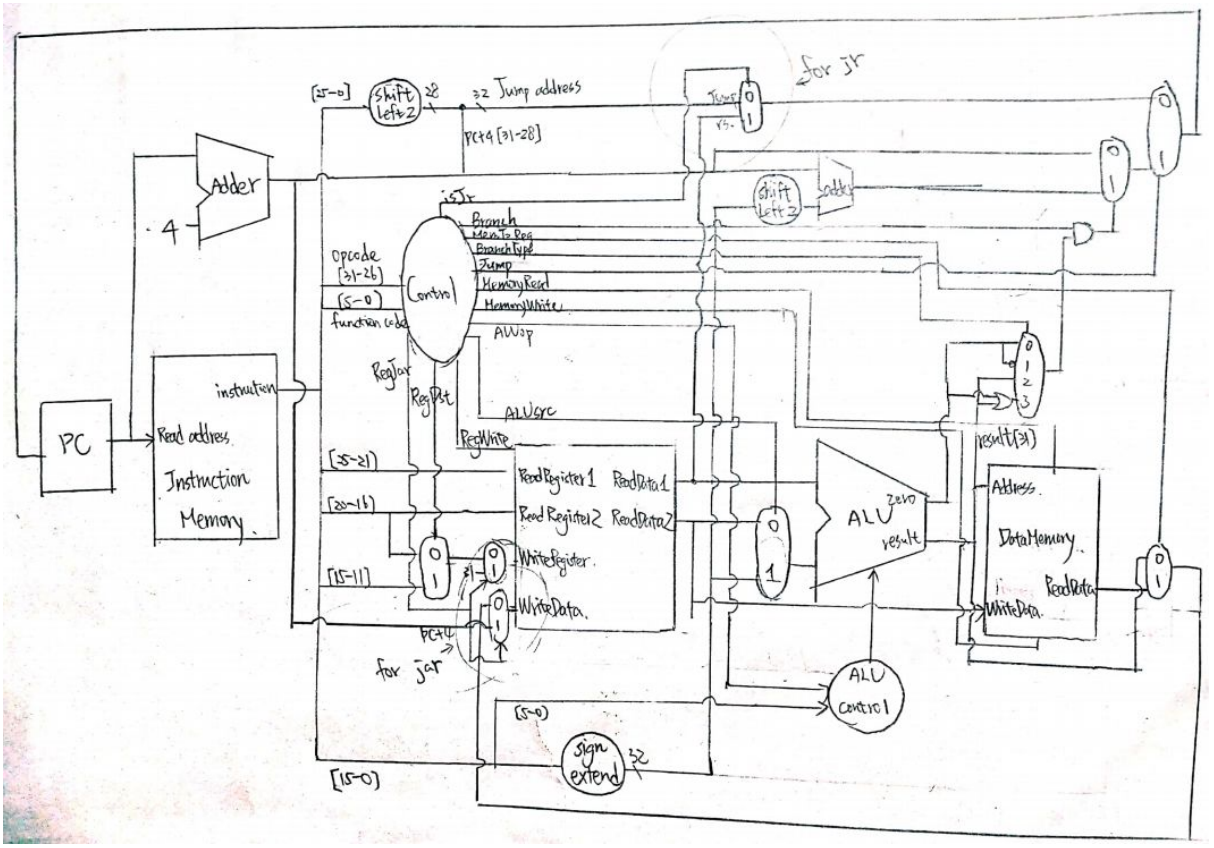
# Computer Organization Lab-3

0616014楊政道, 0616225張承遠

## Architecture diagram:




## Detailed description of the implementation:

1. 實作細節們

(1) Data Memory

多加兩個Control分別為MemoryRead跟MemoryWrite，控制對Memory的操作

(2) jump

jump指令把instr[25-0]先*4再跟PC+4的最高四位接上，並且寫回PC即可

(3) jr

jr要將ra($31)的值讀出來，再寫回PC。需要一個MUX來控制寫回去的值是jump還是($31)，並用一個control signal(命名為isJr)來選擇。

(4) jal

跟jump是一樣的操作，但是多了一個要回寫PC+4進($31)的動作，因此WriteRegister跟WriteData前面個要多一個MUX，並加一個Control Signal從ALU Control拉出來(因為jar為R-type，要從function code才能辨認出他屬於jal指令)。當執行jal，除了執行jump操作，也要把PC+4存進去($31)內。

2. Control Signal Table

| | RegWrite | ALU_src | RegDst | Branch | MemWrite | MemRead | MemtoReg | Jump | RegJal | ALU_op | isJr |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **R_type 000000** | **1** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0010** | **0** |
| **Addi 001000** | **1** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0011** | **0** |
| **Beq 000100** | **0** | **0** | **X** | **1** | **0** | **0** | **0** | **0** | **0** | **1010** | **0** |
| **Lw 100011** | **1** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0000** | **0** |
| **Sw 101011** | **0** | **1** | **x** | **0** | **1** | **0** | **1** | **0** | **0** | **0001** | **0** |
| **J 000010** | **0** | **X** | **X** | **0** | **0** | **0** | **0** | **1** | **0** | **x** | **0** |
| **Jal 000011** | **1** | **X** | **X** | **0** | **0** | **0** | **0** | **1** | **1** | **x** | **0** |
| **Jr 000000** | **0** | **x** | **X** | **0** | **0** | **0** | **0** | **1** | **0** | **x** | **1** |
| **Ble 000110** | **0** | **0** | **X** | **1** | **0** | **0** | **0** | **0** | **0** | **1000** | **0** |
| **Bnez 000101** | **0** | **0** | **X** | **1** | **0** | **0** | **0** | **0** | **0** | **0111** | **0** |
| **Bltz 000001** | **0** | **0** | **X** | **1** | **0** | **0** | **0** | **0** | **0** | **1001** | **0** |
| **Li 001111** | **1** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0000** | **0** |

# Problems encountered and solutions:

**乍看之下這次lab只是上次lab的延續，應該加上幾個功能就完成了，但這次的lab跟上次比起來多花了我大概兩三倍的心力。首先遇到的問題就是要熟悉每個指令的參數及意義，以及每個指令對應到的control變數。一開始寫decoder檔的時候很快就被一堆指令搞得暈頭轉向，而且常常在Debug，到後來我先建好一個每個指令對到每個control的表格，建好之後再來寫code思路便清晰許多。**

**接著遇到的問題便是接線的問題，其中最困擾我的便是jr這個指令。Jr這個指令屬於R-type，但jr指令有些性質卻與R-type不一樣。像是R-type的RegWrite一般來說都是1，但jr並不用寫入結果到result裡面，因此必須特殊處理這個指令。其他問題還有兩個值比大小的時候要判斷 sign bit 再比，不能直接比大小，以及Debug比上次還要花時間，因為這次的測試資料都比較大。**

## Lesson learnt (if any):

**這次的Lab，當然地讓我更加深的CPU的架構，寫完之後腦海裡都可以浮現出一個清楚的SimpleCPU架構。同時這次的Lab也讓我了解到細心是很重要的一件事情，稍微打錯一個數字都可能會導致與正確結果相差甚遠。而且了解CPU架構跟實作CPU也是有很大的差別，實作CPU的時候要考慮到很多實作的小細節。**

## How to compile:

```
iverilog testbench.v Simple_Single_CPU.v Adder.v ALU.v ALU_Ctrl.v Data_Memory.v
Decoder.v Instr_Memory.v MUX_2to1.v ProgramCounter.v Reg_File.v Shift_Left_Two_32.v
Sign_Extend.v
```