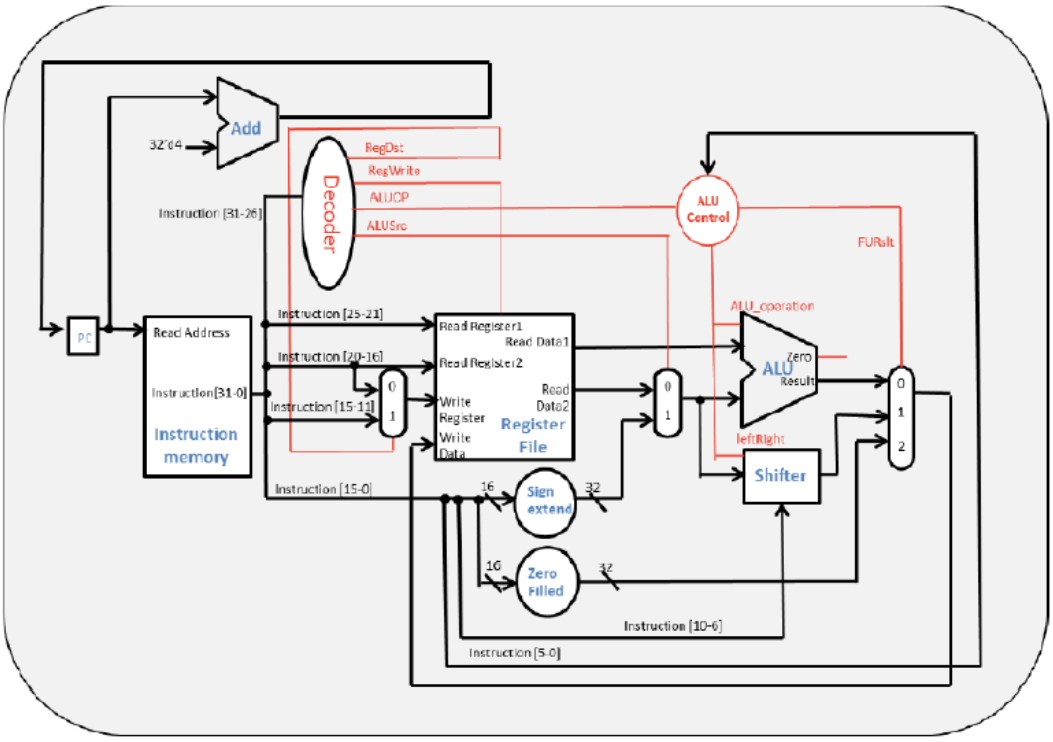
# Computer Organization

## Architecture diagrams:




## Hardware module analysis:

1. Adder

將兩個 32 bits 的 in put 的資料直接相加，實作上直接用 assign 相加即可。

2. Decoder

Input instr_op，再根據 instruction set 中的 Op field 對應出不同的 output (RegWrite, ALUOp, ALUSrc, RegDst)。

3. ALU_Ctrl

根據 Decoder 給出的 ALUOp 作為 input，再 input funct，根據 instruction field 中的 Function Field 對應給出不同的 output (ALU_operation, FURslt)。

4. MUX2to1

根據 input 的 1bit 的 select_i，選擇輸出為 input1 或 input2。

5. MUX3to1

根據 input 的 2bit 的 select_i，選擇輸出為 input1,2 或 3。

6. Sign_extend

將 16 bits 的 data 拓展成 32 bits 的數據，將 0-15 位元的資料設為原本 input 的值，並將 16-31 位元的資料設為 input data 最高位(第 15 位元)的值。

7. Zero_filled

將 16 bits 的 data 拓展成 32 bits 的數據，將 0-15 位元的資料設為原本 input 的值，並將 16-31 位元的資料設為 0。

8. ALU_1bit

此 module 為上次的作業內容。根據 input (a, b invertA, invertB, op, cin, less)給出 ALU 計算過後的 output。實作上 a, b 為 1 bit 的 data，依據 op 及 invertA, invertB 決定做何種運算。

9. ALU

此 module 為上次的作業內容。利用 32 個 ALU_bit module 去計算 32bits 的 ALU 結果。實作上利用 generate 及 for 迴圈做出 1 到 31 位元的 ALU，第 0 位元的 cin 及 less 需另外接線。

10. Shifter

根據 leftRight 判斷 shift left or right。

11. Simple_Single_CPU

將所有 module 統整起來，依照 architecture diagram 接線，做出 simple single cycle CPU。

## Finished part:

完成了所有 module 並通過助教提供的三個測資

## Problems you met and solutions:

在實作 Decoder 及 ALU_Ctrl 的時候，搞反 operation 和 function 的 6 bits input，中間判斷上又寫錯一些東西，結果就大爆炸，花了一些時間搞懂 spec 中的 instruction set 的內容，最後才修正成功。

## Summary:

我覺得最複雜的點應該就是判斷 op 及 funct 的 data 並給出對應個 output，要了解每一步每一個 module 每一條線代表的功用，最後在 simple_single_CPU 中接線，接線只要按照圖上的內容一條一條接就好，挺有趣也挺麻煩的。