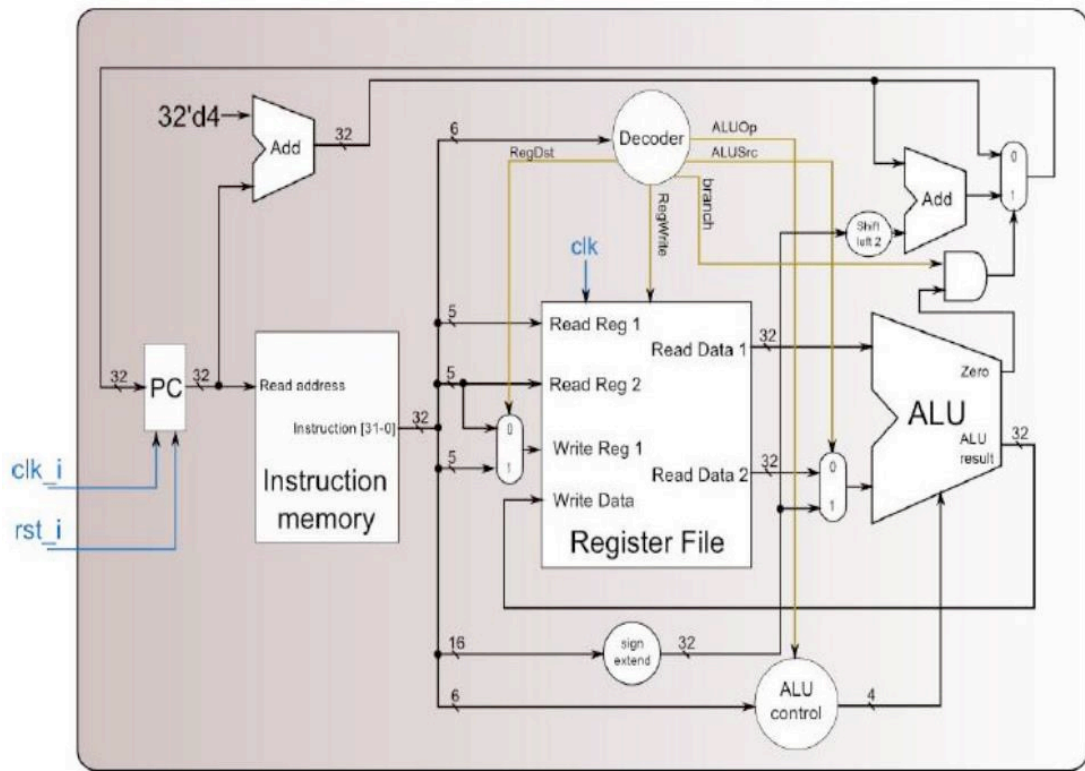


1. Architecture diagram:



Top module: Simple_Single_CPU

2. Detailed description of the implementation:

以下分成四個部分：PC，Instruction Memory，Register File，ALU。

一、PC：PC 有 3 個輸入與 1 兩個輸出，3 個輸入為 pc_in_i,clk_,rst_i，pc_in_i 從 2 to 1 MUX 輸入，並由 pc_out_o 輸出到 adder 進行加 4(PC+4)，clk_i 為同步器，當 rst_i 反相時，clk_並不會使 pc_in_i,pc_out_o 同步，否則，兩者進行同步改變(pc_out_o <= pc_in_i)。

二、Instruction Memory：Instruction Memory 有 1 個輸入和 6 個輸出。唯一的輸入來自 PC，而輸出則有 32bits 資料，依照 6 個輸出分為不同大小的部分，首先，6 個 bits[31:26]會輸出到 decoder，5bits[25:21]會輸出到 Read Reg1，5bits [20:16]會輸出到 Read Reg2 以及 2 to 1 MUX，5bits[15:11]輸出到 2 to 1 MUX，16bits[15:0]輸出到 sign extension，6 個 bits[5:0]輸出到 ALU_control。

三、Register File：Register File 有 7 個輸入，兩個輸出。其中 2 個輸入是從 instruction memory 來，1 個從 2 to 1 MUX 來，決定寫入資料的位址，1 個從 decoder 來，1 個從 ALU 的輸出而來。兩個輸出讀取暫存器，Read Data 1 輸出到 ALU，第 2 個輸出到 2 to 1 MUX。

四、ALU：ALU 有 3 個輸入和 2 個輸出，2 個輸入來自 Read Data1(src1_i)和

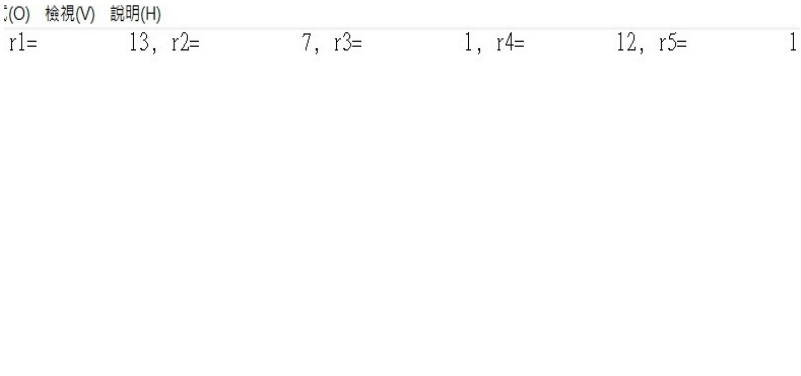
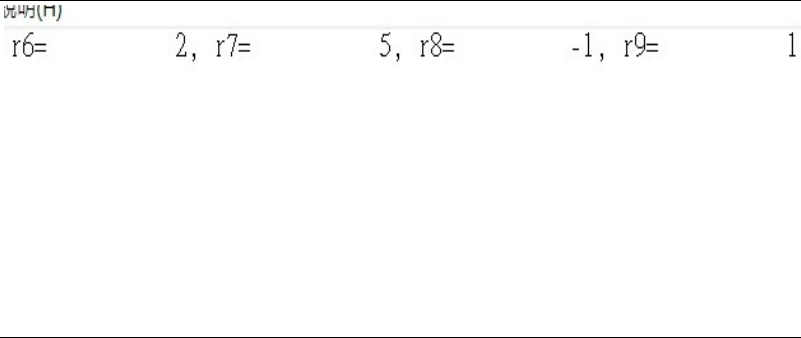
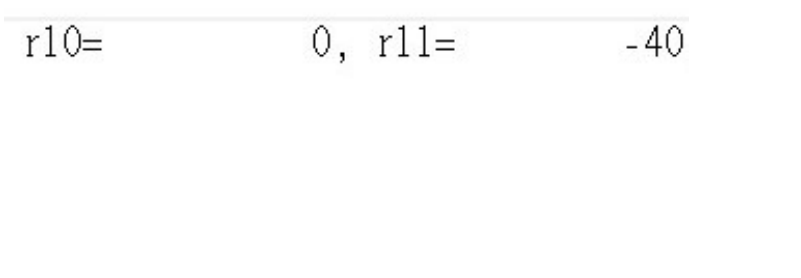
2 to 1 MUX(src2_i) , 1 個輸入來自 ALU control 。 1 個輸出是 32bits 的 ALU result(result_o) , 1 個輸出是 Zero(zero_o) 。

src1_i 給定 R 值

src2_i 值從 register file 和 sign-extension 兩者之間決定。

ctrl_i 來自 ALU control , 決定 ALU 的功能。

以下是 testbench 的結果截圖：

	測資結果截圖	說明
1		addi r1,r0,13 addi r2,r0,7 sltiu r3,r1,0xFFFF beq r3,r0,1 slt r4,r2,r1 and r5,r1,r4 subu r4,r1,r5
2		addi r6,r0,-2 addi r7,r0,5 or r8,r6,r7 addi r9,r0,-1 addi r6,r6,2 addu r9,r9,r6 beq r6,r0,-3
3		ori r10,r0,3 lui r11,-10 sra r11,r11,8 srav r11,r11,r10 addi r10,r10,-1 bne r10,r0,-3

3. Problems encountered and solutions:

一、釐清模組

Lab2 的 diagram 比 Lab1 得更複雜許多，因此一開始要釐清 inputs 跟 outputs 花了一點時間。主要花了一點時間研究 PC、Instruction memory、register file 等等的程式，釐清內部變數之間的關係。

此外，將每個部份串接起來發揮正確 instruction 的功用，也是一個難題。

4. Lesson learnt:

對於 **architecture** 的觀念更清楚，每條線路跟每個 **alu** 之間的關係也在做 **lab** 時才明白。