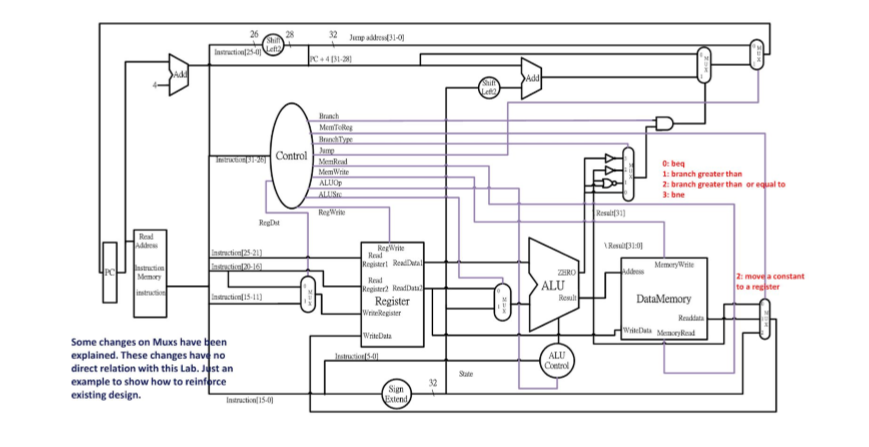
**Computer Organization**

**Architecture diagrams:**



**Hardware module analysis:**

**Program Counter：**

目前指令的位置。

**Decoder：**

判斷目前所要執行的operation為何，輸出RegWrite\_o/ ALU\_op\_o/ ALUSrc\_o/ RegDst\_o/ Branch\_o/ MemToReg\_o/ Jump\_o/ MemRead\_o/ MemWrite\_o等控制訊號。

**Instruction Memory：**

將目前要執行的指令存到Instruction Memory。

**Register File：**

依照R-type/I-type來讀取1-2個register檔案內容並輸出做運算。

**ALU：**

依據收到ALU Control的訊號，將輸入的值作相對應的operation。

(這次相較上次增加了 lw / sw / jal)

**Adder：**

**Adder1**

將pc的指向下一個指令(pc+4)

**Adder2**

依據分支給予的偏移量加上原本pc的值來更新pc的值。

**ALU Control：**

控制ALU所要做的operation。

**Shift Left 2：**

因為instruction的單位為word(32 bit)，所以要向左移2 bit將單位轉為以byte(8 bit)來計算

**Sign Extend：**

像是immediate等的值在Iinstruction中只存放16 bit，因此需要透過sign extend為32 bit才能當作輸入。

**MUX**

**MUX\_2to1**

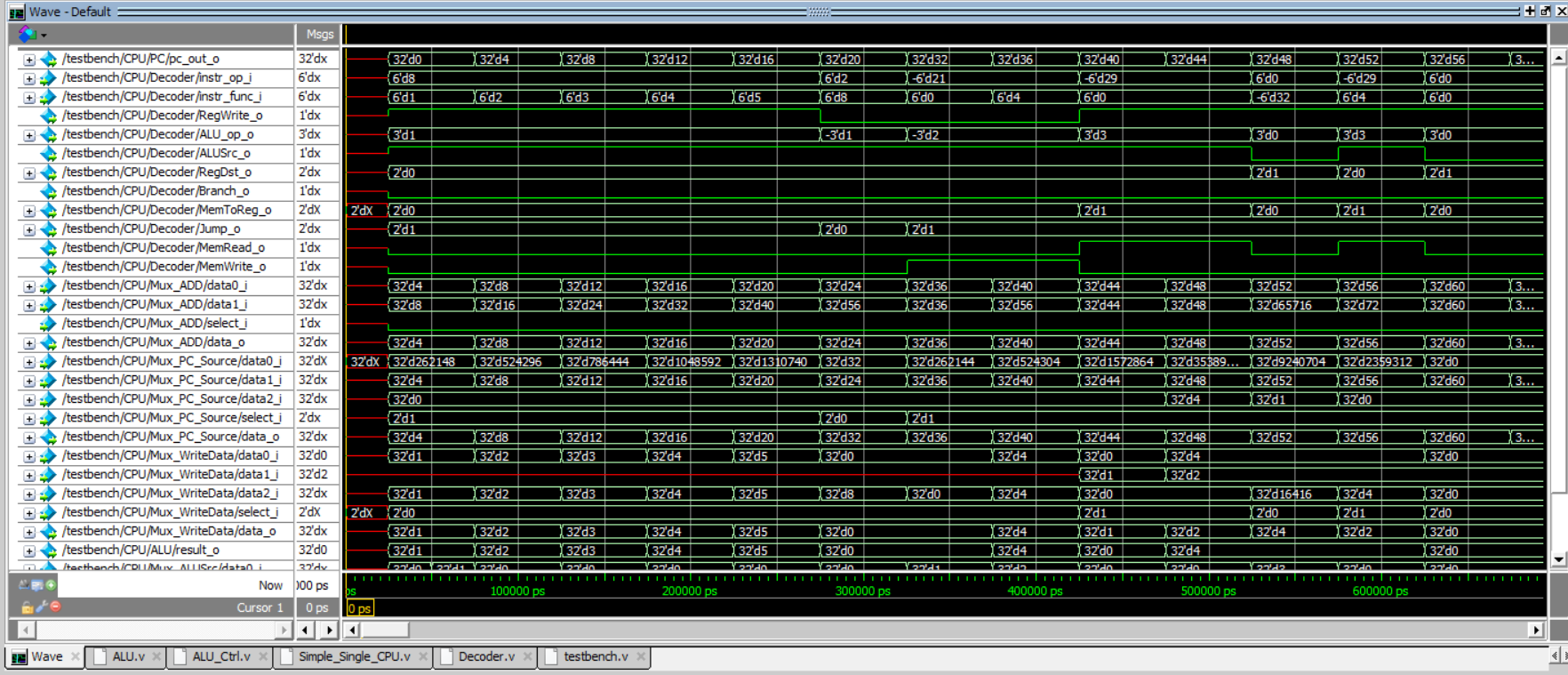
1. Mux\_ALUSrc (choose which data (reg\_read\_data2/ sign\_ext\_out) be written into reg)
2. Mux\_ADD (choose which data (add1\_out/ add2\_out) be written into Mux\_PC\_Source)

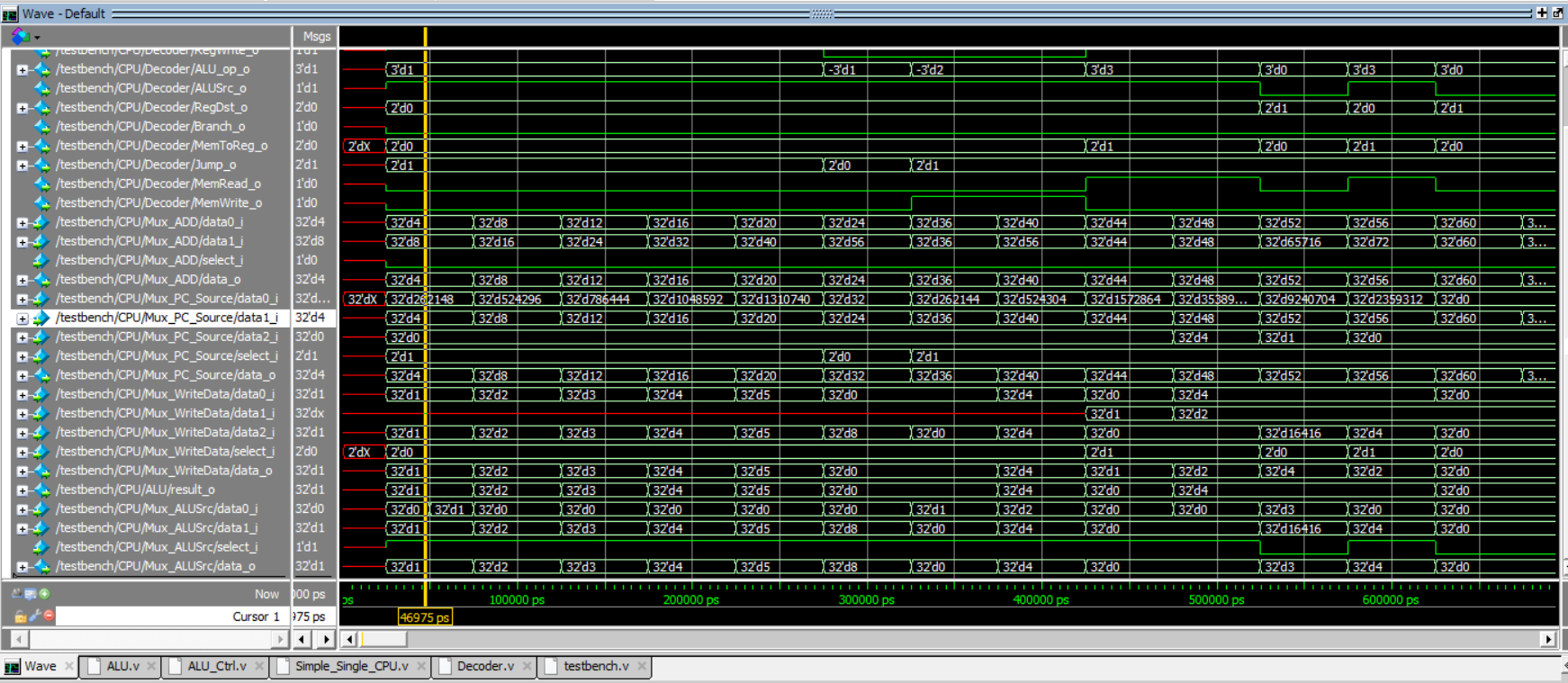
**MUX\_3to1**

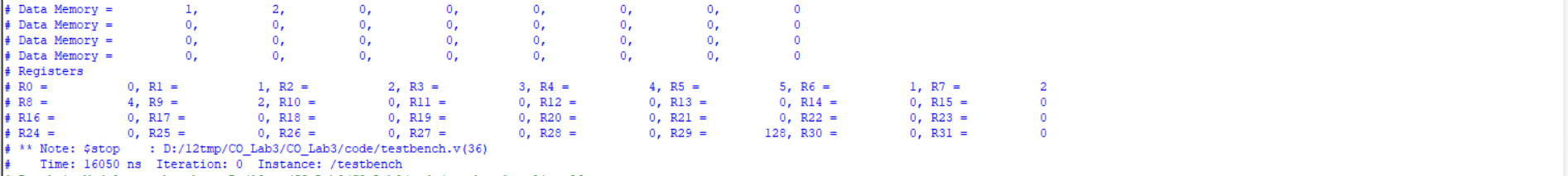
1. Mux\_Write\_Reg (data written to reg)
2. Mux\_PC\_Source (update pc address)
3. Mux\_WriteData (data written into reg)

**Finished part:**

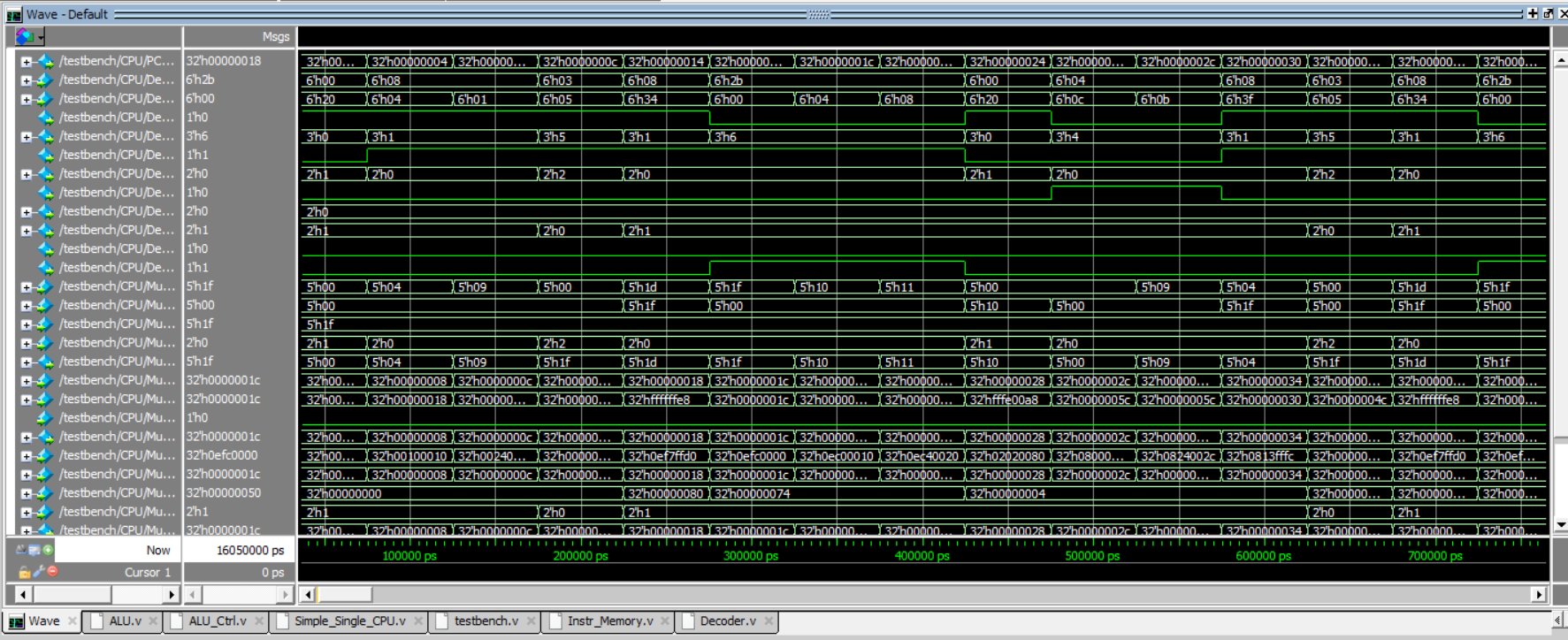
**Test1**

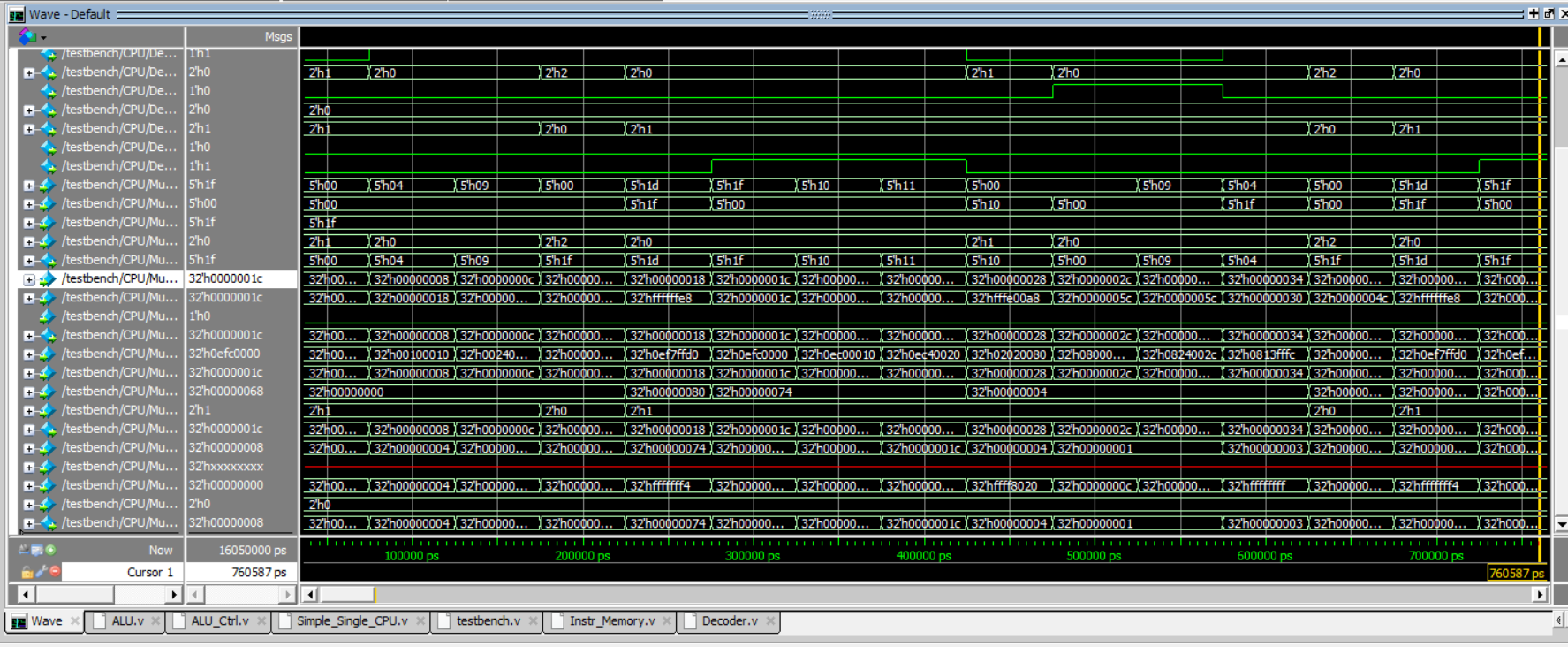


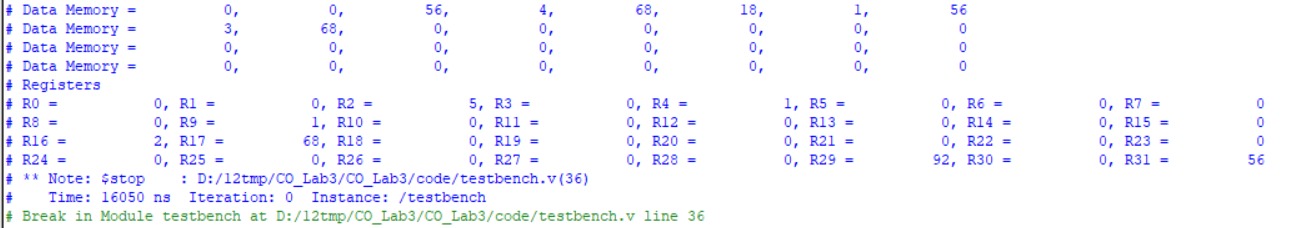




**Test2**







**Problems you met and solutions:**

1. **SW / LW的地址錯誤**

Sol:在ALU中將SW/LW的的指加上立即值(offset)，否則存到的地址都為同一個。

1. **Jal存值方式錯誤，最後ra跳回的指令錯誤**

Sol:透過將Mux\_Write\_Reg改為MUX\_3to1，當收到jal時RegDst設為2，輸入RegDst為31。

1. **在test2時reg內存值有誤**

Sol：由於我是在ALU中處理jal及jr的地址存值及其offest，而游於藝開始我將傳入ALU中的的記憶體位置傳成RS的讀取範圍(instr\_memory\_out[25:21])，之後將傳入值改為reg\_read\_data1就沒問題了。

**Summary:**

這次LAB相較於前兩次寫比較慢的原因主要都是指令之間沒有傳好而致，例如：遇到jal指令後要將的指存起來、該如何存，遇到jr指令時又要如何將reg中的值讀出來，還有在遇到sw/lw時的指的offset要如何去加，都是這次LAB主要遇到的問題。