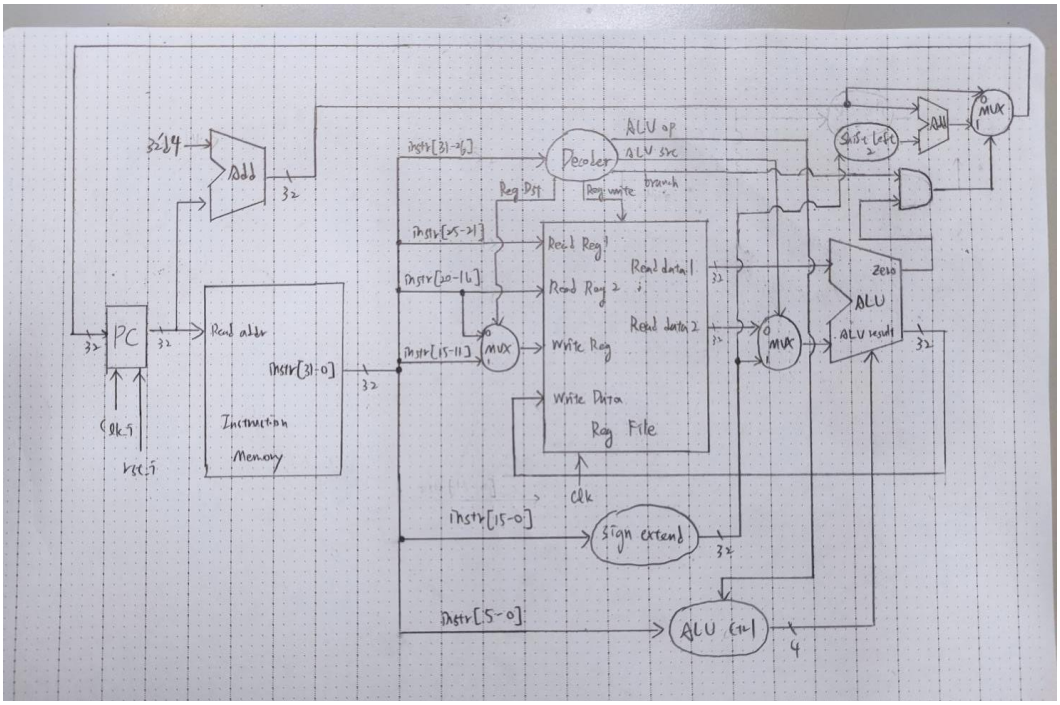
Computer Organization Lab2

Name: 單宇晟

ID: 109550087

Architecture diagrams:



Hardware module analysis:

這次 diagram 的實作方法，主要可以參考 simple_single_CPU.v。首先，會用 PC 來提供指令位址，之後會計算出下個指令的位址(PC 上面那個 Add)，同時 instruction memory 也會給出指令，並且將各位元分別傳送到指定的地方做運算，並且 decoder 在給出控制指令，告訴各個 module 現在要做甚麼之後(像是 branch 這項訊號線就和我們是否要做跳躍有關，ALUop 則決定了現在 ALU 的動作)，Reg

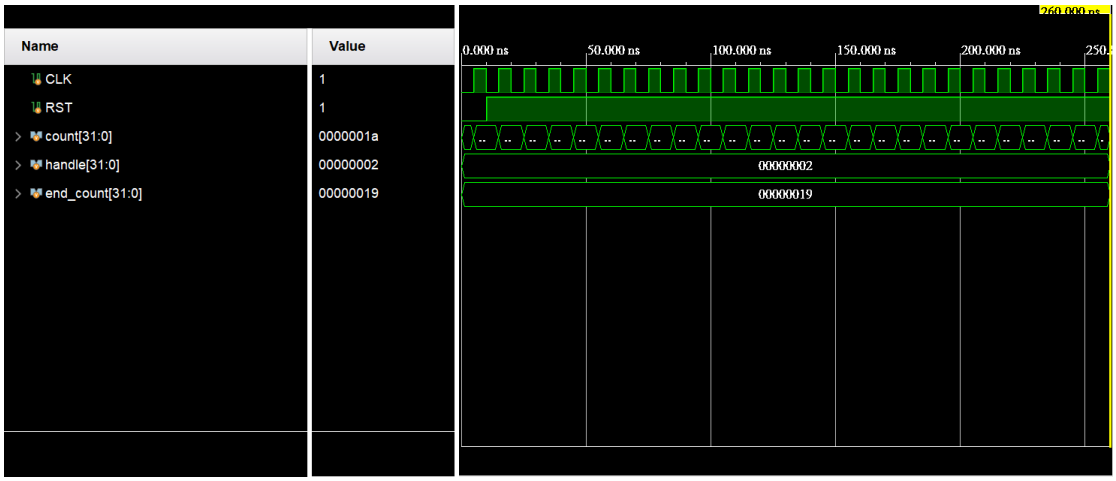
File 和 ALU 就會負責執行 R-type 指令，至於條件跳躍指令則會用 ALU 來確認條件是否成立，也會額外用一個 Adder 來計算跳躍的目標位址(PC + shift left 2 之後的 instr[15:0])。

雖然 singlecycle 的結構較為簡單，但缺點是相比 multicycle，singlecycle 的 clk 週期必須是相同長度，而此長度會由電路裡最長的路徑來決定，因此效率比較不好，且我們無法在每個指令中使用不同數目 clk。

Finished part:

Case1:

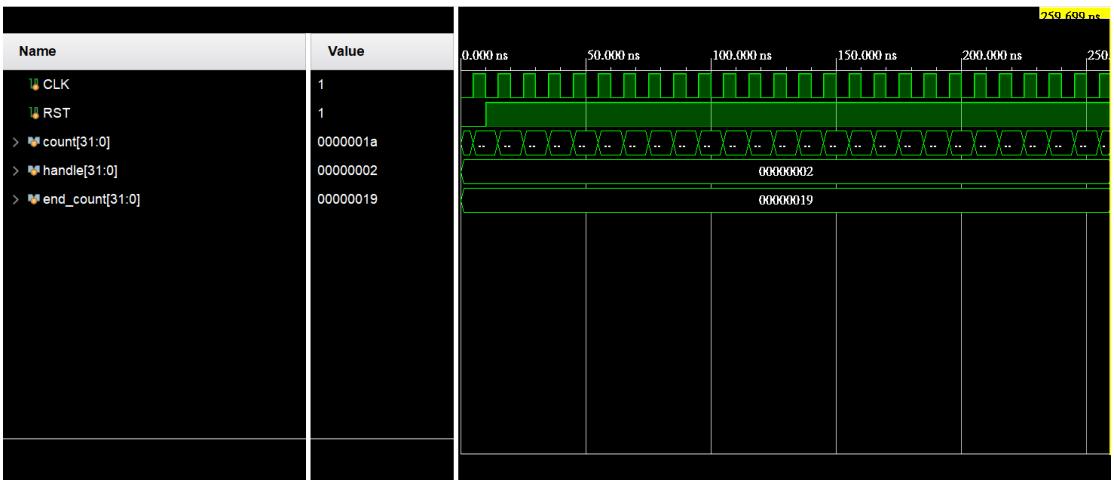
```
# run 1000ns
          2
r0=           0
r1=          10
r2=           4
r3=           0
r4=           0
r5=           6
r6=           0
r7=           0
r8=           0
r9=           0
r10=          0
r11=          0
r12=          0
```



第一個指令是 addi r1, r0, 10，r1 在這裡會變成 10(r0 一開始都是 0)。第二步為 addi r2, r0, 4，所以 r2 會等於 4。之後再接著做 slt r3, r1, r2，而由於 r1 > r2(10 > 4)，所以 r3 會被指派成 0。然後就到了 beq r3, r0, 1，因為 r3 = r0，所以這個地方會跳過下一步，也就是說 add r4, r1, r2 不會被執行到，而會直接跳到 sub r5, r1, r2，r5 就會等於 10 – 4 = 6。

Case2:

```
# run 1000ns
          2
r0=       0
r1=       1
r2=       0
r3=       0
r4=       0
r5=       0
r6=       0
r7=      14
r8=       0
r9=      15
r10=      0
r11=      0
r12=      0
```



和 case1 類似，先做 addi r6, r0, 2，所以 r6 = 2；

addi r7, r0, 14，所以 r7 = 14；

and r8, r6, r7，這時候的 r6 = 2、r7 = 14，所以 r8 = 2(用 binary 去判斷)；

or r9, r6, r7 一樣 r6 = 2、r7 = 14，所以 r9 = 14；

addi r6, r6, -1，這時候 r6 變成 1；

slti r1, r6, 1，這裡 r6 = 1，所以 r1 = 0；

beq r1, r0, -5，因為 r1 = r0(都是 0)，所以會跳到-5(從後面數回來第五個)的指令(and r8, r6, r7)；

and r8, r6, r7，r6 = 1、r7 = 14，r8 = 0；

or r9, r6, r7，r6 = 1、r7=14，r9 = 15；

addi r6, r6, -1，r6 = 0；

slti r1, r6, 1，r6 < 1，r1 = 1；

beq r1, r0, -5，r1 != r0，結束指令。

Problems you met and solutions:

這次 lab 中，一開始我對 CPU 還不太熟悉，也不知從何下手，之後把課本以及上課的講義詳讀一遍後，了解到其實這次 lab 主要是「接線」的過程，其他大部分的 function 助教都寫好了，剩餘部分也不會太複雜，很多 module 的 main function 用 if else、

case 就可以寫出來了。不過這次我卡最久的問題是 beq，因為 beq 和 sub 的電路邏輯不太一樣，加上這次助教給的模板是希望我們用 sub 去實作 beq，而我剛開始並沒有注意到，因此跑出來的結果跟答案不一樣，後來才注意的這個部分，也理解到為何要用到 branch&zero。

Summary:

總體來說，幸好這次 lab 給出了十分寬裕的時限，讓我能夠慢慢去理解 single cycle CPU 在幹嘛，我覺得相比於 lab1 的 ALU，lab2 各個 module 的難度其實並沒有比較高，有些甚至更簡單，但難的地方就在要如何把它們接在一起，就像我上面提到的，這次 lab 更像是一種「接線」的實作，我也是一邊看 lab2 的 diagram，一邊把 simple_single_CPU.v 給做出來的。雖然這次花的時間比上次多上許多，但做出來的那一刻，我也得到蠻大的成就感，另外也想謝謝助教們在討論區的熱心解答!