Computer Organization

Hardware module analysis(簡短解釋一下每個 module 的功能，以及在 CPU 裡扮演的角色):

Adder:如字面上做兩個數字的加法，在 cpu 裡用來做像 pc＋4 的指令和 branch 地址＋pc_next 的指令．
Alu_ctrl:來控制 alu 要做什麼事．
Alu：作加減法和 slt 等等的兩個變數的運算，最後輸出．
Decoder:用 opcode 和 function 來 derive 出一些 control signal．
Instr_memory:讀取 instruction 的 module;
MUX_2to1：讓兩個 input 訊號二選一之後輸出結果，通常用 control 的訊號的值來做選擇。
ProgramCounter:來有順序的讀取 instruction，並記錄目前讀到哪．
Reg_file:存放 register 的地方，讀取和寫入 reg 的地方．
Shift_Left_Two_32:將一個 32bits 的訊號往左 shift 2 個 bits 然後最後面補兩個 0，在 cpu 裡用來將 branch 的地址往左 shift 2 個 bits．
Sign_Extend:將一個 16bits 的訊號擴展為 32bits 的訊號，正數就前面補 16 個 0，負數則是補 16 個 1，在 cpu 裡用來讓 immediate 指令的數字和地址來補齊 32 個 bits．:
Simple_Single_CPU:將上面所述的所有 modules 全部串接再一起的 module，也就是 top module．

Problems you met and solutions:

在寫的時候我遇到很多困難，第一個遇到的就是要去 derive decoder 的 alu_contorl，那個花了一點時間才得到正確的答案，在接下來遇到的就是 single cycle cpu 裡接線的問題，因為線路有點繁瑣所以花了一段時間將其好好地接好，最後又 debug de 了好長一段時間，最後發現是 alu 的 default 要設為 0，不然跑出來的結果永遠都是錯的，還有 decoder 的 regdst derive 錯了，最後這兩個地方用了好久才用出來，算是這次的 lab 遇到的最大困難了吧！

Summary:

這次的 lab 將以前在數電學的一些像 mux 啊 decoder 啊等等的全部都自己手動實做出來，雖然難度沒有很高但還是有一種之前學的東西有派上用場的感覺，因此在寫的過程豬還滿有成就感的，最後因為這次有很多 module 所以 debug 會困

難很多，花了很多時間才 de 出一些小錯誤，不然之前怎麼跑 result 都是亂碼，因此也讓我學會了很多 debug 的方法，算是有學到東西，這次寫 verilog 發現自己很多基礎都不太會，像是有些地方要用 reg 都不曉得，讓自己多花了很多時間，不過最後能自己時做出一個簡單 cpu 還是非常的有成就感的！