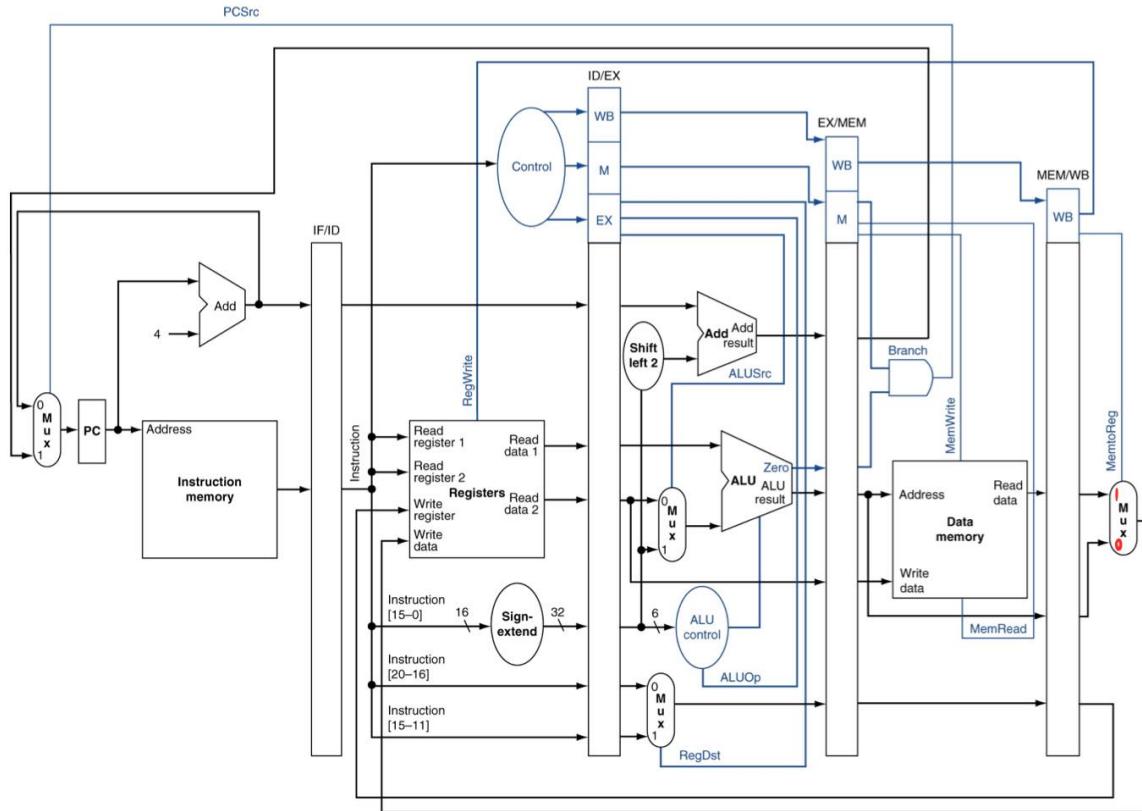


# Computer Organization Lab4

Name: 郭昀 ID: 109550018

## Architecture diagrams:

我這次都是按照 spec 中給的 Architecture diagram 去實作的，唯一不同的是如同助教在 spec 中所說的，我更改了最右邊 MUX 的順序，讓 control input 為 0 或 1 時的輸出順序相反。另外也在各個 stage 之間加了 register。



## Hardware module analysis:

這次的 Lab 跟上次相比少了一些 branch 相關的指令，所以原本 Lab3 有做的一些 MUX 就用不到了，主要就是接 Pipe\_CPU1 裡面的線路，其他幾乎都是沿用 Lab3 的或是做一些小更改。

## Finished part:

Test 1 :

Register=

r0=	0, r1=	3, r2=	4, r3=	1, r4=	6, r5=	2, r6=	7, r7=	1
r8=	1, r9=	0, r10=	3, 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=	0, r31=	0

Memory =

m0=	0, m1=	3, 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
r16=	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



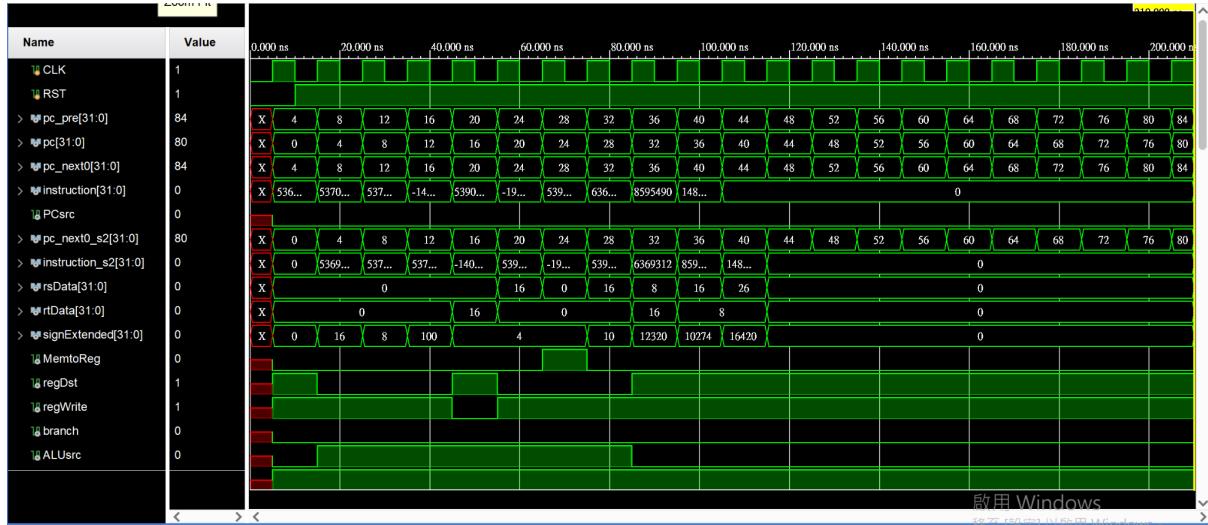
## Test 2 :

Register=

r0=	0, r1=	16, r2=	20, r3=	8, r4=	16, r5=	8, r6=	24, r7=	26
r8=	8, r9=	100, 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=	0, r31=	0

Memory

<code>m0=</code>	<code>0, m1=</code>	<code>16, m2=</code>	<code>0, m3=</code>	<code>0, m4=</code>	<code>0, m5=</code>	<code>0, m6=</code>	<code>0, m7=</code>	<code>0</code>
<code>m8=</code>	<code>0, m9=</code>	<code>0, m10=</code>	<code>0, m11=</code>	<code>0, m12=</code>	<code>0, m13=</code>	<code>0, m14=</code>	<code>0, m15=</code>	<code>0</code>
<code>r16=</code>	<code>0, m17=</code>	<code>0, m18=</code>	<code>0, m19=</code>	<code>0, m20=</code>	<code>0, m21=</code>	<code>0, m22=</code>	<code>0, m23=</code>	<code>0</code>
<code>m24=</code>	<code>0, m25=</code>	<code>0, m26=</code>	<code>0, m27=</code>	<code>0, m28=</code>	<code>0, m29=</code>	<code>0, m30=</code>	<code>0, m31=</code>	<code>0</code>



Test 1 的結果跟解答一樣所以應該是沒有問題，然後我對 Test 2 的 machine code 有做了如下的重新排序，讓每一個 hazard 都至少間隔 2 個指令以上，然後結果也成功跟助教給的解答相同，另外，我也在 Testbench 中加入了 Pipe\_CPU 讓波形圖輸出這些訊號，然後看起來也沒也沒有問題！

-----  
Reorder machine codes :

```

addi    $1, $0, 16
addi    $3, $0, 8
addi    $9, $0, 100
sw      $1, 4($0)
addi    $2, $1, 4
lw      $4, 4($0)
addi    $7, $1, 10
add    $6, $3, $1
sub    $5, $4, $3
and    $8, $7, $3
-----
```

## Problems you met and solutions:

在 run test 1 的時候我發現 sw \$1, 4(\$0) 會一直讀不到 \$1 的值，後來在問過同學後發現很多人都有這個問題，於是我們上網查才發現，原來在 Reg\_File 中，如果是透過 RDaddr / RDdata 儲存資料，要在下一個 clock 才能從 RS / RT 正常讀取，所以需要判斷 RSaddr / RTaddr 是否跟 RDaddr 相同，如果相同就直接輸出 RDdata。

## Summary:

因為這次的主要是根據 Lab3 的內容去延伸實做的，所以比較複雜的像是 Decoder 的訊號分析在上次就做完了，這次就比較不用特別研究，只要研究要怎麼使用 Pipe\_Reg 就可以了，然後根據助教的命名方式，我覺得助教可能想要我們全部寫在同一個就可以了，所以我就用 {} 把兩個 stage 間要傳的訊號接在一起了，然後我覺得因為這次的 module 跟訊號都比上次多很多，所以我覺得這次很考驗寫程式的細心度，像我不小心打錯名字就 debug 很久...