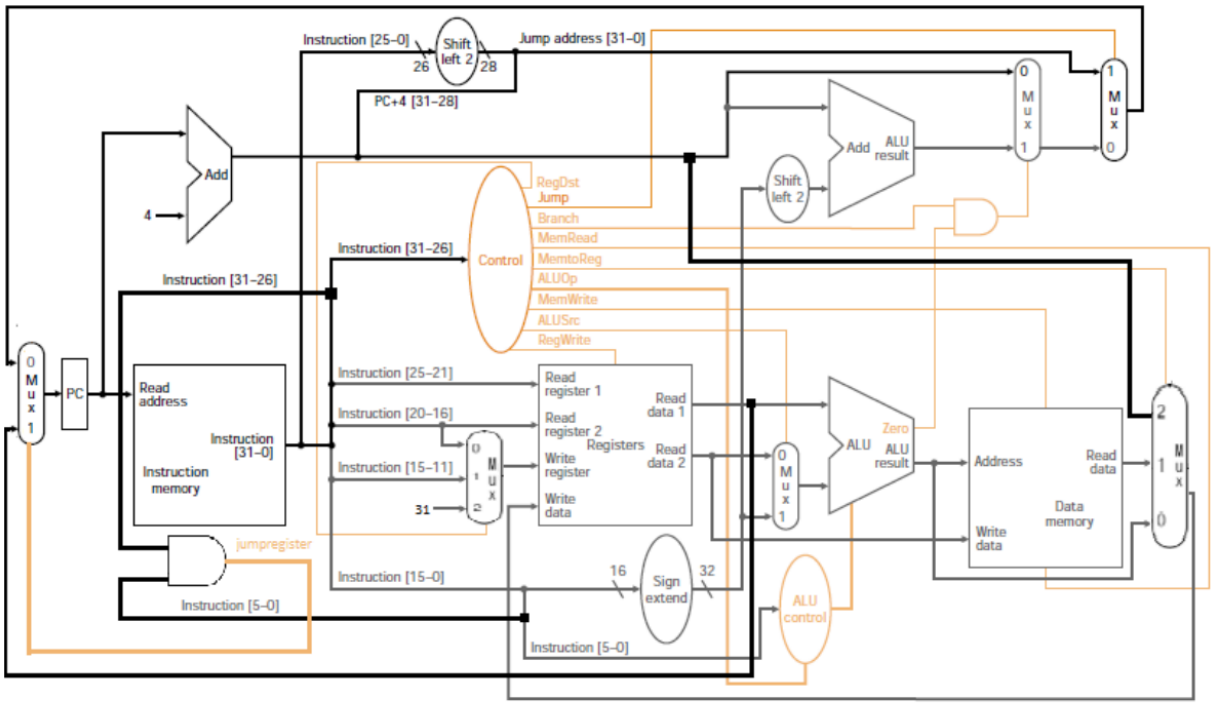
Computer Organization Lab3

Name: 吳文心

ID: 109550022

Architecture diagrams:



Hardware module analysis:

(explain how the design work and its pros and cons)

從 PC counter 開始描述計算 PC 相關的各種元件：

1. PC counter 連出去的 adder 是計算 PC+4 (變數名稱：pc_next)，即沒有各種 jump 的情況下 PC 的變化。
2. 再連出去的下一個 adder 是計算 branch 的 address (變數名稱：pc_branch)，並用一個 2 to 1 的 MUX 去選擇接下來要用 pc_next 還是 pc_branch，即要繼續走下去還是要 branch 去別的地方。
3. 接下來的 MUX 會選擇要 jump 還是要用上一步決定出的 address，jump address 的部分我直接把所含的 data concatenate 在一起放 MUX 的輸入。
4. 最後是 diagram 最左邊的 MUX，這個 MUX 選擇回到$ra 還是用上一步決定的

address，這個 MUX 的控制訊號 jump_reg，當 instruction 的 operation code 全為 0 且 function field 為 001000 時 jump_reg 為 1，除此之外都為 0。

再來是關於$ra 的存取：

1. 存的部分，在 register 的 Write register 的 input 使用了 3 to 1 的 MUX，當此 MUX 的 control signal 是 2 時，會選擇 r31 作為 write register，也就是寫入到 $ra，至於要存入的資料，在 Data memory 後面使用了一個 3 to 1 的 MUX，當此 MUX 的 control signal 是 2 時，會選擇 pc_next 作為 data 傳回到 register 的 Write data，也就是當成是 return 回來之後應該繼續執行的下一個指令。
2. 取的部分，jal 指令傳入$ra 的 address，由 Register 讀取並輸出於 Read data 1，Read data 1 即上述 PC counter 的解釋中第 4 項的 MUX 的 input 之一，當 control signal 為 1 時就會選擇$ra 的值。

Finished part:

(show the screenshot of the simulation result and waveform, and explain it)

test 1:

```
Data Memory =    1,    2,    0,    0,    0,    0,    0,    0
Data Memory =    0,    0,    0,    0,    0,    0,    0,    0
Data Memory =    0,    0,    0,    0,    0,    0,    0,    0
Data Memory =    0,    0,    0,    0,    0,    0,    0,    0
Registers
R0 =    0, R1 =    1, R2 =    2, R3 =    3, R4 =    4, R5 =    5, R6 =    1, R7 =    2
R8 =    4, R9 =    2, R10 =    0, R11 =    0, R12 =    0, R13 =    0, R14 =    0, R15 =    0
R16 =    0, R17 =    0, R18 =    0, R19 =    0, R20 =    0, R21 =    0, R22 =    0, R23 =    0
R24 =    0, R25 =    0, R26 =    0, R27 =    0, R28 =    0, R29 =  128, R30 =    0, R31 =    0
```

test 2:

```
Data Memory =    0,    0,    0,    0,    0,    0,    0,    0
Data Memory =    0,    0,    0,    0,    0,    0,    0,    0
Data Memory =    0,    0,    0,    0,   68,    2,    1,   68
Data Memory =    2,    1,   68,    4,    3,   16,    0,    0
Registers
R0 =    0, R1 =    0, R2 =    5, R3 =    0, R4 =    0, R5 =    0, R6 =    0, R7 =    0
R8 =    0, R9 =    1, R10 =    0, R11 =    0, R12 =    0, R13 =    0, R14 =    0, R15 =    0
R16 =    0, R17 =    0, R18 =    0, R19 =    0, R20 =    0, R21 =    0, R22 =    0, R23 =    0
R24 =    0, R25 =    0, R26 =    0, R27 =    0, R28 =    0, R29 =  128, R30 =    0, R31 =   16
```

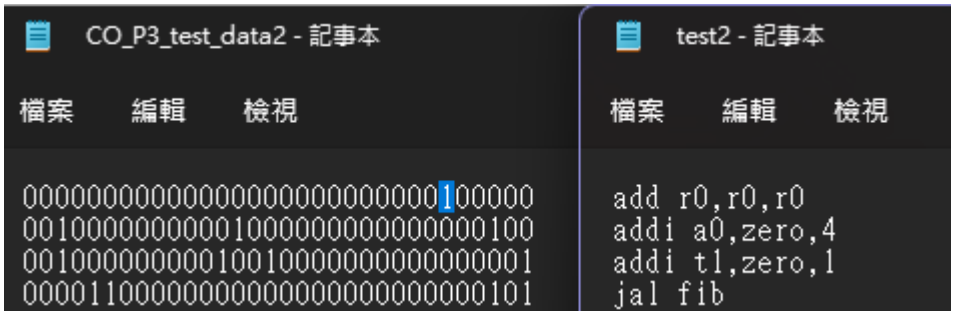
my test: (測了所有 test 1,2 沒有包含的指令)

00100000000000010000000000000111	addi	r1, r0, 7	r1 = 7
00100000000000100000000000000010	addi	r2, r0, 2	r2 = 2
00000000001000100001100000100010	sub	r3, r1, r2	r3 = 5
00000000001000100010000000100100	and	r4, r1, r2	r4 = 2
00000000001000100010100000100101	or	r5, r1, r2	r5 = 7
00000000001000100011000000101010	slt	r6, r1, r2	r6 = 0
00101000001001110000000000001010	slti	r7, r1, 10	r7 = 1

```
Registers
R0 =    0, R1 =    7, R2 =    2, R3 =    5, R4 =    2, R5 =    7, R6 =    0, R7 =    1
R8 =    0, R9 =    0, R10 =    0, R11 =    0, R12 =    0, R13 =    0, R14 =    0, R15 =    0
R16 =    0, R17 =    0, R18 =    0, R19 =    0, R20 =    0, R21 =    0, R22 =    0, R23 =    0
```

Problems you met and solutions:

在測試 test 1 的時候一切順利，有了上一個 lab 的基礎，這應該是有史以來最順利的一次 verilog 作業了，但在 test 2 的時候，result 一直不對，後來仔細比對了一下測資發現測資怪怪的：



框起來的地方原本是 0，但 add 的 function field 應該會是 100000，改成 1 之後輸出就都正常了。

Summary:

在寫上一個 lab 的時候就有點困惑為什麼沒有 lw 跟 sw，原來都在這次 lab 裡～我在寫 verilog 的路上又向前邁進了一步呢～