**CO project 3 – Simple Single Cycle CPU**

*ID: 0340249*

*Name: 孫聖*

1. **Introduction**

**這個實驗中我們要進一步完善之前所完成的CPU，首先要增加memory unit實現lw和sw的功能，之後還要實現幾個跳轉的指令。**

1. **Procedure**

**首先，我根據架構圖填充了DM中每個port的訊息，從而實現lw和sw。**

**在Jump指令中我用了MUX\_JJr來選擇下次PC的值，用2號端口作Jump地址的輸入。**

**Bgt指令，這個指令執行時ALU執行減法，MUX\_BranchType輸入為端口1，如果結果最高位為0且最後結果不為0的話即跳轉。**

**Bgez指令，這個指令rt處並不是取寄存器的值而是直接取立即值，因此我添加了MUX\_ReadData2來選擇是取立即值還是寄存器的值。這個操作rt給的是1，於是我選擇做乘法，結果輸入到MUX\_BranchType端口2。最後結果如果最高位為0即跳轉。**

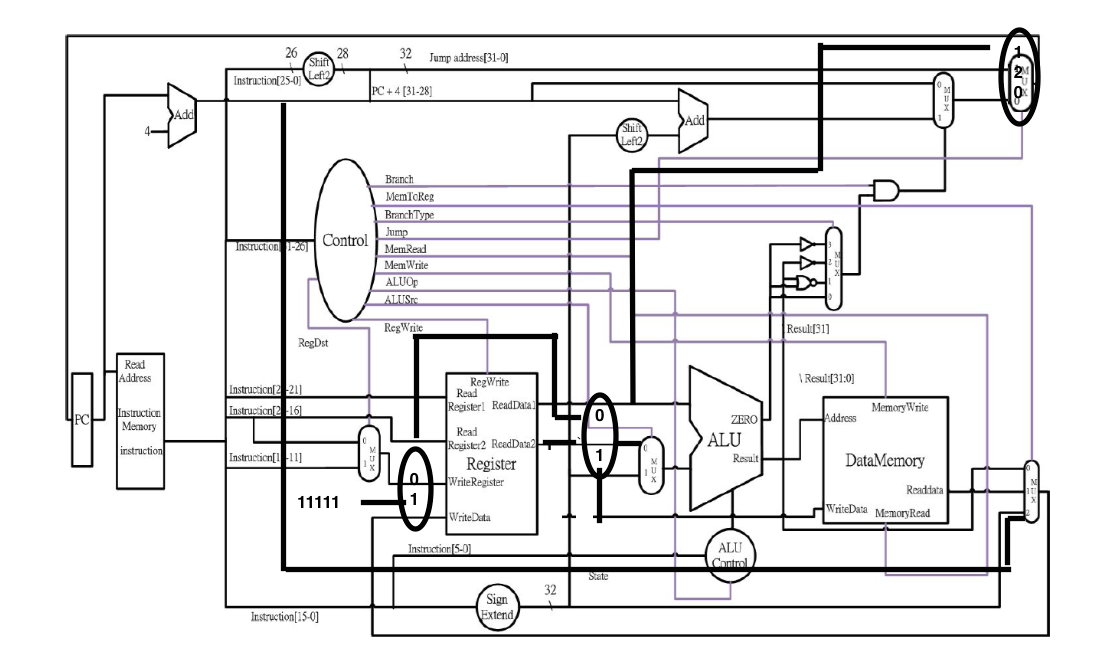
**Jal指令，大致上於Jump同理，但是需要把PC+4保存到$ra寄存器裏，我用MUX\_Jal選擇要寫入的寄存器，在這個指令中為11111。**

**Jr指令，將讀到的寄存器值通過MUX\_JJr的1號端口進行選擇。**

1. **Results**

**三個測資應該都能通過。自己修改最後一個Fibonacci Function的時候發現最大只能將$a0設為5，如果大於5就不能正常工作了。**

**整體架構圖：**

****

1. **Conclusion**

**這次實驗遇到了幾個問題。第一，在實現bgez的時候發現rt為1，一開始一直想要ALU做減法操作，但是無法判斷。通過和1做或運算並不改變最高位的的值，從而解決問題。第二， 由於Jr是R-type指令，一定要寫寄存器，所以讓ALU和0做與運算，結果始終為0，剛好保存到rd，$0中，不改變它的值。第三，還是由於R-type指令，不能讓它跳轉，於是一開始在ALU\_Ctrl中，設置IsJr的標誌，但卻不能通過第一和第二個測資。只能在decoder中加入function code的部分，來判斷是否是Jr指令。**