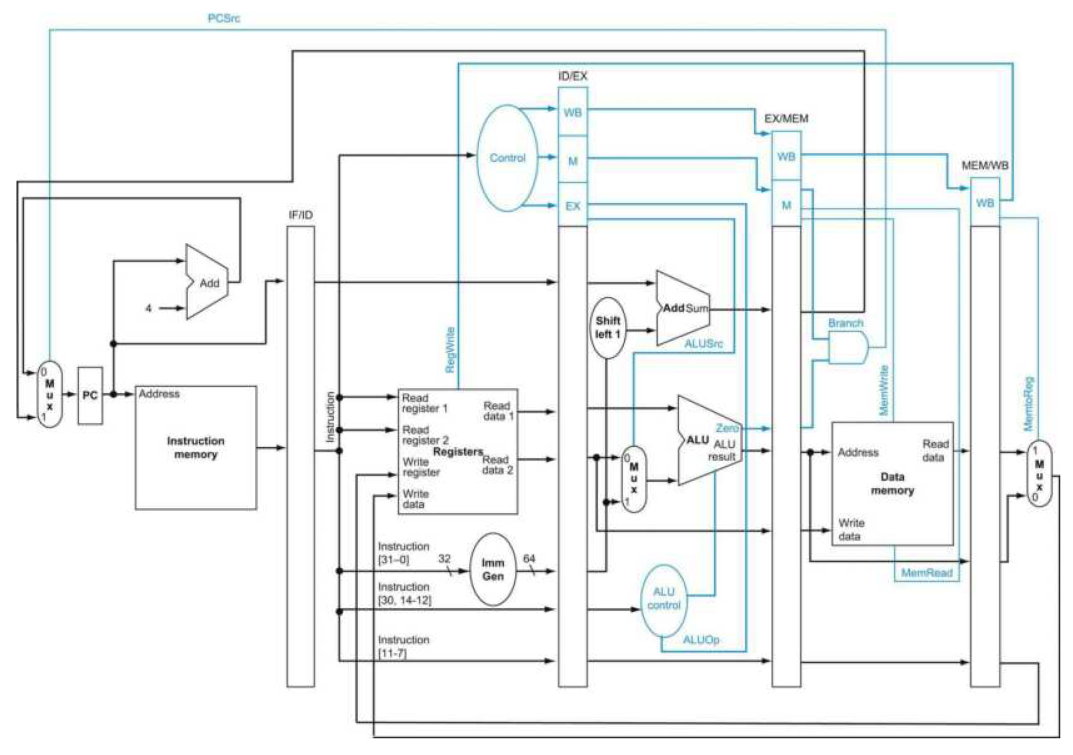
Lab5

Architecture diagram:



Your implementation detail:

* Decoder: 基本上就是將本次作業需要完成的那個表寫成code，透過判斷opcode就可以將正確的值輸出到每個控制單元
* ALU\_Control: 透過傳進來的instr和instr[30, 14:12]去判斷這個指令要做哪個運算，在提供該指令的值傳送出去
* Imm\_gen: 判斷不同的opcode使得我們可以提供適當的immediate的值
* Alu: 按照alu\_ctrl的提供的指令做相對應的運算得到result，並提供zero, overflow, cout的值
* Pipeline\_CPU: 將每個運算單元接線接再一起，組成一個完整的pipeline CPU，是這次作業比較複雜的地方
* Adder: 將兩個input相加變成output
* Exe\_register: 去判斷clk\_i和rst\_i的值，如果說是我們要的將相對應的input接到output去，若不是的話，輸出0
* ForwardingUnit: 先去判斷rs1和rs2是否和傳入mem stage的return register有相同的暫存器，如果有的話用傳入暫存器的result；在去判斷rs1和rs2是否和傳 wb stage的return register有相同的暫存器，如果有的話傳入要寫入暫存器的值；如果都沒有的話，用上一個stage傳入的值
* IF\_register: 去判斷clk\_i和rst\_i的值，如果說是我們要的將相對應的input接到output去，若不是的話，輸出0
* Imm\_Gen:
* Mem\_register: 去判斷clk\_i和rst\_i的值，如果說是我們要的將相對應的input接到output去，若不是的話，輸出0
* MUX\_2to1: 判斷傳入的值(select)為何，若為0，傳回data0；若為1，傳回data1
* MUX\_3to1: 判斷傳入的值(select)為何，若為0，傳回data0；若為1，傳回data0；若為2，傳回data2
* Shift\_Left\_1: 全部左移一個位元，最低位補0
* WB\_register: 去判斷clk\_i和rst\_i的值，如果說是我們要的將相對應的input接到output去，若不是的話，輸出0

Result:



What’s the program you encounter?

1. 在測試code的時候，傳入的值常常因為時間不一致導致結果有錯

Solution: 將pipeline register的”=”換成”<=”之後，變成全部皆為同時傳入

1. 在測試時，forwarding unit一值輸入錯誤的值

Solution: pipeline\_CPU接錯，改了之後就解決了

Anything what you want to say:

0711540: Pipeline CPU接線工作必須要很小心，只要結果出錯，檢查就會變得很麻煩，尤其是FW unit的部分。

0712511: 這次作業除了接線上比較困難外，還有一些寫程式上的細節沒有注意到和pipeline CPU的圖都零零散散的要想一下之外，其餘的部分應該都不算太難。