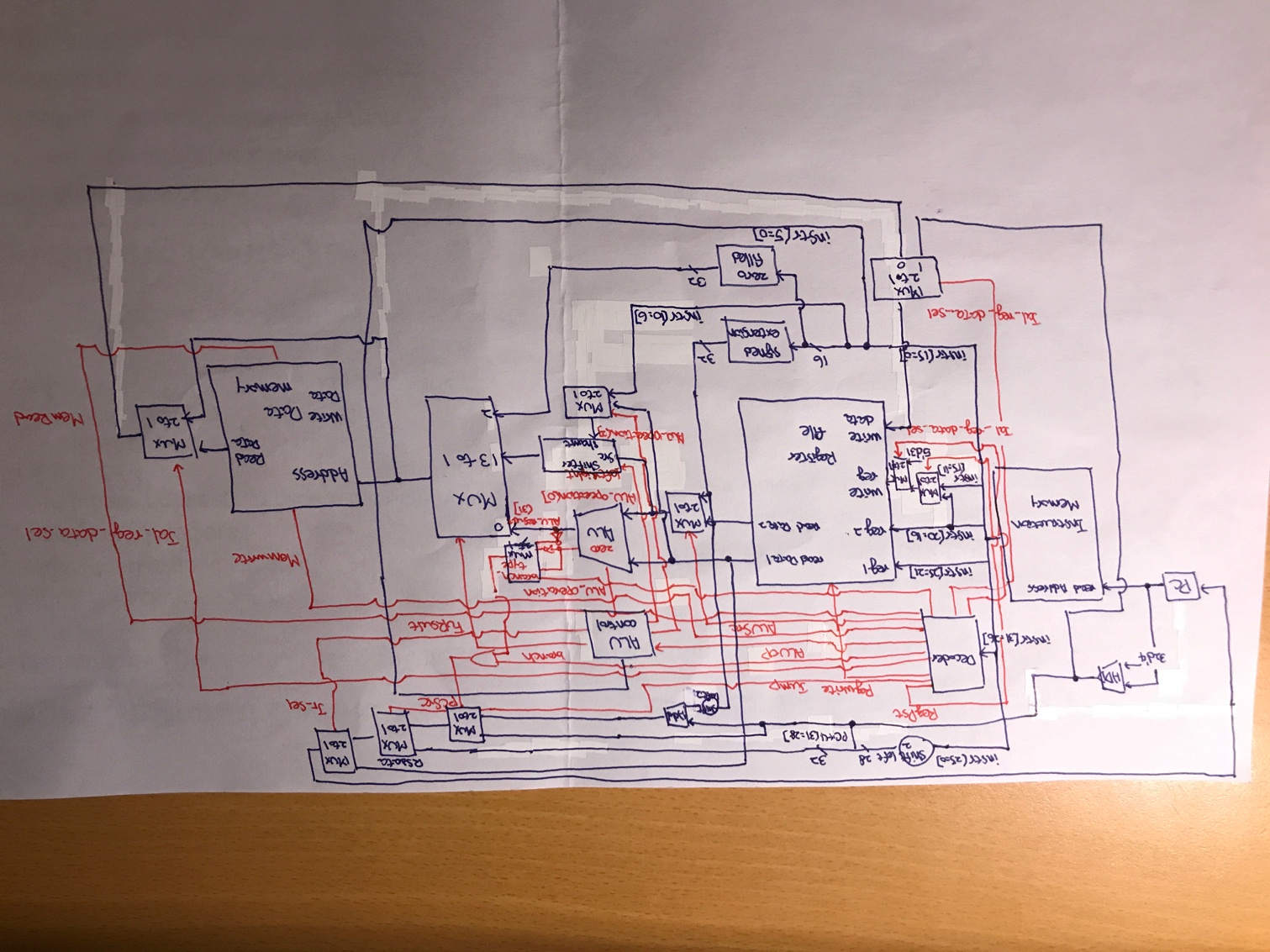
**Computer Organization**

**Architecture diagrams:**

**Hardware module analysis:**

**PC: Program counter, 用來指向instruction memory裡面指令的位置，從0開始每次加4。**

**Instruction Memory: 用來記憶cpu指令，測資會從此讀入cpu。**

**Register File: 儲存cpu內的reg值，模擬register的讀出跟寫入。**

**ALU: 執行從register file內讀出或是從指令讀出的數值，對其進行運算。**

**Shiftrer: 將拿到的數值進行Left 或是Right Shift並傳給Mux 3 to 1。**

**Signed Extension: 看輸入的數值最高位為1或0，而對higer 16 bits補1或0。**

**Zero Filled: 將輸入的數值往左推16個bits，而lower 16 bits補0。**

**Decoder: 將讀到的指令解析判斷是哪一種形式的指令（R type, lui 或是其他），進而得知後面的module（register file , mux ,alu等）該做什麼。**

**ALU Control: 根據decoder解析出來的ALUOP與instr[5:0] (function code)，來判斷該讓ALU進行什麼樣的功能運作。**

**Data\_Memory: 進行Sw, Lw 的功能的時候，會需要用到此register，主要為memory讀寫之用途。**

**MUX: 此次的重點，主要是基於上一個lab為基礎，再加上額外的mux，例如像是在Data\_Memory之後加上的決定Register File 的wirte\_data的兩個mux電路，還有在做branch的時候，不一樣的branch功能，會需要用到alu不同的output進行branch判斷，或是在Jump的電路中，除了要把收到的值進行運算、補位，也要使用mux來判斷此instruction是否為jump，還是只要送原本pc+4的counter回去即可。**

**Finished part:**

**Adder.v , ALU.v , ALU\_Ctrl.v , Decoder.v , Mux3to1.v , Mux2to1.v , Simple\_Single\_CPU.v , Shifter.v , Sign\_Extend.v , Zero\_Filled.v, Data\_Memory.v, Reg\_File.v**

**Problems you met and solutions:**

**由於前次lab的設計還算完整，因次在這次的lab中，基本上只要事先將要加上去的電路先規劃好，並且對線路做一些修改。然而，在設計某些指令的時候，往往會需要把前面設計的東西再做大或修改，因此成為這次lab中最大的挑戰，要非常謹慎小心的把每一個電路設計好，不然debug將會是一條漫長的路。**

**Summary:**

**隨著一次又一次的lab，看著cpu設計越來越複雜，覺得欣慰卻又覺得害怕，因為越複雜的電路，當下一次作業要在放新功能的時後，需要考慮的地方又更多了，而下一次就要把simple\_single變成pipeline了，希望到時候能夠順利的把cpu升級成功。**