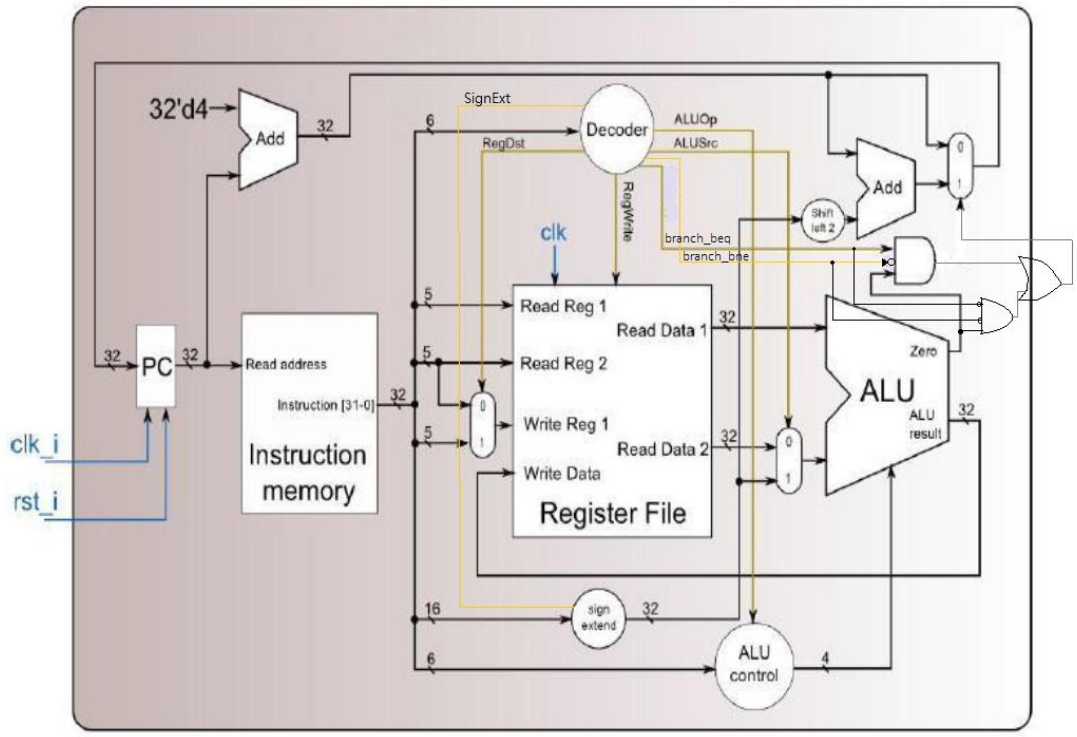
Computer Organization

Architecture diagram:



Top module: Simple_Single_CPU

1. **Add a wire 'SignExt' from Decoder to Sign_Extend module to implement 'ori' case.**
2. **Replace wire 'branch' with 'branch_beq' and 'branch_bne' from Decoder to Sign_Extend module to implement 'beq' and 'bne'.**

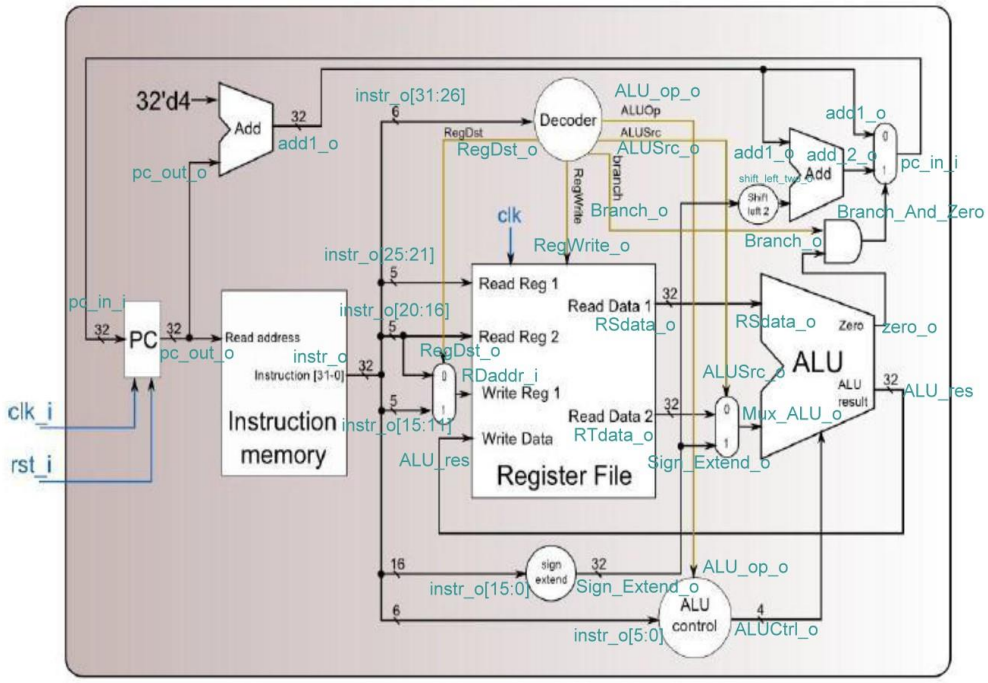
Detailed description of the implementation:

一開始進入之後 PC 會去讀每一行的指令，並且存到 IM 裡面，IM 會幫我們找到 reg 的位置；再來要判斷這個指令是做什麼運算，首先，decoder 會判斷指令為 R-type、BEQ、BNE、ADDI、SLTI、LUI、ORI、SRLV 的哪一項，並且會將 RegDst、ALUSrc、RegWrite、Branch(分成兩個：Branch_bne 及 Branch_beq)、ALU_op、signExt 的值正確地傳入所對應的 module；而 RF 會幫我們分配好 RS、RT、RD 的位置，並且傳進 ALU 裡面做運算；在 ALU 裡面，ALU 會從 ALU_Ctrl 得到指令要做 BEQ/BNE、ADDI、SLTI、LUI、ORI、SRLV 還是 R-type 裡的哪一個 function，然後使用兩個 source 做相對應的運算，將結果存到 result 以及 Zero 裡面，將

result 再傳回 RF，使值存回某個 reg（if RegWrite==1）；Zero 和 Branch_bne 和 Branch_beq 做 AND/OR 運算後，結果若為 1，就要把經過 Adder2 作運算後的 address 傳回 PC，告知 PC 要 shift 幾個指令。

Problems encountered and solutions:

（一） 礙於此 simple cpu 用到的 module 較多，接的線常常會搞混，於是做了另一張有寫上各 module input 及 output 的圖，做完之後，思緒較整潔也較不易接錯。



（二） 在寫 advance 時，原本沒有把 branch 分成 bne 及 bqe 兩個情況考慮，導致出來的結果錯誤。最後把所有 simple cpu 的 module 依各種 operation 走過一次，發現 bne 及 bqe 出錯，將負責選 address 的 2to1mux 的 select input 改成”
Branch_bne_o&(~Branch_beq_o)&(~zero_o)|(~Branch_bne_o)&Branch_beq_o&zero_o”，就成功了。

Lesson learnt (if any):

1. Learnt how to implement the datapath of R-format and I-format.
2. Realize how each module works.
3. Learnt how to combine each datapath by using multiplexer.