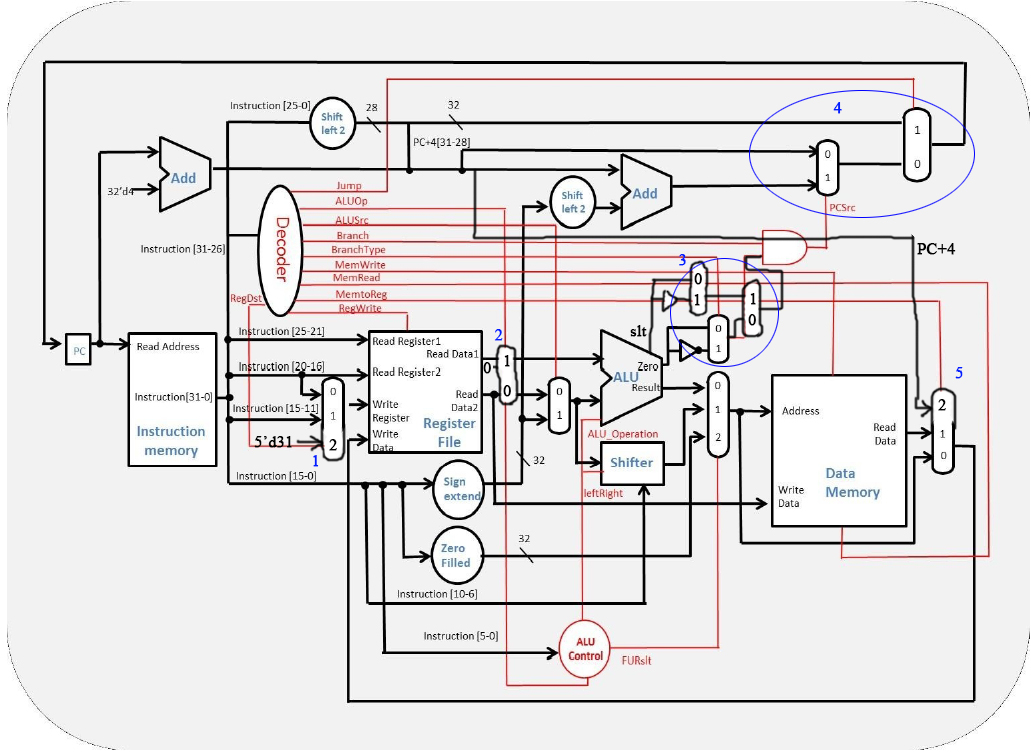
**Computer Organization**

**Architecture diagrams:**

****

**Hardware module analysis:**

**1.新增的硬體**

1 : 預設5’d31的暫存器位置用來讓jar寫入$r31

2 : 預設rt = 0，用來給blt,bgez判斷的0(防止r0被改掉時功能錯誤)

3 : 用2個flag(zero,slt)來判斷，再用一個mux來選要用哪種flag的結果做branch判斷

4 : 左邊的mux是用來判斷在bne,beq時有沒有要branch，如果PCsrc=1代表符合條件需要branch。右邊的mux是用來判斷是直接jump還是條件判斷後branch

5 : 用於jar將pc+4傳到Reg File寫入

**2.新增的control signal**

Select\_Zero\_branch = rs=rt時為1，rs!=rt時為0

Select\_Slt\_branch = rs<rt時為1，rs>=rt時為0

Select\_branchType : bne,beq用==判斷的指令為0，blt,bgez等用<或>=判斷的指令為1

RegDst : 新增2用來選擇5’d31

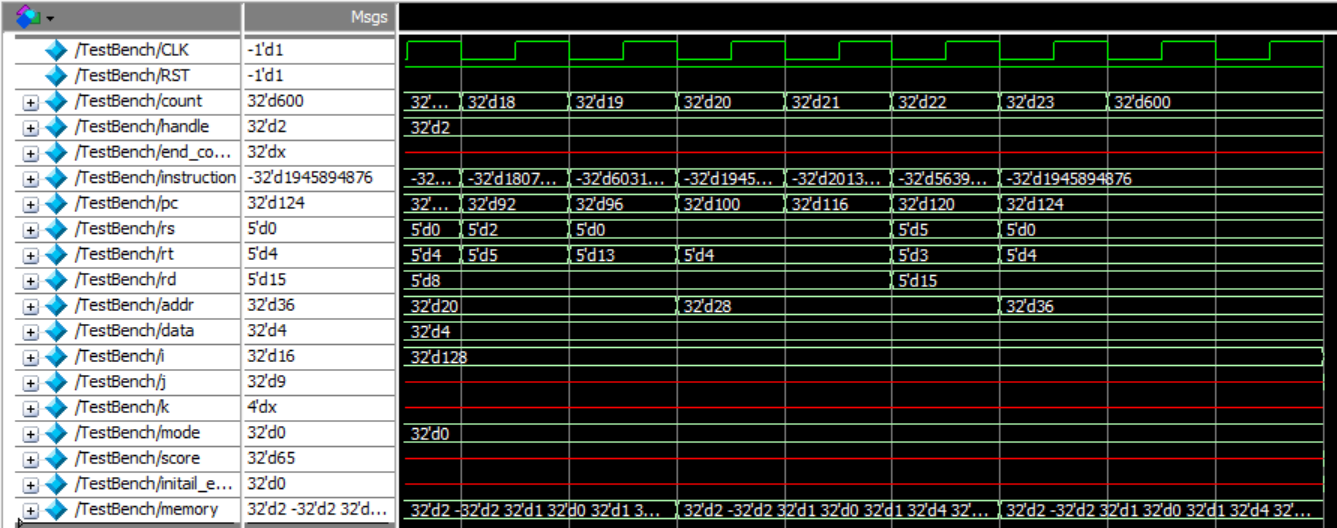
MemtoReg : Rtype指令輸出=0，Ltype指令輸出=1，PC+4輸出=2(jar)

MemRead : 只有lw=1，其餘為0

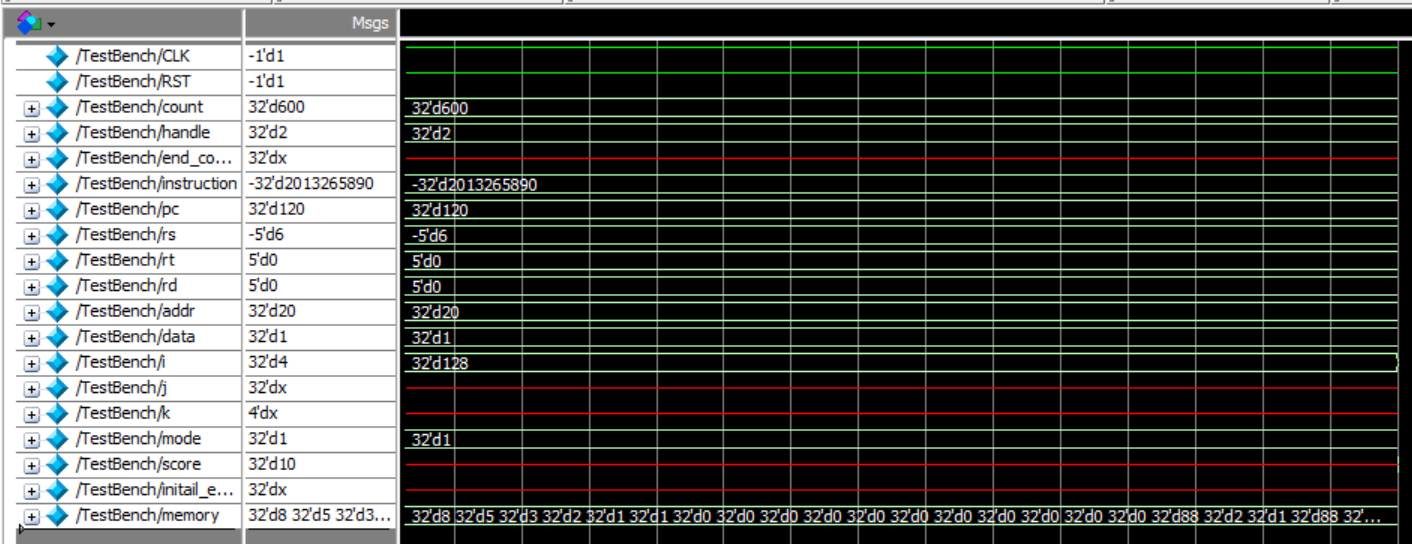
MemWrite : 只有sw=1，其餘為0

**Finished part:**

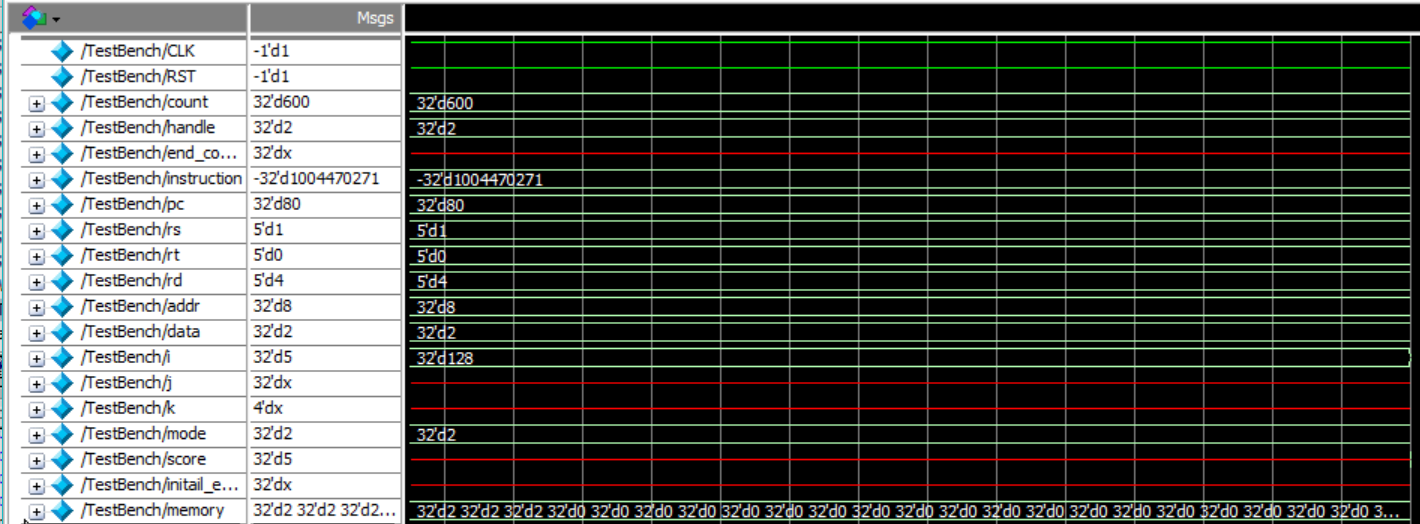
**Test data 1**



**Test data 2**



**Test data 2\_2**



**Problems you met and solutions:**

經過上次的Lab，這次上手就比較快了，但還是在一些小地方容易出錯。

像是我的decoder為了要做出jr的功能所以把輸入訊號多加了一個instr\_func\_i，結果always@()裡面忘了加進去判斷，導致每次做到add的時候都會直接jump，找個很久才發現錯誤。

**Summary:**

經過這次的lab我對指令的架構更加了解了，也更熟悉branch指令的執行過程。