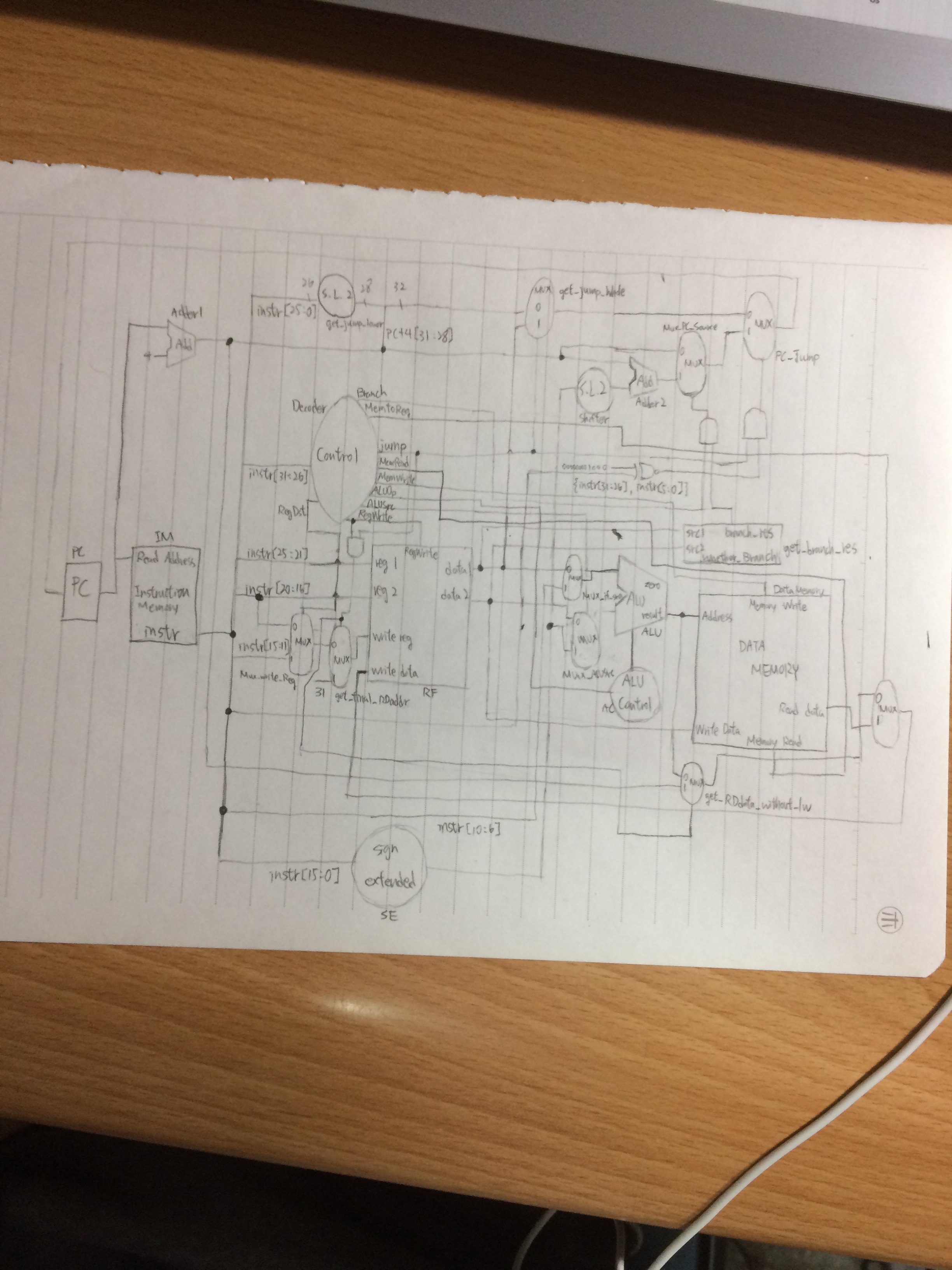
**Computer Organization**

**Architecture diagram:**



**Detailed description of the implementation:**

**接續Lab2**

**lw:**

**首先在decoder，ALUSrc, RegWrite, MemoryRead, MemtoReg都設為1，ALU\_op則設為和addi一樣的000，因為最後需要取用memory中result的值。**

**接著要對memory做存取的動作，這部分直接拿ALU的result當作address。**

**最後把剛才存取的資料放到register中，這部分需要一個Mux來判定要使用之前的data當作存進去的值還是剛才得到的資料，MemtoReg為select。**

**Sw:**

**首先在decoder，ALU\_src, MemWrite為1，ALU\_op則設為和addi一樣的000，因為最後需要讀寫memory中result的值。。**

**接著要對memory做讀寫的動作，寫入的資料為暫存器rt的值，位址則是剛剛ALU算出來的result。**

**Jump:**

**首先要在Decoder中多一個判斷，如果判定1~5 bit是00001，就讓jump=1。但是在判定裡面要再判定如果是指令j，就要讓reg\_write=0，jal則為1。jr的話會先被當成R-type，只是會再跟其他R-type做判定區別，一樣jump=1，reg\_write=0。**

**如果是j, jal，PC要變成後26bits的shift left 2並補上原本PC+4的前4bits。jr則是取用rs暫存器的值做位址。接著還需要一個Mux來選擇要使用j,jal的位址還是jr的位址，這部分就直接用opcode+functioncode來當作select。**

**還有關於jal存取r31的部分要再用兩個Mux分別決定最後的RDaddr和RDdata。**

**最後還要跟之前的PC+4或branch用Mux來決定PC最終要到哪個位址，這部分的select則是指接用jump即可。**

**Branch:**

**首先在decoder中把branch設為1，接著要看是哪種branch，這裡我們沒有照著教科書多設一個branchtype，而是直接拿opcode來判定，並新增一個whether\_branch.v來判定branch的結果是否為真，並把結果存在branch\_res中。**

**得到判定結果之後，就放到Lab2的Mux，以Branch & branch\_res為select，得到下一個PC。**

**Problems encountered and solutions:**

**Branch的部分一開始想寫在ALU裡面，但是後來發現ALU\_op只有3bits不夠用，又不想改太多以前的東西，所以就另外寫個whether\_branch.v來得到branch\_res。**

**Debug超麻煩的，我們最後還一步一步拆解，最後才成功debug。QAQ**

**Lesson learnt (if any):**

**學到如何寫一個完整的CPU**