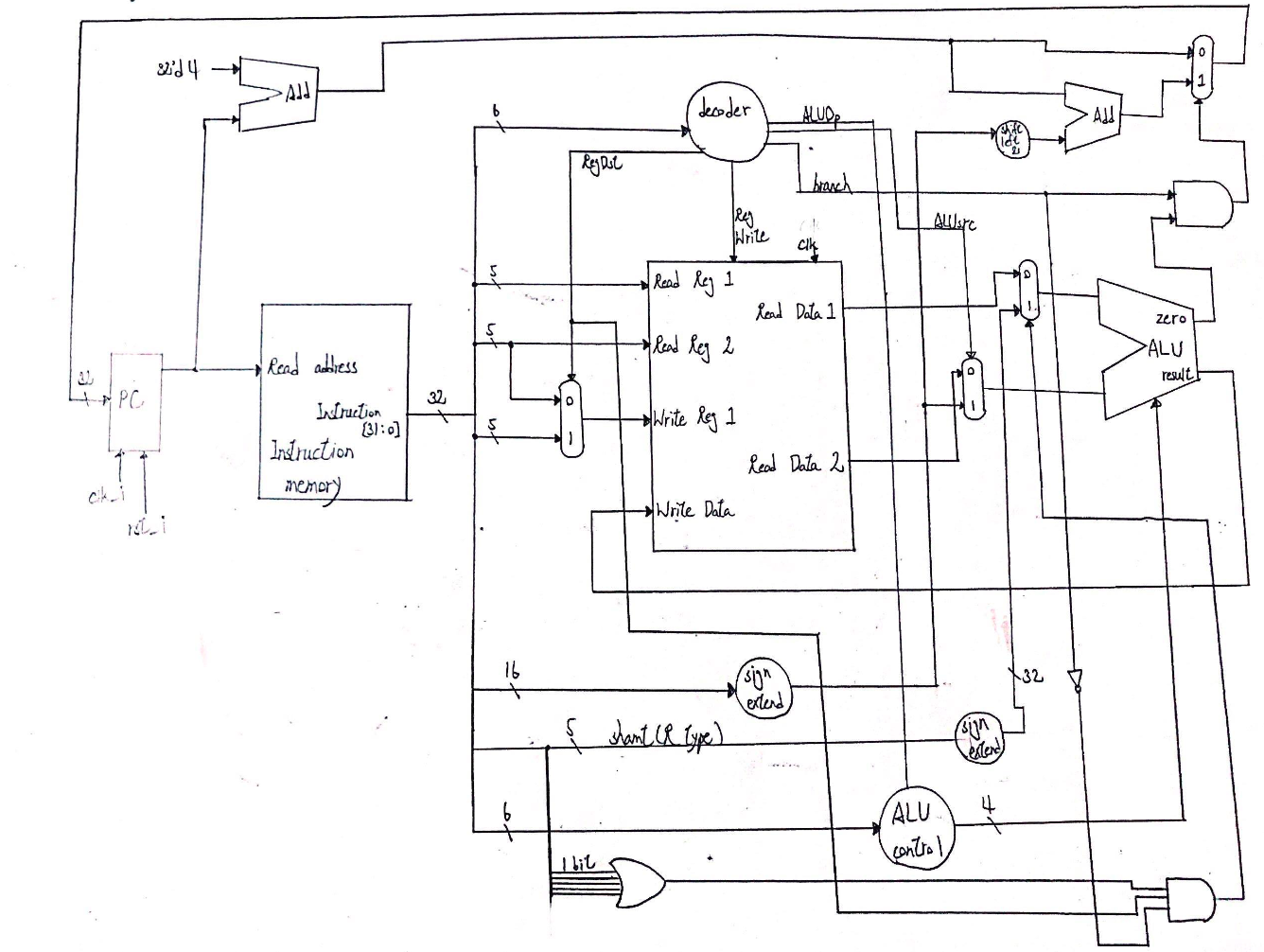
**Computer Organization**

**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!!