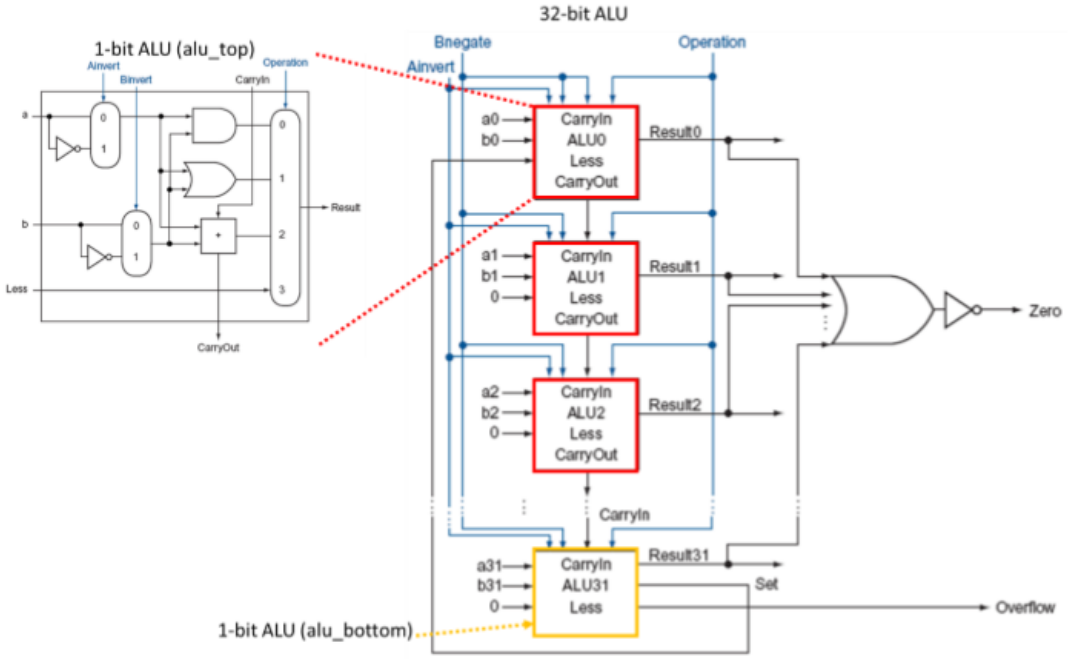
Computer Organization (Lab 1)

0613144 葉之晴

Architecture diagram:



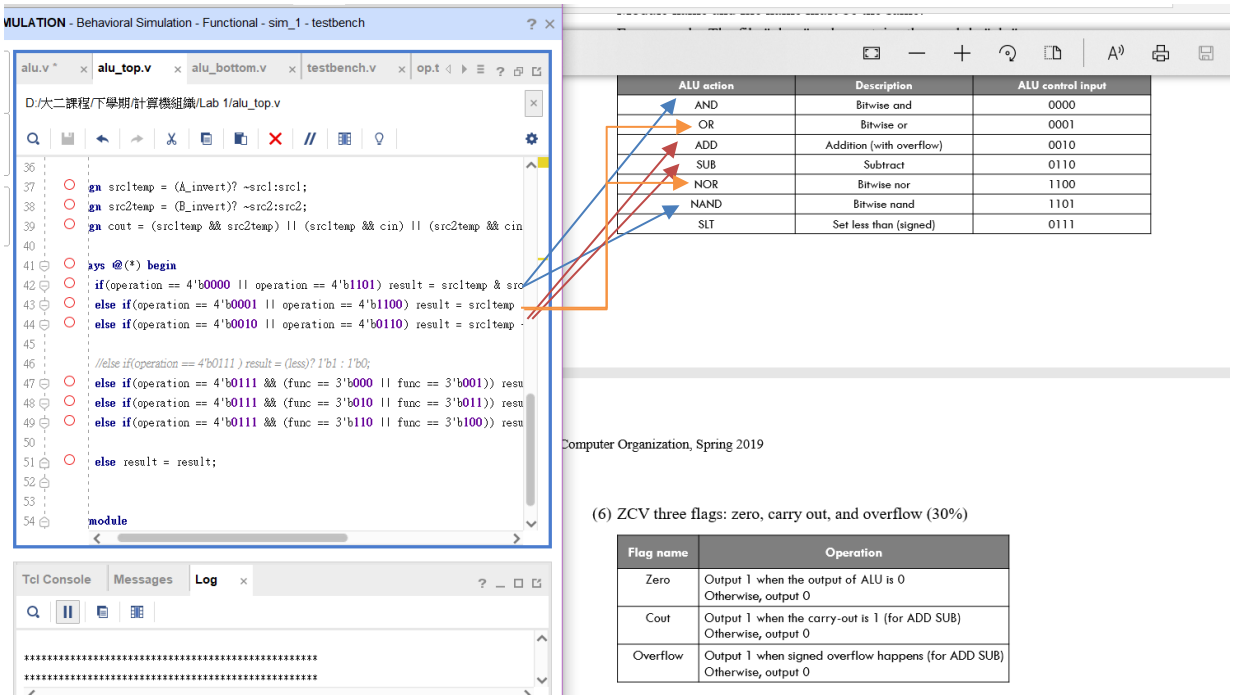
Detailed description of the implementation:

首先這題 32 bit ALU，先將程式碼分為三個部分，一個是主程式的 alu 檔，另外為前面 0-30 個的 alu_top 檔，以及最後第 31 個會做 overflow 的 bottom 檔。

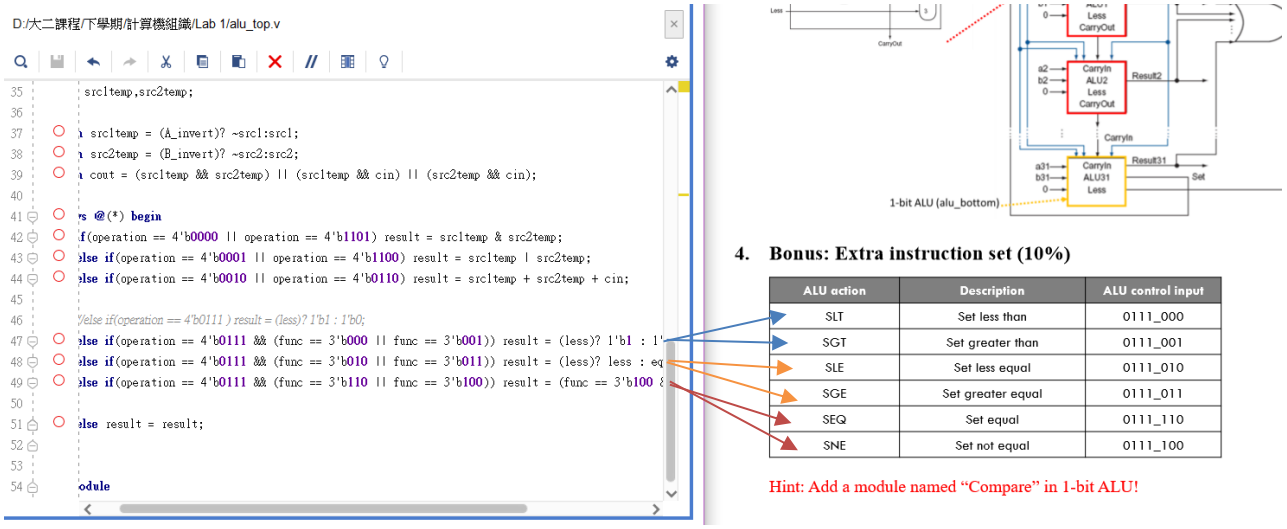
在 alu 檔內，除了建構 input 和 output 外，這題目為 32bit，所以在數字的設定上，先寫完 32 個，且前一次的 carryout 為後一次的 carryin 做設定，而後者的 overflow 和 zero 皆為輸出的值，最後一個拿來做 reset 的動作。

在 alu_top 檔內，進行各個 basic 和 bonus 的運算元計算。特別在

assign haha 內，先去設定為 1 做補數的計算，才可在 set_less 內取得題目要求的。其餘 basic 指令如下圖 1 箭頭對照到，bonus 如圖 2。



(圖 1)



(圖 2)

在 alu_bottom 檔內，如上面 diagram 圖可知，bottom 和前面 top 不同的地方在於輸出 overflow，所以程式碼上會多設一個 overflow

的 output。

Problems encountered and solutions:

Problem1：overflow 無法正確判讀

Solution1：原本我只有定義 if 內定義成 carry_out 為 0 且做加法或是 carry_out 為 1 且做減法，但跑出來 data 4 一是錯的，後來我重新在書本上找 overflow 的定義，為正數加正數結果卻為負數，以及負數加負數結果為正數。因此我們將原先 carry_out 為 0 且做加法後，再&&結果 src1 和 src2 為 1 的時候，則 overflow；反之 carry_out 為 1 且做減法後，需再與得到的結果做&& src1 和 src2 為 0 的時候，才也會為 overflow，其餘皆不是 overflow。

Problem2：bonus 無法正常讀取

Solution2：在做 bonus 時，發現 basic 後的 data 都無法正常讀取，多次測試了每個 result，也嘗試把不能動的 testbench 檔內的 basic 程式碼刪掉，只留下 ifdef BONUS 後的讀檔動作，才有辦法正常讀 bonus 的 data，後來再檢查 testbench 檔時，發現我沒把 define BONUS 前的註解刪掉，才導致無法正確讀檔。

Lesson learnt (if any):

從這次的 lab1 不僅僅可以更熟悉 verilog 的基本動作，也更了解 32 bit ALU，同時在思路上，因為需將理論寫成 verilog 可

讀取的語法，所以必須非常清楚，才能完整表達出來，也在寫作業時，查了很多相關資料，讓過去所學的內容更加深刻和了解。如上面 problem1 所述，一開始並沒有很確定 overflow 的定義，後來重新查書後，才有了更明確的邏輯，同時也得到解決。