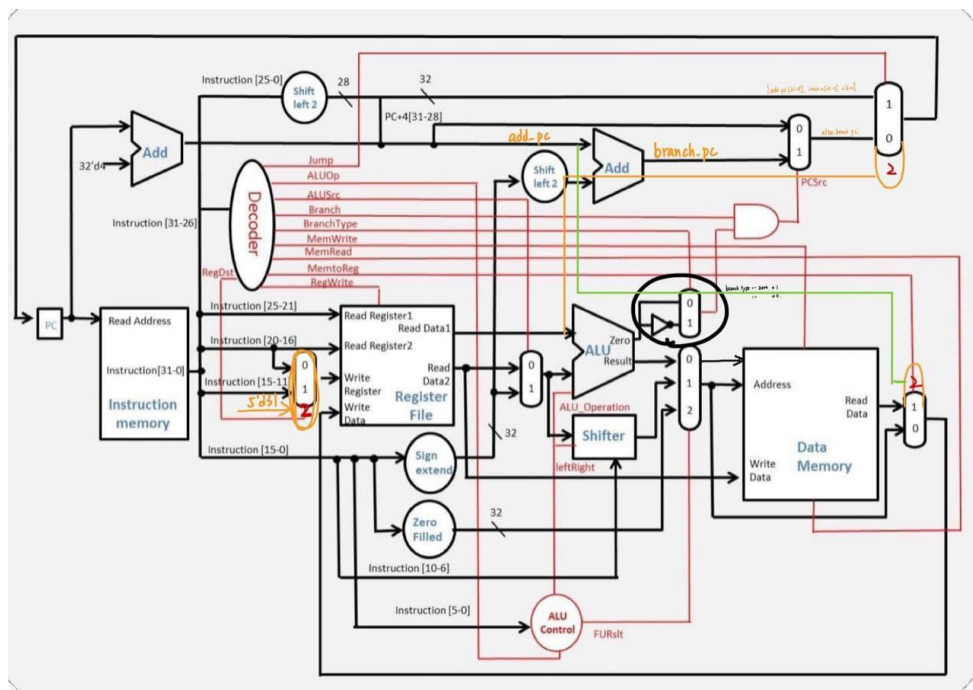


# Computer Organization Lab4

## 1. Architecture diagrams:



## 2. Hardware module analysis:

使用助教給的檔案來實行，上次做的部分在此不額外介紹。

- 原本負責選擇寫入的 reg 的 Mux2to1 改成了 Mux3to1，為了讓 jal 可以把 pc+4 寫入 reg[31] 所做的設計，所以也將 RegDst 改成了 2bits。
- Decoder：這次新增了 Branch\_o、Branch\_type、Jump\_o、MemRead\_o、MemWrite\_o 和 MemtoReg\_o，用來控制 Branch、Jump 和 Memory access 的部分。
- Data\_Memory：以 MemWrite 和 MemRead 做為控制訊號，負責讀取或寫入 Memory。以 rs 和 offset 相加所得到的值為 addr，rt 做為 sw 的 data。
- Shifter Left\_two：在處理 pc 的部分以 Shifter 為 base 去做 shift left two。
- Adder Adder2：算 branch 的 pc。
- Mux2to1 PC\_after\_branch：以 branch 和 branch type 處理過後當作控制訊號，去選擇是否要做 branch。
- Mux3to1 PC\_final：以 Jump 為控制訊號，去選擇輸出為 branch 還是 jump(or jal)還是 rs(jr)。
- Mux3to1 after\_jal：以 MemtoReg 為控制訊號，看要將前面所得出的 write data 還是 Mem data 還是 pc+4 送入 reg。

### 3. Finished part:

全部的要求皆完成。

主要處理的地方為 Decoder 的控制訊號，比較特別的部分有 Reg File[29]要設成 128，RegDst 改成了 2bits，把 10 設給 jal 做使用，Jump 也設了 2bits 以區分要跳的位置的來源為 address(or jal)還是 rs(jr)，Branch 用來區分是不是這類型，Branch\_type 再細分種類(以 zero 為 0 或 1 做分界)，再將 ALU\_Ctrl 裡的 op 設好並配合 ALU 做剩餘判斷。

```

C:\Users\User\Desktop\code\iverilog -o lab4 TestBench.v
WARNING: TestBench.v:139: $readmemb(C0_F4_test_data1.txt): Not enough words in the file for the requested
range [0:31].
=====
a) basic score: 75 / 75
=====
Congratulation, You pass TA's pattern
TestBench.v:599: $stop called at 5720000 (ips)
** FPP Stop() **
** Flushing output streams.
** Current simulation time is 5720000 ticks.
. finish
. Continue **

C:\Users\User\Desktop\code\iverilog -o lab4 TestBench.v
C:\Users\User\Desktop\code\comp lab4
=====
b) advance set1: 15 / 15
=====
Congratulation, You pass TA's pattern
TestBench.v:599: $stop called at 5720000 (ips)
** FPP Stop() **
** Flushing output streams.
** Current simulation time is 5720000 ticks.
. finish
. Continue **

C:\Users\User\Desktop\code\iverilog -o lab4 TestBench.v
C:\Users\User\Desktop\code\comp lab4
WARNING: TestBench.v:141: $readmemb(C0_F4_test_data2.txt): Not enough words in the file for the requested
range [0:31].
=====
c) advance set2: 10 / 10
=====
Congratulation, You pass TA's pattern
TestBench.v:599: $stop called at 180000 (ips)
** FPP Stop() **
** Flushing output streams.
** Current simulation time is 180000 ticks.
. finish
. Continue **
```

### 4. Problems you met and solutions:

一開始有點想不到 jal 和 jr 怎麼寫，blt、bgez、bnez 也有思考一下要怎麼分辨，後來再多想想和看說明就行了，最主要 debug 很久的地方是自己抄錯 instr\_op，花了很多很多時間在上面找不出錯的原因，後來改掉後就順利通過測資了。

### 5. Summary:

這次的 simple cycle CPU 更完整了，讓自己更加了解原本其實沒那麼熟的 jal、jr、lw 和 sw，也觸及到了 Data Memory。做出來的那瞬間很有成就感，只是中間覺得自己架構沒什麼問題又 debug 不出來的時候很懷疑人生。做出來真是太好了。