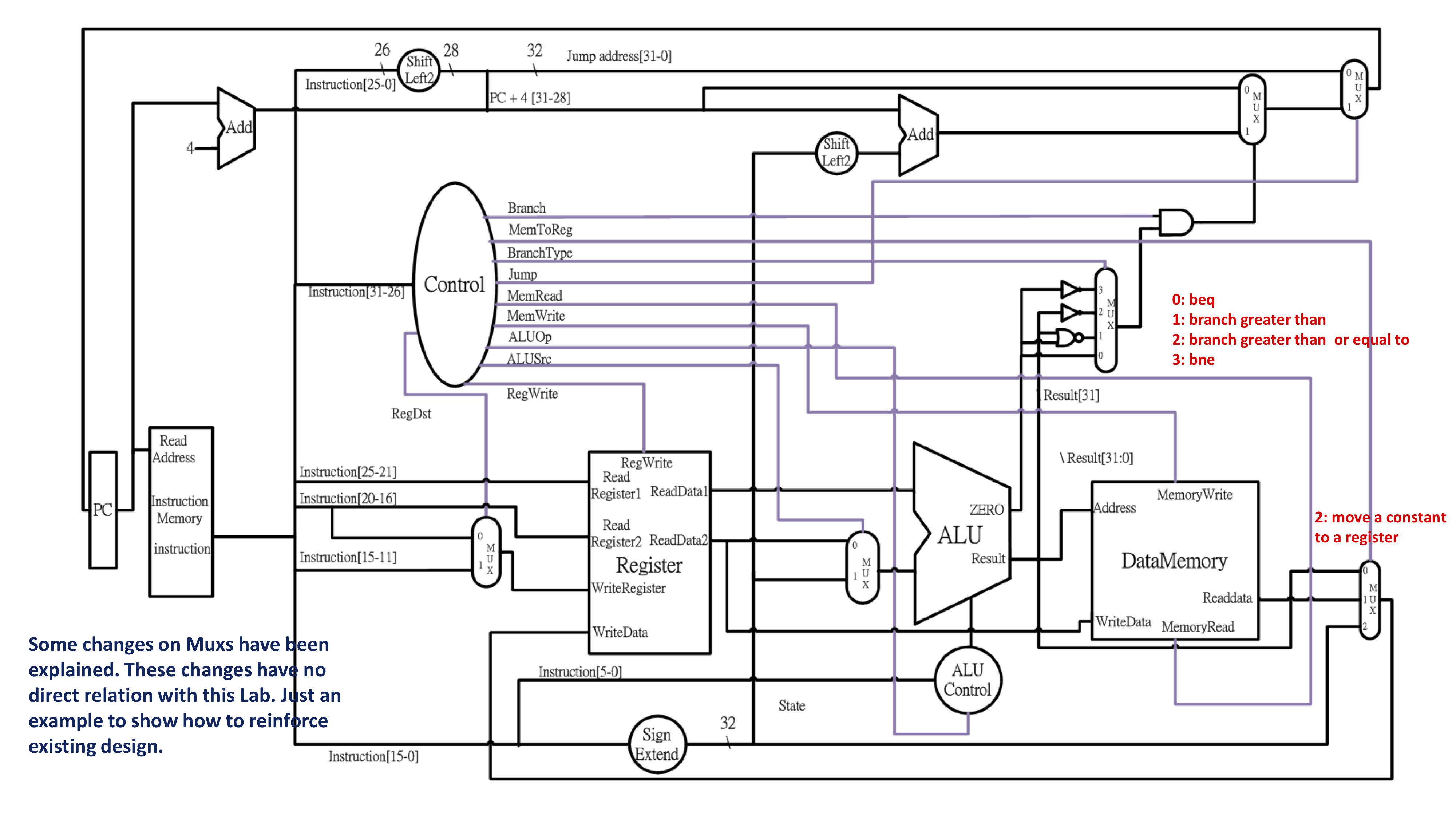
**Computer Organization**

0511105 李頤

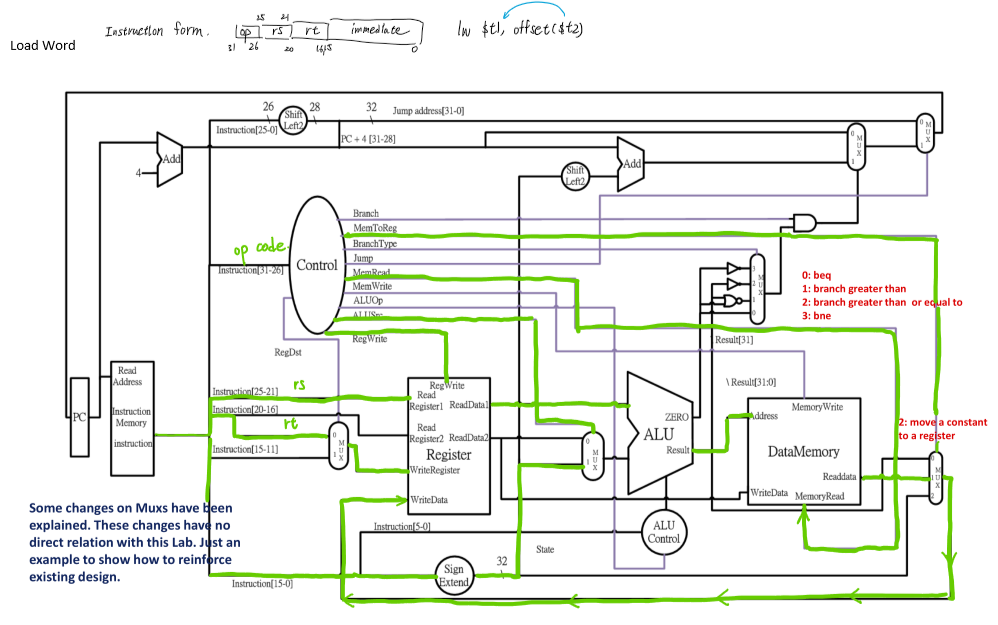
**Architecture diagrams:**

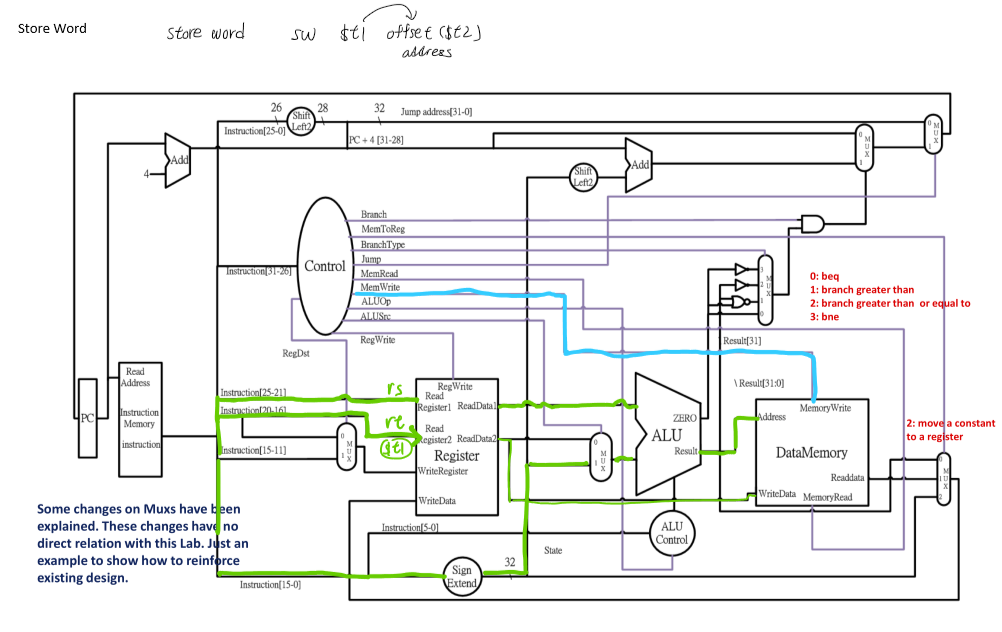
基本上架構使⽤用pdf上的圖片，jal和jr的設計在 hardware module analysis 裡面



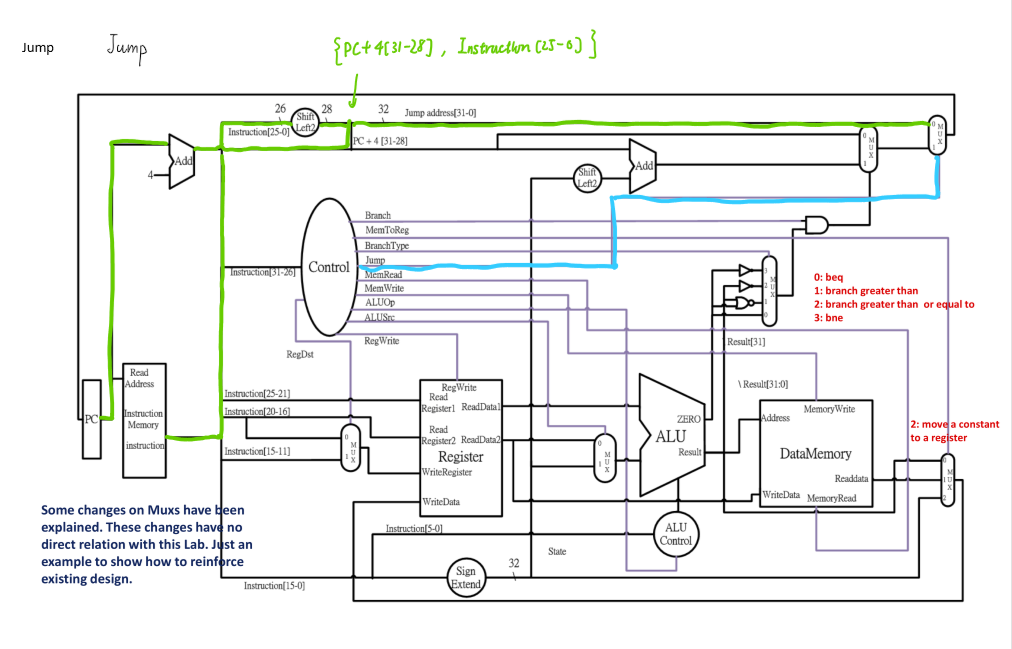
**Hardware module analysis:**

1. Load Word (lw)





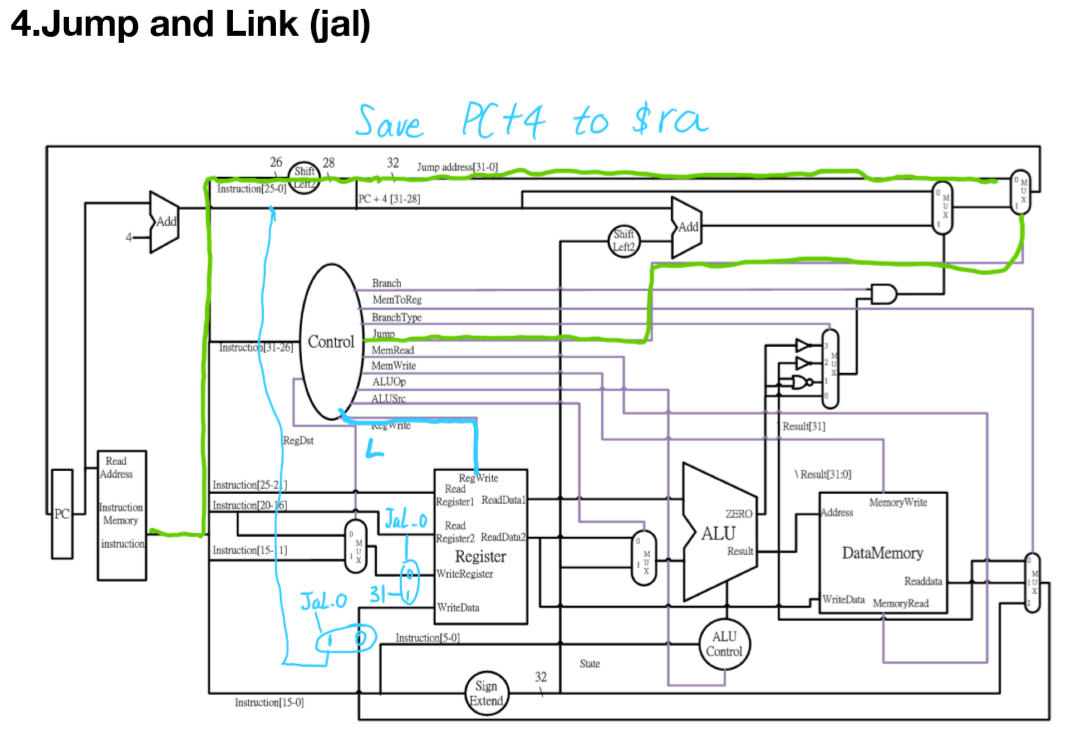
Jump



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）

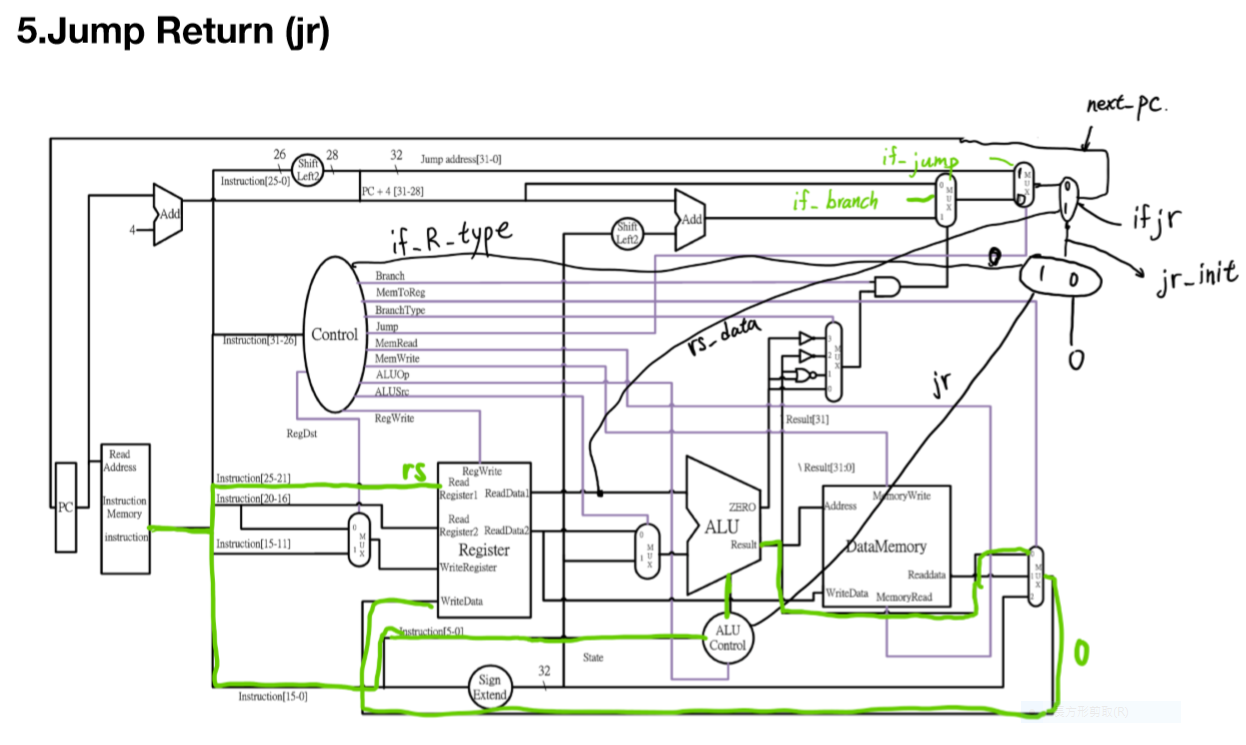


我多加了了兩個MUX，Decoder也多加了了Jal\_o輸出控制

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



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ㄉ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