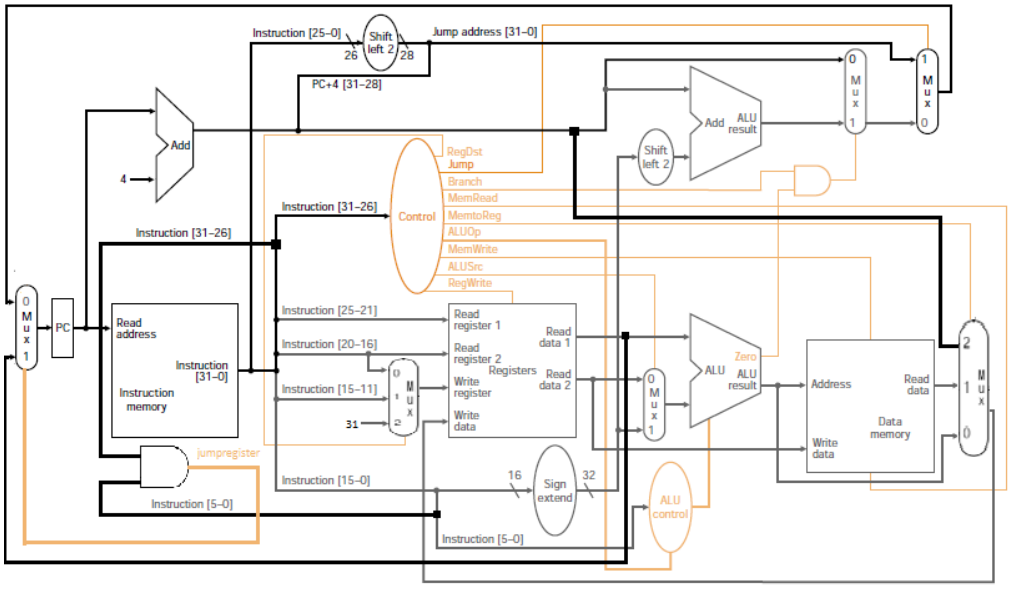
Computer Organization

Architecture diagrams:



Hardware module analysis:

Adder: 使用 **2** 個 **Adder**

- **(i)** 負責計算 **PC+4**
- **(ii)** 負責計算 **imm**

ALU: 負責邏輯及加減運算

ALU_Ctrl: 負責 **ALU** 的 **control signal**

Data_Memory: 負責存放 **data** 的 **memory**

Decoder: 針對不同的 **operation**，給予對應的 **control signal**，個人認為最重要的核心

Instr_Memory: 負責解讀 **instruction**

MUX_2to1: 使用 **4** 個 **2 x 1 MUX**

- **(I)** 檢查是否執行 **jr**
- **(II)** 檢查 **R – format** 還是 **I - format**
- **(III)** 檢查是否執行 **jump**
- **(IV)** 檢查是否執行 **branch**

MUX_3to1: 使用 **2** 個 **3 x 1 MUX**

- **(I)** 判斷 **Write Register** 的值
- **(II)** 判斷 **Write Data** 的值

ProgramCounter: 了解目前的 **PC**

Reg_File: 拆解 **operation** 里要用到的 **register**，并存到對應 **register**
Shift_Left_Two_32: 向左移 **2** 個 **bit**
Sign_Extend: 延伸 **leftmost bit**
Simple_Single_CPU: 把 **module** 接起來

Finished part:

-CO_P3_test_data1

```
# PC =        128
# Data Memory =          1,          2,          0,          0,          0,          0,          0,          0
# Data Memory =          0,          0,          0,          0,          0,          0,          0,          0
# Data Memory =          0,          0,          0,          0,          0,          0,          0,          0
# Data Memory =          0,          0,          0,          0,          0,          0,          0,          0
# Registers
# R0 =          0, R1 =          1, R2 =          2, R3 =          3, R4 =          4, R5 =          5, R6 =          1, R7 =          2
# R8 =          4, R9 =          2, R10 =          0, R11 =          0, R12 =          0, R13 =          0, R14 =          0, R15 =          0
# R16 =          0, R17 =          0, R18 =          0, R19 =          0, R20 =          0, R21 =          0, R22 =          0, R23 =          0
# R24 =          0, R25 =          0, R26 =          0, R27 =          0, R28 =          0, R29 =          0, R30 =        128, R31 =          0
```

-CO_P3_test_data2

```
# PC =        128
# Data Memory =          0,          0,          0,          0,          0,          0,          0,          0
# Data Memory =          0,          0,          0,          0,          0,          0,          0,          0
# Data Memory =          0,          0,          0,          0,         68,          2,          1,         68
# Data Memory =          2,          1,         68,          4,          3,         16,          0,          0
# Registers
# R0 =          0, R1 =          0, R2 =          5, R3 =          0, R4 =          0, R5 =          0, R6 =          0, R7 =          0
# R8 =          0, R9 =          1, R10 =          0, R11 =          0, R12 =          0, R13 =          0, R14 =          0, R15 =          0
# R16 =          0, R17 =          0, R18 =          0, R19 =          0, R20 =          0, R21 =          0, R22 =          0, R23 =          0
# R24 =          0, R25 =          0, R26 =          0, R27 =          0, R28 =          0, R29 =          0, R30 =        128, R31 =          0
```

Problems you met and solutions:

實作 **jal** 和 **jr** 時，助教說只需要 **1** 個 **MUX 3x1** 就能完成，不過怎麼看都覺得不可能，應該是需要 **2** 個，花了蠻多時間在這一 **part**

Summary:

通過這次的 **Lab** 更加了解 **CPU control unit** 的重要性，任何一個 **control** 只要給錯值就會導致計算結果錯誤