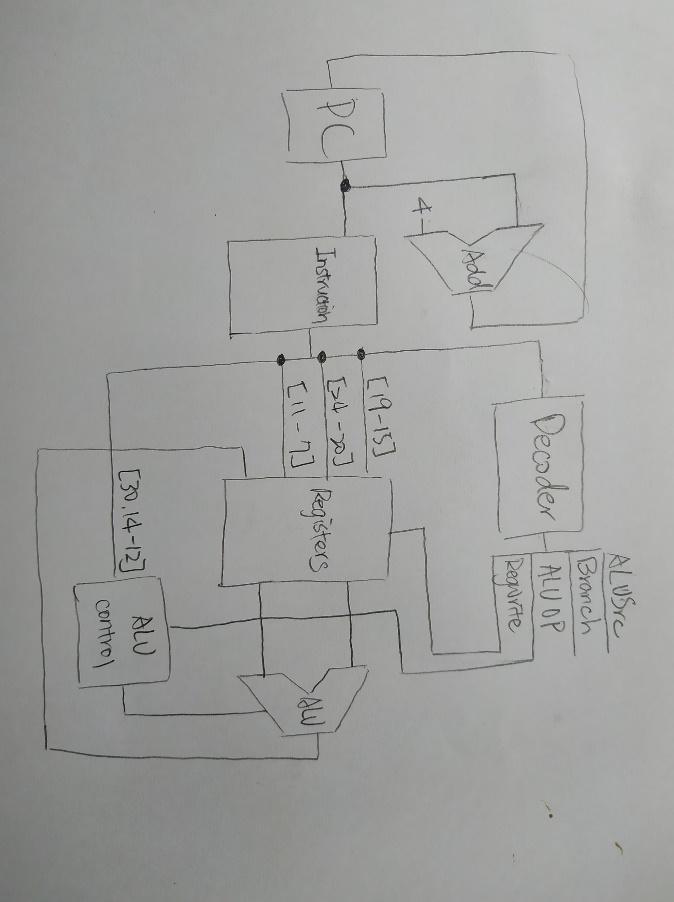
**Computer Organization**

**Architecture diagram:**

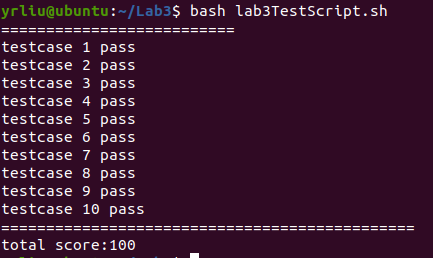
****

**Detailed description of the implementation:**

* Adder：透過 assign wire 另外兩條 src 的和即可。
* ALU control：先透過ALUOp進行分類，提供相對應的ALU\_Ctrl；若ALUOp為”2’b1x”，我們再透過instruction[30, 14-12]去判斷這個指令是要做哪一種operation，輸出相對應的ALU\_Ctrl。
* ALU：依據ALU\_Ctrl為case的條件，對兩條src做出相對應的運算而得result，並且另外儲存過程中的carryout以便計算ZCV。zero由nor gate可得；cout由carryout[31]可得；overflow由carryout[30] xor carryout[31] 可得。
* Decoder：根據指令type為case的條件(由instruction的opcode可知)，而在本次作業中由於經過簡化，因此我們可以直接透過opcode直接傳出所需的Branch, ALUOp, ALUSrc, RegWrite。
* Simple\_Single\_CPU：透過這個檔案拉線將整個simple single CPU組起來。

**Implementation results:**

* 10筆測資全數通過



**Problems encountered and solutions:**

* 遇到sra, sll遇到ALU control時的編碼問題

-->最終透過自行設定代號作為條件解決

* 處理ZCV時無法直接運算

-->透過存取carryout解決問題

* 實作sra時，由於剛開始沒發現這次的作業是做sign做運算而使用”>>”

-->將operator改為”>>>”運算結果就正確了

**Comment:**

* 這次的作業相較32 bits ALU輕鬆，而且有前次lab2的訓練，對於verilog的語法掌握程度提高很多
* 在實作CPU中，由於許多結構在這次的架構中不需要時做出來，因此只需要處理進入的data以及將每個構造的線一個一個接起來