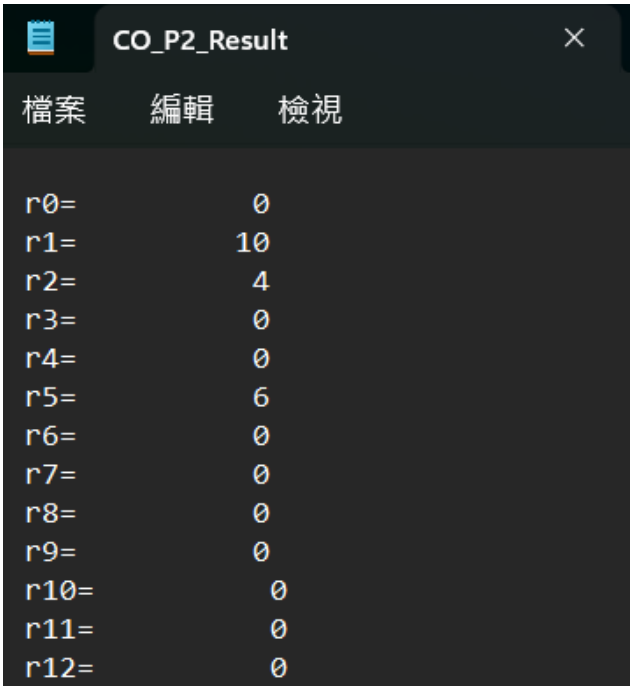
Sign_Extend.v: 藉由把 sign bit 延伸到第 17~32 位，將 16bit 的數值延伸到 32bit。

Simple_Single_CPU.v: 將全部的 module 統整再一起，完成一個簡易的 CPU。

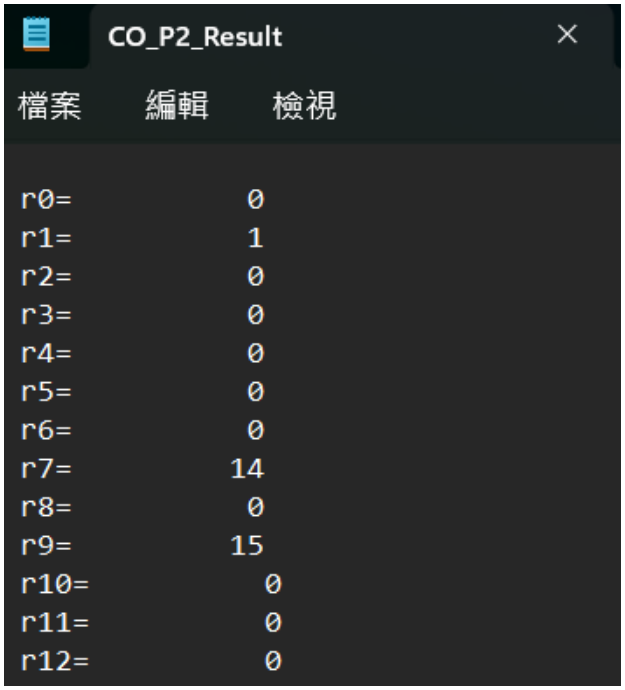
Finished part:



Test case 1:



Test case 2:



Problems you met and solutions:

一開始我不知道 decoder 中 ALU_op 要如何決定，於是就去 ALU_Ctrl 中看 ALU_op 的用途是什麼，後來才發現 ALU_op 沒有一定要什麼值，只要每個 operation 的 ALU_op 不一樣且跟 ALU_Ctrl 裡的 ALU_op 一致就可以了。

Summary:

完成了這次的 lab 後，讓我對 CPU 是如何將各個 module 統整在一起有了更深入的了解，也學到了電路要如何設計才能完成這些簡易的 MIPS 指令。