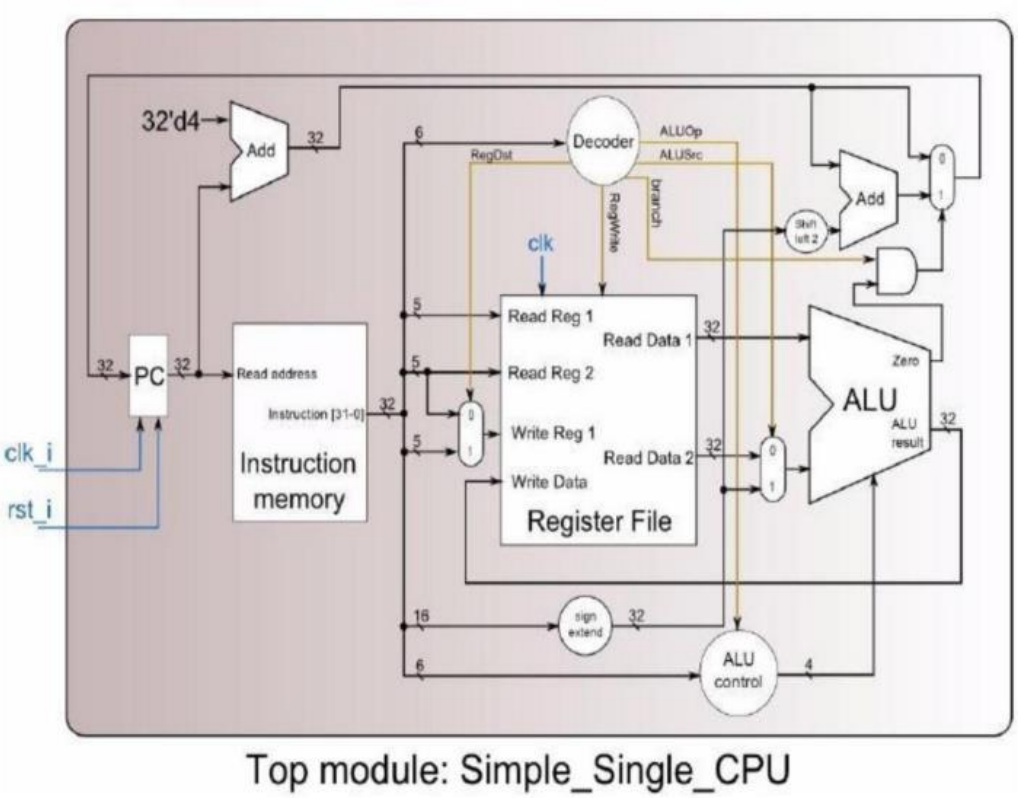
Computer Organization Lab2

Name: 龔祐萱

ID: 0713216

Architecture diagrams:



Top module: Simple_Single_CPU

Hardware module analysis:

1. **R-type instruction 執行過程**

   (1) PC 會指到 instruction memory 中的一個位置，並將該位置的指令輸出

   (2) 指令的 31-26 bit 會輸入 decoder 進行解碼，並輸出 control signal
   指令的 25-21 bit 會輸入 register file 作為 read reg1，並輸出該 reg 的資料
   指令的 20-16 bit 會輸入 register file 作為 read reg2，並輸出該 reg 的資料
   指令的 15-11 bit 會輸入 register file 作為 write reg
   指令的 5-0 bit 會輸入 ALU control，判斷此指令的 function field 是對應到哪種運算方式，並輸出至 ALU 中

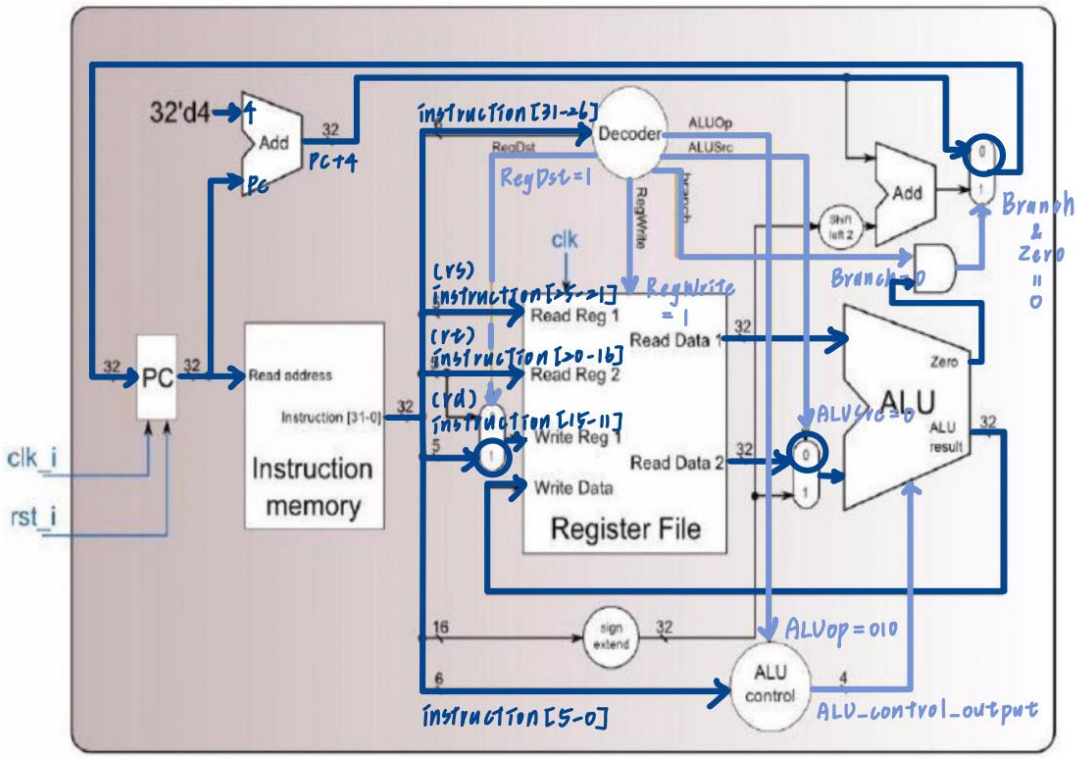
   (3) 從 reg 的 data 會輸入 ALU 進行運算，而運算方式是由 ALU _control_ output 判斷

   (4) 將 ALU 的運算結果送回 register file，並寫入 write reg 中

   (5) PC 會指到下一個要執行的指令(PC+4)

   (6) 以下是 R-type instruction 的 control signal 整理表格以及執行流程圖

RegDst	RegWrite	Branch	ALUSrc	ALUOp (3 bits)	ALU_control_output (4 bits)
1	1	0	0	010	add : 0010 sub : 0110 and : 0000 or : 0001 slt : 0111



2. addi / slti instruction 執行過程

(1) PC 會指到 instruction memory 中的一個位置，並將該位置的指令輸出

(2) 指令的 31-26 bit 會輸入 decoder 進行解碼，並輸出 control signal
指令的 25-21 bit 會輸入 register file 作為 read reg1，並輸出該 reg 的資料
指令的 20-16 bit 會輸入 register file 作為 write reg
指令的 15-0 bit 會輸入 sign extend 中，將 16 bits 的 constant 變成 32 bits 的 constant，並輸入至 ALU 執行運算

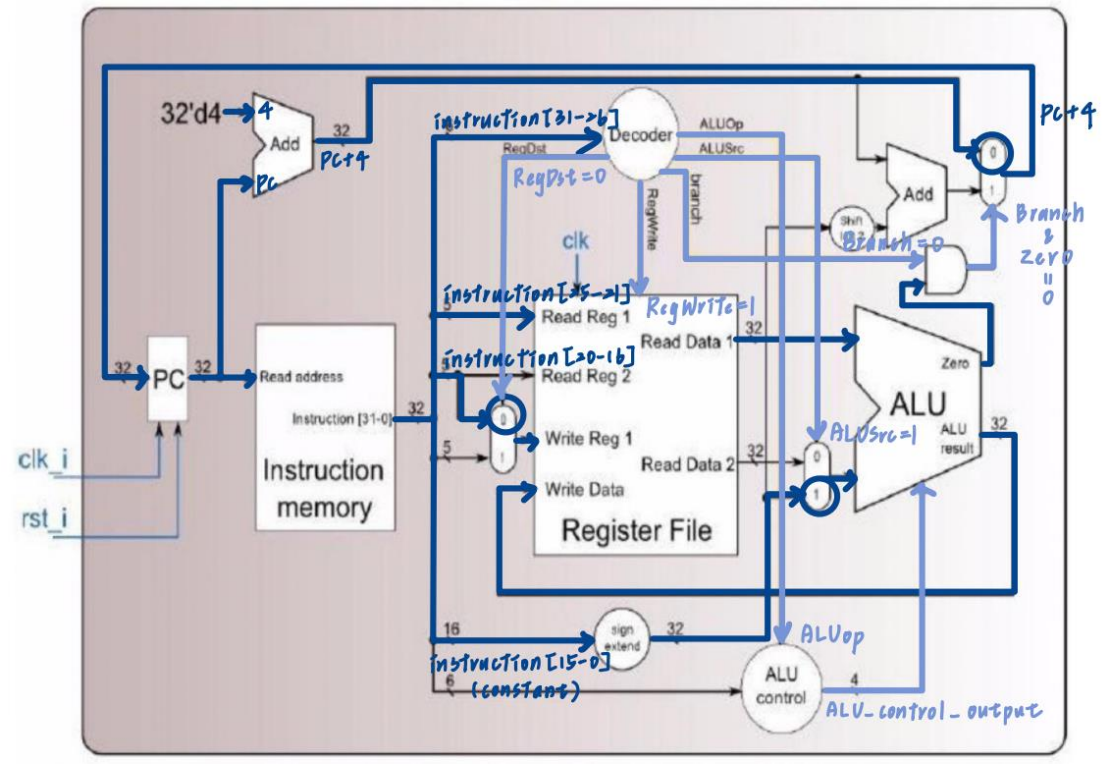
(3) 從 reg 的 data 以及執行過 sign extension 的 32 bits constant 會輸入 ALU 進行運算，而運算方式是由 ALU _control_ output 判斷

(4) 將 ALU 的運算結果送回 register file，並寫入 write reg 中

(5) PC 會指到下一個要執行的指令(PC+4)

(6) 以下是 addi / slti instruction 的 control signal 整理表格以及執行流程圖

RegDst	RegWrite	Branch	ALUSrc	ALUOp (3 bits)	ALU_control_output (4 bits)
0	1	0	1	addi : 000 slti : 011	addi : 1000 slti : 0101



3. beq instruction 執行過程

(1) PC 會指到 instruction memory 中的一個位置，並將該位置的指令輸出

(2) 指令的 31-26 bit 會輸入 decoder 進行解碼，並輸出 control signal
指令的 25-21 bit 會輸入 register file 作為 read reg1，並輸出該 reg 的資料
指令的 20-16 bit 會輸入 register file 作為 read reg2，並輸出該 reg 的資料
指令的 15-0 bit 會輸入 sign extend 中，將 16 bits 的 address 變成 32 bits 的 address，並輸入至 shifter 向左移 2 bits 變成 target address

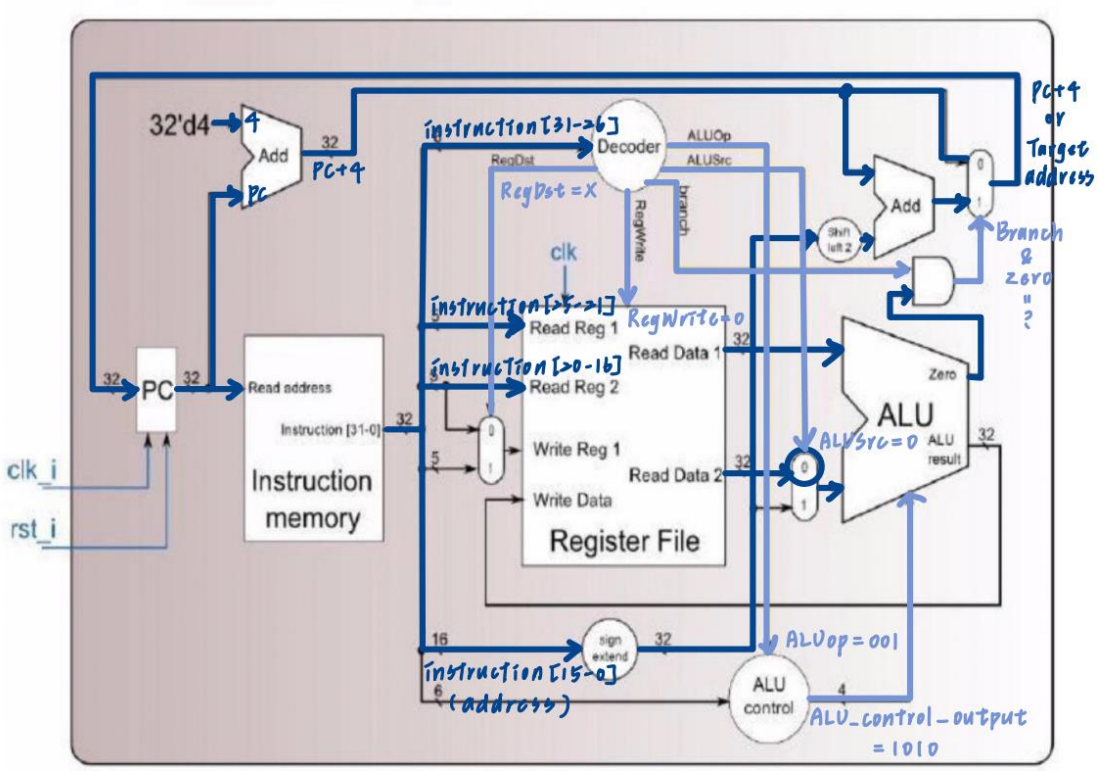
(3) 從 reg 的 data 會輸入 ALU 進行運算，而運算方式是由 ALU _control_ output 判斷

(4) ALU 會輸出 Zero 判斷兩個輸入 ALU 的值是否相等，Zero 和 Branch 會決定 PC 下一個指到的指令位置

(5) PC 會指到下一個要執行的指令(PC+4 or target address)

(6) 以下是 beq instruction 的 control signal 整理表格以及執行流程圖

RegDst	RegWrite	Branch	ALUSrc	ALUOp (3 bits)	ALU_control_output (4 bits)
x	0	1	0	001	1010



4. 優缺點

可以透過 control signal 去執行不同的指令，不需要更複雜的硬體去實作，且使用 2 level 的 control 可以先利用 opcode 區分 R-type、addi / slti 和 beq 的指令，再利用指令中的 function field 去判斷指令為 R-type 中的哪種運算，實作上會比將 opcode 和 function field 放在一起輸入 control unit 判斷更有效率。

Finished part:

以下是測資的執行指令、指令執行過程、執行結果以及程式執行結果截圖:

1. Case 1

執行指令	指令執行過程	執行結果	程式執行結果截圖
(1) addi r1, r0, 10 (2) addi r2, r0, 4 (3) slt r3, r1, r2 (4) beq r3, r0, 1 (5) add r4, r1, r2 (6) sub r5, r1, r2	(1) r1=r0+10=0+10=10 (2) r2=r0+4=0+4=4 (3) (r1=10)≮(r2=4), r3=0 (4) (r3=0)=(r0=0) goto PC+4 (5) r5=r1-r2=10-4=6	r1 = 10 r2 = 4 r3 = 0 r4 = 0 r5 = 6	# run 1000ns 2 r0= 0 r1= 10 r2= 4 r3= 0 r4= 0 r5= 6 r6= 0 r7= 0 r8= 0 r9= 0 r10= 0 r11= 0 r12= 0

2. Case 2

執行指令	指令執行過程	執行結果	程式執行結果截圖
(1) addi r6, r0, 2 (2) addi r7, r0, 14 (3) and r8, r6, r7 (4) or r9, r6, r7 (5) addi r6, r6, -1 (6) slti r1, r6, 1 (7) beq r1, r0, -5	(1) r6=r0+2=0+2=2 (2) r7=r0+14=0+14=14 (3) r8=r6 AND r7=2 (4) r9=r6 OR r7=14 (5) r6=r6-1=2-1=1 (6) (r6=1)≮1, r1=0 (7) (r1=0)=(r0=0) goto PC-20 (8) r7=r0+14=0+14=14 (9) r8=r6 AND r7 = 0 (10) r9=r6 OR r7 = 15 (11) r6=r6-1=1-1=0	r1 = 1 r6 = 0 r7 = 14 r8 = 0 r9 = 15	# run 1000ns 2 r0= 0 r1= 1 r2= 0 r3= 0 r4= 0 r5= 0 r6= 0 r7= 14 r8= 0 r9= 15 r10= 0 r11= 0 r12= 0

	(12) (r6=0)<1, r1=1 (13) (r1=1)≠(r0=0) , no branch		

3. Case 3

執行指令	指令執行過程	執行結果	程式執行結果截圖
(1) addi r1, r0, 2 (2) addi r2, r0, 4 (3) slt r3, r1, r2 (4) beq r3, r0, 1 (5) add r4, r1, r2 (6) sub r5, r1, r2	(1) r1=r0+10=0+2=2 (2) r2=r0+4=0+4=4 (3) (r1=2)<(r2=4), r3=1 (4) (r3=1) ≠ (r0=0), no branch (5) r4=r1+r2=2+4=6 (6) r5=r1-r2=2-4=-2	r1 = 2 r2 = 4 r3 = 1 r4 = 6 r5 = -2	# run 1000ns 2 r0= 0 r1= 2 r2= 4 r3= 1 r4= 6 r5= -2 r6= 0 r7= 0 r8= 0 r9= 0 r10= 0 r11= 0 r12= 0

Problems you met and solutions:

這次寫程式的過程較順利，但還是有遇到一些小問題，就是在執行測資時忘記考慮 branch 指令的因素，所以導致程式在判讀測資時出現錯誤，後來將程式進行微調之後，測資就可以跑出正確的結果了。

Summary:

透過這次 lab2 的實作，使我更加了解 simple single CPU 的架構以及其運作的方式，雖然在設計電路的過程碰到了一些小問題，但也透過網路上的參考資料以及再三檢查自己的程式內容後，找到程式的問題並解決問題。