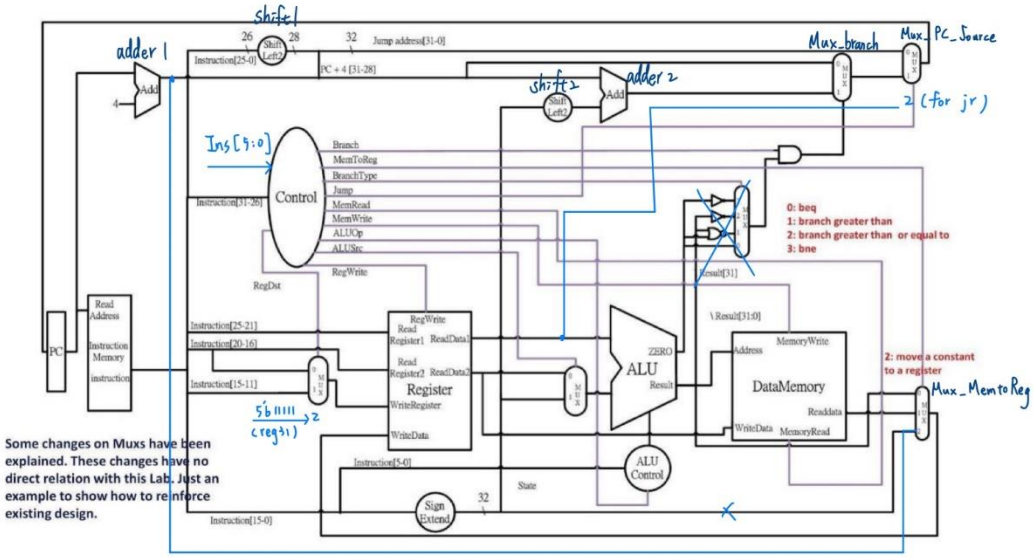
Computer Organization Lab3

Name:尤茂為

ID:110550065

Architecture diagrams:



Hardware module analysis:

	OP	ALUOp	ALUSrc	RegWrite	RegDst	MemRead	MemWrite	MemReg	Br	J
R-format	000000	000	00	1	01	0	0	00	00	01
beq	000100	001	00	0	XX	0	0	XX	01	01
addi	001000	010	01	1	00	0	0	00	00	01
slti	001010	011	01	1	00	0	0	00	00	01
lw	100011	100	01	1	00	1	0	01	00	01
sw	101011	100	01	0	XX	0	1	XX	00	01
jump	000010	101	XX	0	XX	0	0	XX	00	00
jal	000011	101	XX	1	10	0	0	10	00	00
jr				0						10

上圖為根據不同的 op field 透過 decoder 出來的內容，在原本的 decoder 中加入了

MemRead:判斷是否需要從 DataMemory 讀取 data

MemWrite:判斷是否需要將 data 寫入 DataMemory

MemToReg:判斷需要將何種 data 再寫入 Register

Jump:選擇要回傳的 PC 形式

function:用來進一步判斷 R-type 中的 jr, 將其與他種 R-type 運算區別

這次 lab 所提供的 MUX 為 2*1，為了方便 Register 中 WriteReg、WriteData 與 PC 的選擇,，我將其架構改為 3*1，MUX 中的 select_i 也要改為 2bits，所以就在 decoder 中稍做修改，而 MUX_branch 的 select 需要與 1 bit 的 zero 做 and，於

是就將 branch[0]取出與 zero 做 and，而少數幾個 MUX 並不會使用到第三個輸入，這邊一致填入 0;而這次的 lab 並沒有使用到>,>=, bne，於是就將此 4*1MUX 移除。

ADDER1 輸出的值為 PC+4，ADDER2 則為 PC+4+beq 經過 shift 後的值

MUX_PC_Source *(jump)*
Data1:{PC[31:28],address<<2}//j jal
data2:PC+4 or PC+4+beq…//default
Data3:Reg[RS]// jr

MUX_MemToReg *(MemToReg)*
Data1:Result of ALU// R-type I-type
Data2:Data from DataMemory// lw
Data3:PC+4// jal

MUX_Write_Reg *(RegDst)*
Data1:instruction[20:16]//I-type lw
Data2:instruction[15:11]// R-type
Data3:31// jal

Finished part:

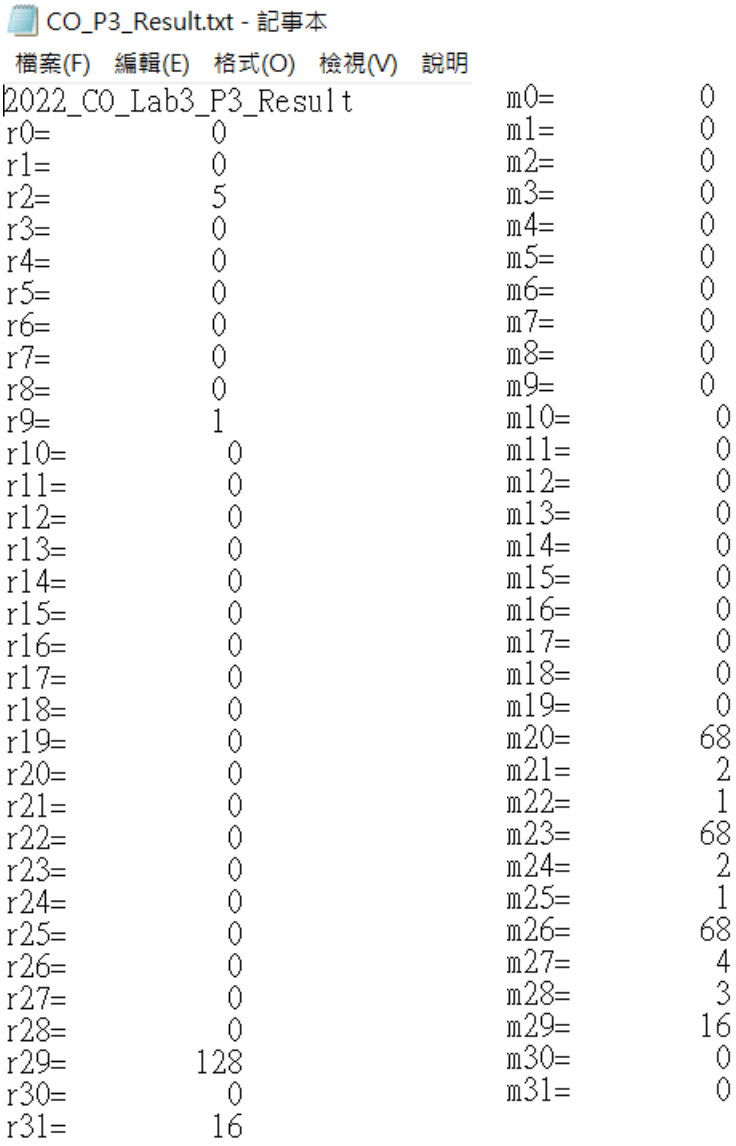
For Part 1:

CO_P3_Result.txt - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
2022_CO_Lab3_P3_Result
r0=          0          m0=          1
r1=          1          m1=          2
r2=          2          m2=          0
r3=          3          m3=          0
r4=          4          m4=          0
r5=          5          m5=          0
r6=          1          m6=          0
r7=          2          m7=          0
r8=          4          m8=          0
r9=          2          m9=          0
r10=         0          m10=         0
r11=         0          m11=         0
r12=         0          m12=         0
r13=         0          m13=         0
r14=         0          m14=         0
r15=         0          m15=         0
r16=         0          m16=         0
r17=         0          m17=         0
r18=         0          m18=         0
r19=         0          m19=         0
r20=         0          m20=         0
r21=         0          m21=         0
r22=         0          m22=         0
r23=         0          m23=         0
r24=         0          m24=         0
r25=         0          m25=         0
r26=         0          m26=         0
r27=         0          m27=         0
r28=         0          m28=         0
r29=       128          m29=         0
r30=         0          m30=         0
r31=         0          m31=         0
```

For Part 2:


CO_P3_Result.txt - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
2022_CO_Lab3_P3_Result
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
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:

只單純用 template code 所提供的 MUX 2 to 1 在實作上需花費更多心思操作，於是稍加更改變成 3 to 1。且求一致性，所有會用到 MUX 的都改成 3 to 1，若 input3 不需使用到且不可能輸出，就填入相對應 bit 數的 0。

在使用到 jal 時須將 return address 存進 reg[31]，原本不太了解這邊的 31 要從哪輸入，上網查了一些資料後才發現原來在 MUX_Write_Reg 的其中一個 input 直接丟 31 進去即可。

Summary:

這次的作業相較於 lab2 難度又上升許多，須對 MIPS 與 CPU 的架構與操作相當熟悉才有辦法完成。在寫這次的作業前，特地回去複習了之前在 CH2 提到的一些相關知識才勉強地寫出這次的作業。