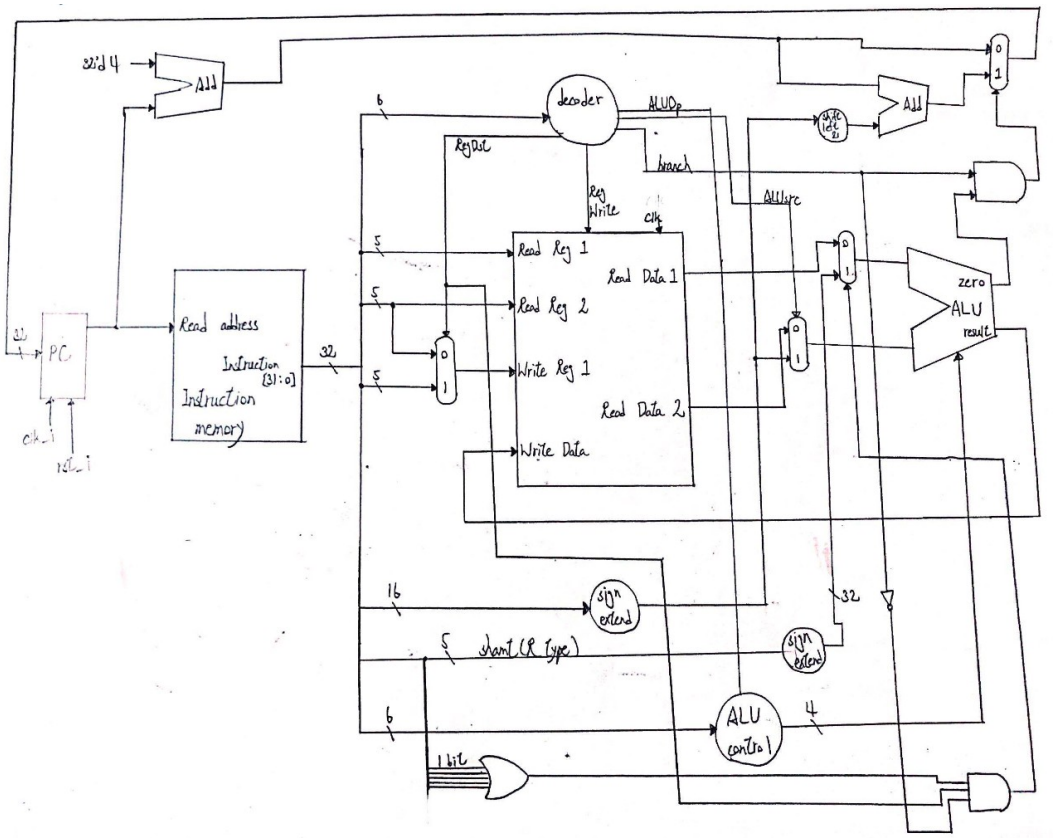
Computer Organization

Architecture diagram:



Detailed description of the implementation:

主要是依照前 6 bits 的 opcode 來作為 decoder 分配 control input 的依據, 決定 RF 要怎讀寫 data (R type , I type, by MUX 2 to 1) 、 ALU 要讀哪個資料(用 MUX 2 to 1)、 要不要 branch 到其他位置(branch = 1)等等, 而 ALU Control 用來判斷 ALU 該做什麼 (依照 function code 及 ALUOp)

其他細節的部份是 adder 是 兩個 32 bits data 相加

mux 2 to 1 是根據 select input 決定 兩個 data 哪個是 output

sign extend 是把 16 bits data 變成 32 bits (根據第一個 bit)

shift left two 是把 data 向左移動兩位 (*4)

Problems encountered and solutions:

一開始遇到最大的問題就是理解題目的意思, 我們嘗試從課本看懂 CPU, 但照著做完後才慢慢理解, 題目的 code 跟課本的很不一樣, 因此大改了 decoder 跟 ALUCtrl
第二個問題就是 SRA, SRAV, 原本以為只是單純的 shift 錯而已, 但仔細思考才發現 SRA,SRAV 是用 R type, shamt 是吃不到值的, 掙扎了很久後才決定不照 pdf 上的 Architecture diagram, 自己新增 5 bit 可以讀 shamt 在 cpu 上, 才測資全過

Lesson learnt (if any):

最大的收穫就是能看懂 CPU 的 control 了, 也讓我對 verilog 語法更加熟悉, 雖然現在還是寫得很醜, 但覺得有看到越來越多黑科技, 持續進步中!!