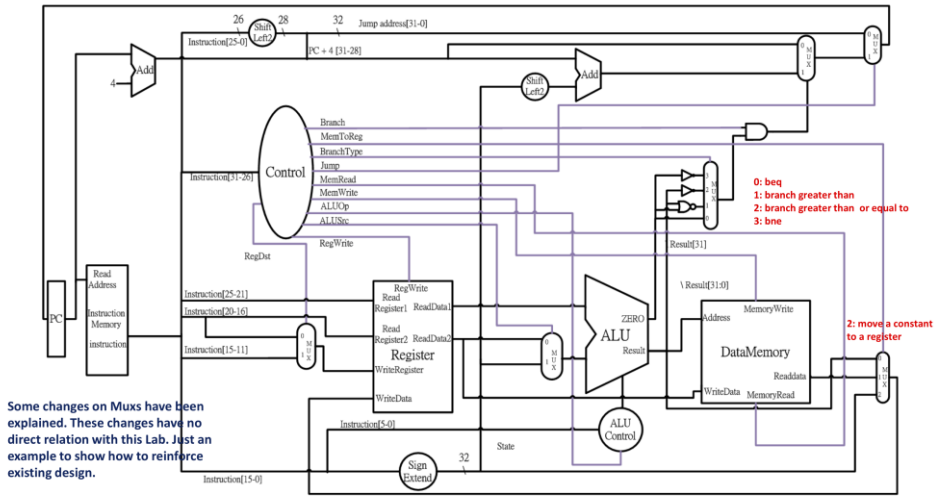
Computer Organization

0511105 李頤

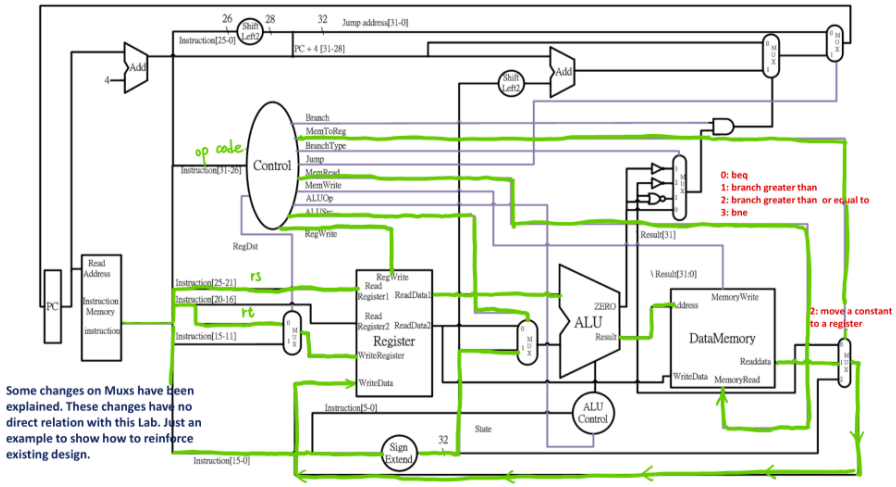
Architecture diagrams:

基本上架構使用用 pdf 上的圖片 ，jal 和 jr 的設計在 hardware module analysis 裡面

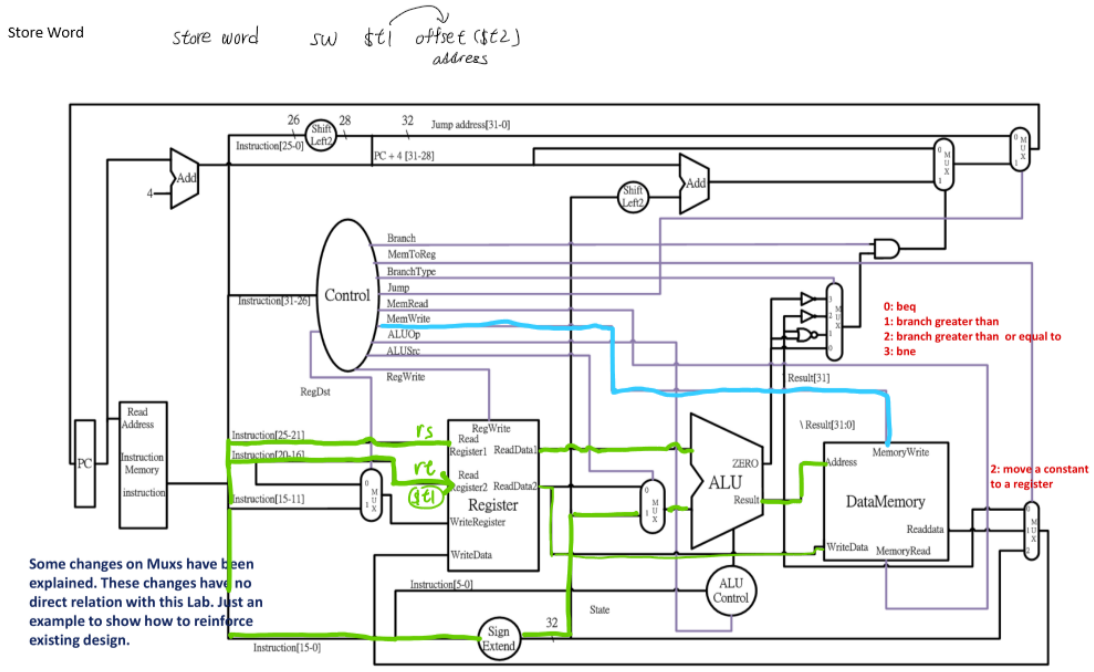


Hardware module analysis:

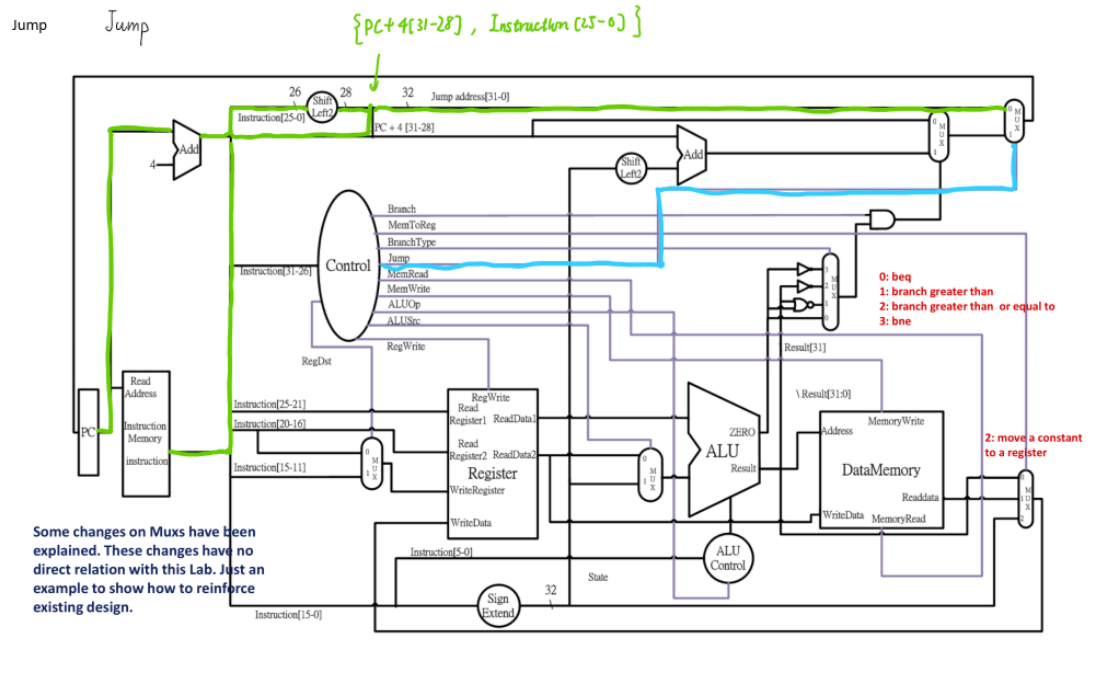
1. Load Word (lw)



Store Word



Jump

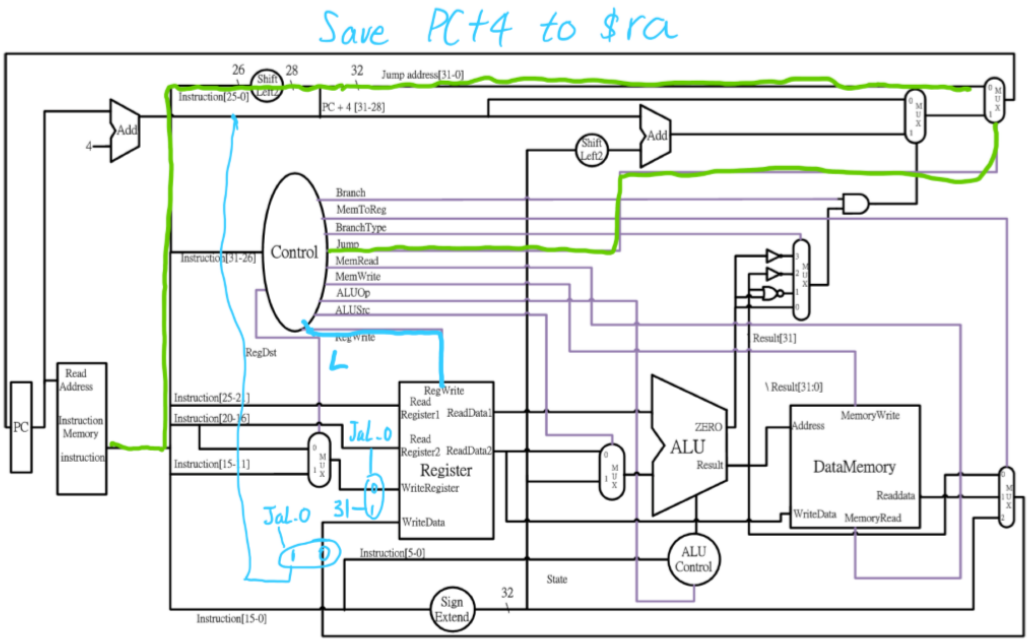


address 的部分我用用以下方方法實現

{curr_pc_plus_4[31:28], rs_addr, rt_addr, rd_addr, shamt, func, 2'b00}

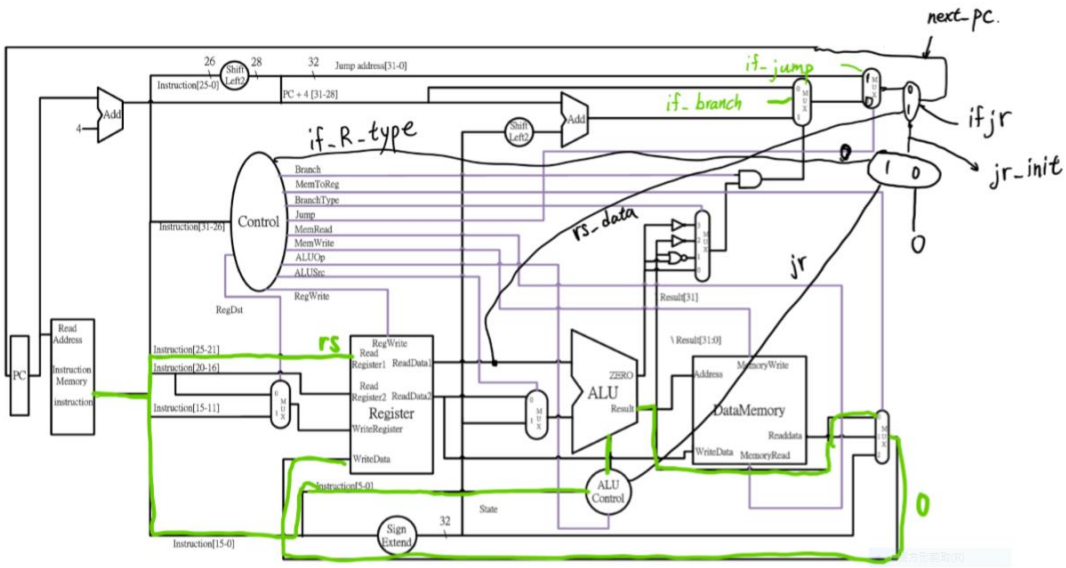
合併輸出成：（pc＋4 前四碼, jump instruction 的 26 位 address, 00）

4.Jump and Link (jal)



我多加了了兩個 MUX，Decoder 也多加了了 Jal_o 輸出控制
Jal_o 為 1 時，WriteRegister 值為 31，WriteData 則是把 PC+4 從上面抓下來
完成把 PC+4 存進 $ra 的動作
Jal_o 為 0 時，不影響其他指令輸出

5.Jump Return (jr)



因為 jr 是 R-type 的關係，這個指令花了了我最多時間(想＋debug)

實現方方式：

加了了兩個 MUX，decoder 多了了 jr_o 輸出，ALU Control 多了了 jr 輸出

設計邏輯：

如果是 jr 指令時，ALU Control 送出 jr = 1，因為是 R-type，if_R_type = 1
選擇 jr = 1 進入 jr_init。

jr_init = 1 時，會選擇 ReadData1 的訊號也就是 return address 輸入至至 PC。

如果不是 R-type 指令，jr_init = 0，PC 輸出維持原路路徑。
如果是 R-type，但不是 jr 指令時，jr_init = jr = 0，輸出仍維持不變。

遇到勿問題：

因為還是 R-type，ALU 還是有輸出結果寫入 Register，這時 rd = 0
$zero 就會被覆寫掉，變成非零值。
解決辦法是把 rs 跟 rt AND起來因為 rt 是零，輸出等於零寫入 rd
不改變$zero 的值

Summary

忘記把 simulation time 調大，還以為是 code 寫錯