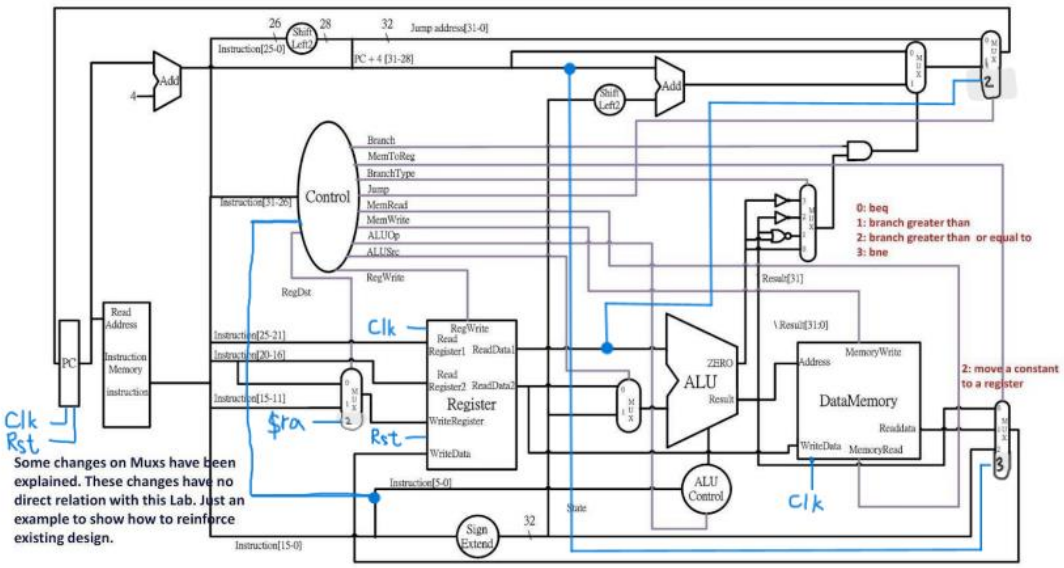
Computer Organization Lab3

Name: 施柏江

ID: 110550108

Architecture diagrams:



Hardware module analysis:

Adder: 負責處理 immediate 指令及計算記憶體位置。

ALU_Ctrl: 根據 opcode 和 ALU_op 來決定要讓 ALU 執行何種運算。

ALU: 負責處理邏輯與加減運算。

Decoder: 為電路中最重要的核心，負責處理各種 control signal。

Instr_Memory: 將 address 轉成對應的 instruction。

ProgramCounter: 指向要執行指令的 address。

Reg_File: 分析此 instruction 需要用到那些 register 的資料。

Shift_Left_Two_32: 將輸入的值左移 2 位。

Sign_Extend: 藉由把 sign bit 延伸到第 17~32 位，將 16bit 的數值延伸到 32bit。

MUX_2to1: 上圖有兩個，一個判斷是否要 jump，一個判斷是 r-format 還是 i-format。

MUX_3to1: 上圖有兩個，一個判斷 destination register 為何處，一個判斷 address 為何處。

MUX_4to1: 上圖有兩個，一個判斷是否要 branch，一個判斷 write data 為何。

Simple_Single_CPU: 將全部的 module 統整再一起，完成一個 CPU。

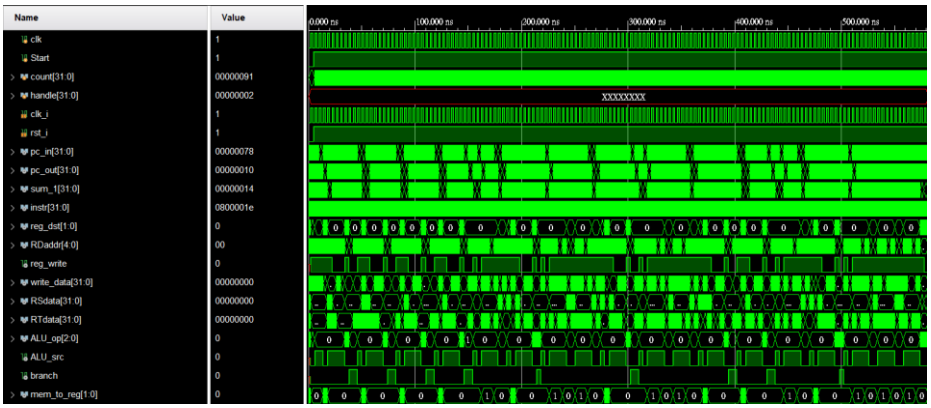
Finished part:

Part 1:

```
- Register File -
 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 =  128 r30 =    0 r31 =    0

- Memory Data -
 m0 =    1  m1 =    2  m2 =    0  m3 =    0
 m4 =    0  m5 =    0  m6 =    0  m7 =    0
 m8 =    0  m9 =    0 m10 =    0 m11 =    0
m12 =    0 m13 =    0 m14 =    0 m15 =    0
m16 =    0 m17 =    0 m18 =    0 m19 =    0
m20 =    0 m21 =    0 m22 =    0 m23 =    0
m24 =    0 m25 =    0 m26 =    0 m27 =    0
m28 =    0 m29 =    0 m30 =    0 m31 =    0
```

Part 2:




```
- Register File -
 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 =  128 r30 =    0 r31 =   16

- Memory Data -
 m0 =    0  m1 =    0  m2 =    0  m3 =    0
 m4 =    0  m5 =    0  m6 =    0  m7 =    0
 m8 =    0  m9 =    0 m10 =    0 m11 =    0
m12 =    0 m13 =    0 m14 =    0 m15 =    0
m16 =    0 m17 =    0 m18 =    0 m19 =    0
m20 =   68 m21 =    2 m22 =    1 m23 =   68
m24 =    2 m25 =    1 m26 =   68 m27 =    4
m28 =    3 m29 =   16 m30 =    0 m31 =    0
```

Problems you met and solutions:

因為提供的檔案只有 MUX_2to1.v，我一開始太侷限於要用 2 to 1 的 MUX 去完成電路，導致我花了許多時間仍然無法完成想要的結果。後來才想到既然圖中有其他種類的 MUX，應該可以把圖上的 2 to 1 MUX 更改成 3 to 1 甚至是 4 to 1，最後終於完成了整個電路。

Summary:

這次的 lab 讓我更深入了解一個能夠處理多種指令的 CPU 內部結構。透過設計各個 module，並將它們整合成一個簡單的單一 CPU，進而更加熟悉它背後的原理，像是如何運用各種 control signal 來執行多樣化的指令。經過這次的 lab，讓我對整個 CPU 的概念有了更清晰的了解。