LAB 4 - Pipeline CPU I

1. 系統架構：

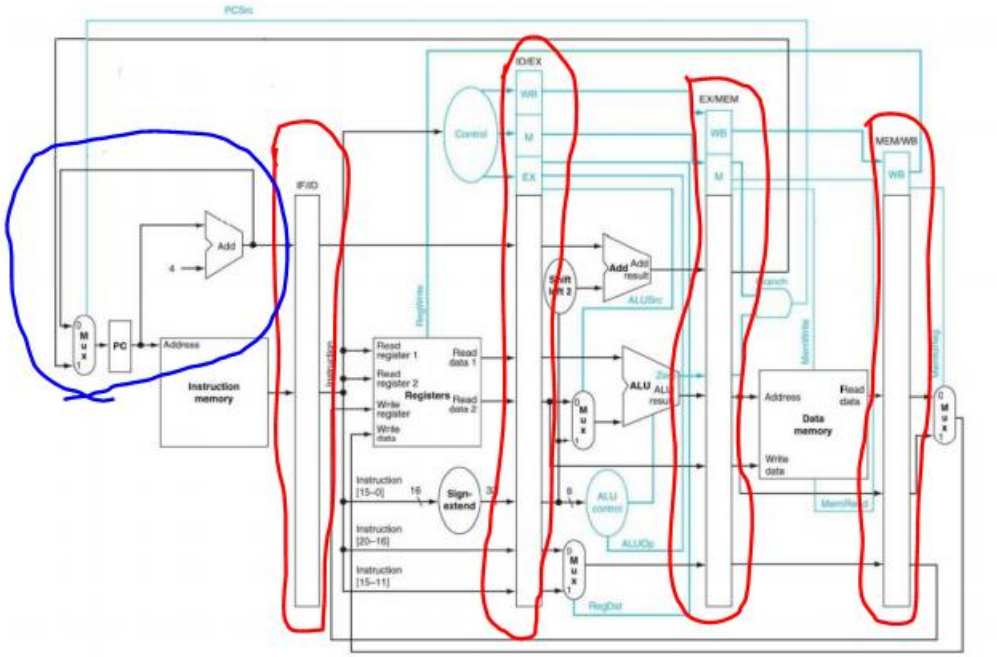
加入 pipeline 的想法，讓 CPU 更有效率。

2. 設計模組分析、設計結果：

藍色圓圈的地方，讓 PC 直接加 4，不用等 instruction 全部跑完，才能跑下一個。而是把 CPU 分成 5 個 state：

IF → ID → EX → MEM → WB

State 的中間則以 register(紅色圓圈)去儲存之後可能要用到的 data。就能以 state 為單位執行 instruction 了



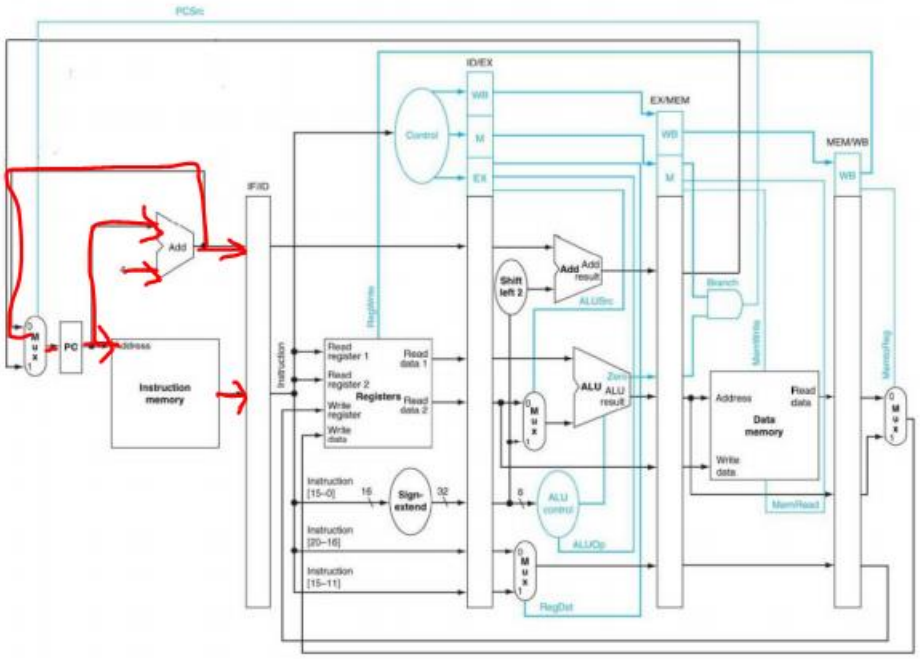
Example：

Test1.

(1) Addi $1, $0, 3 ;

將 PC 輸入 instrction memory，並把讀取出的 instruction 暫存在 IF/ID Register 中。此外 PC+4 之後立刻傳回，才能在下一個 cycle 讀取下一個 instruction。並將 PC+4 暫存在 IF/ID Register 中。

(1)

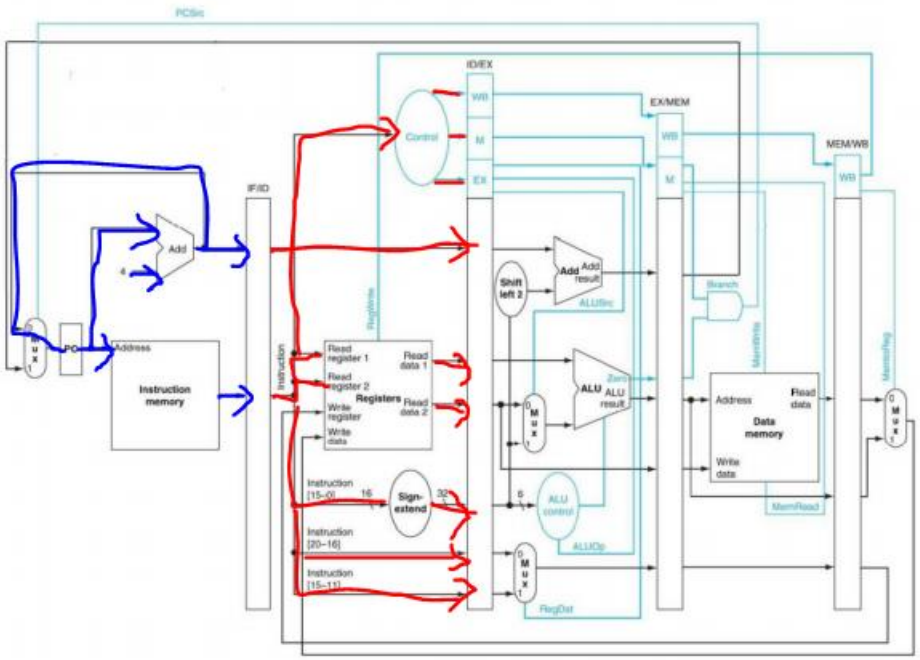


(2) Addi $2, $0, 4 ;

下一個 PC 輸出→下一個 instriction 存入 IF/ID Register，再把 PC+4 回傳。

IF/ID Register 輸出原本儲存的 instruction 進行 Register 讀取、Decoder，並儲存在 ID/EX Register。

(2) (1)

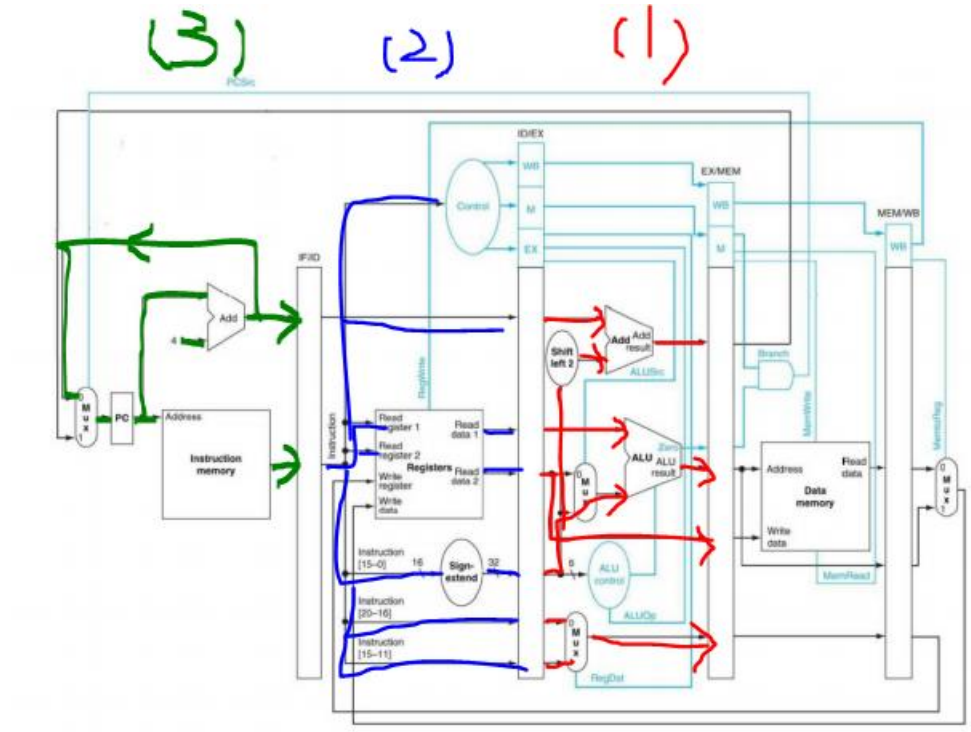


(3) Addi $3, $0, 1;

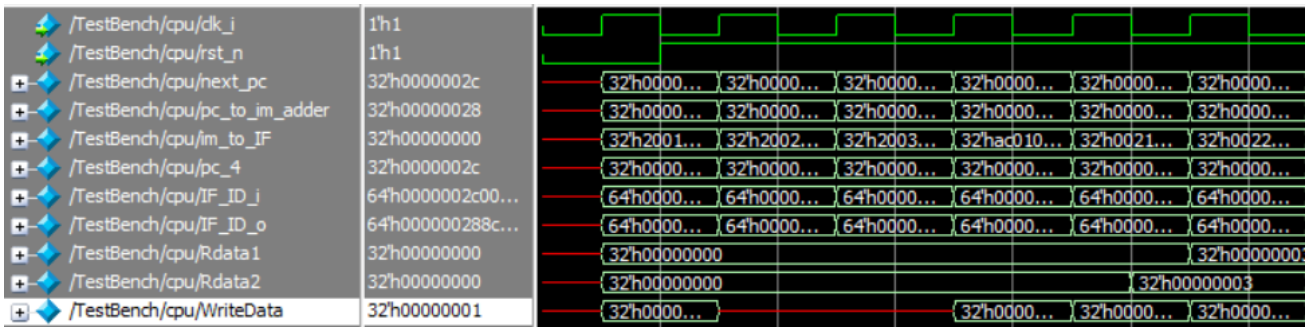
PC 輸出→下一個 instruction 產生並存至 IF/ID Register，PC+4 回傳。

IF/ID Register 輸出讀取 Register 的值和 Control decode。並傳入 ID/EX Register 儲存。

ID/EX Register 輸出 Rs, Rt 的值，送入 ALU 進行運算再傳入 EX/MEM Register。



以此類推，Pipeline 的功能就是分段、同時處理。




```
# 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=       128, r30=         0, r31=         0
```

3. 遭遇的困難與解決方法：

一開始連 PC 都跑不進去，以為是哪條線沒有接好，一條一條寫下來慢慢對，還是沒有發現錯誤。後來才突然看到是 rst_n 寫成 rst_i。

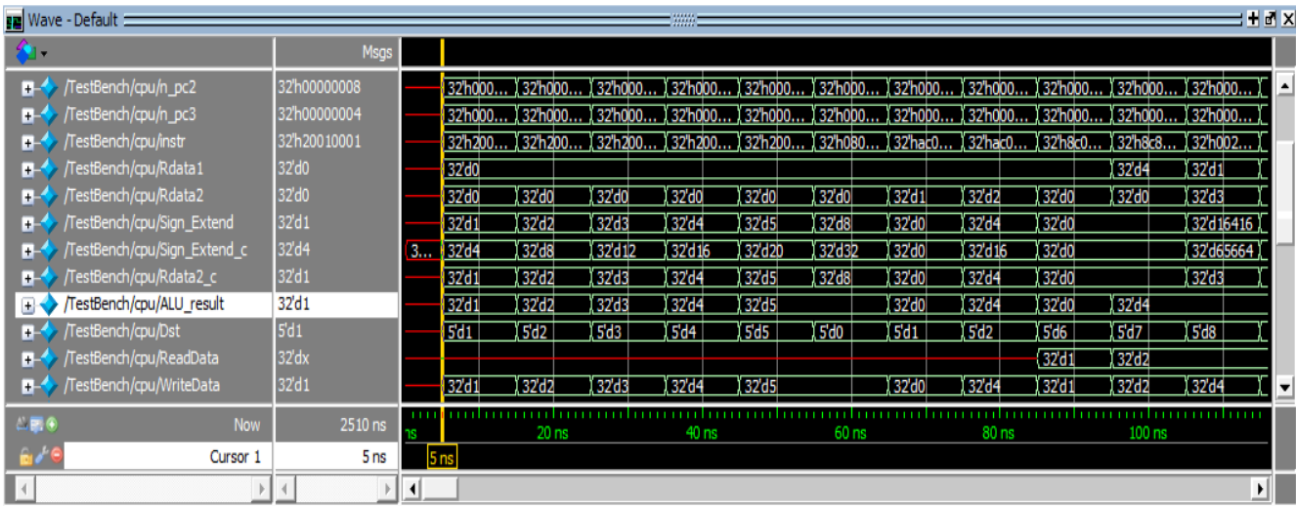
PC 跑進去之後，r0 的值一直會被改掉，發現是 WriteData = xxxxxxxxxx 的時候 r0 會被改寫，RegWrite 從一開始就是 1 了，還沒等到第一步要寫入的時候 RegWrite 就已經回傳 1，才導致後面的錯誤。後來把 ALU Default 時也 output 出 0 才不會被改寫。

因為 Pipeline 的 intruction 是一直接下去的，所以看波形圖的時候還要算到底甚麼時候才到某個 state。

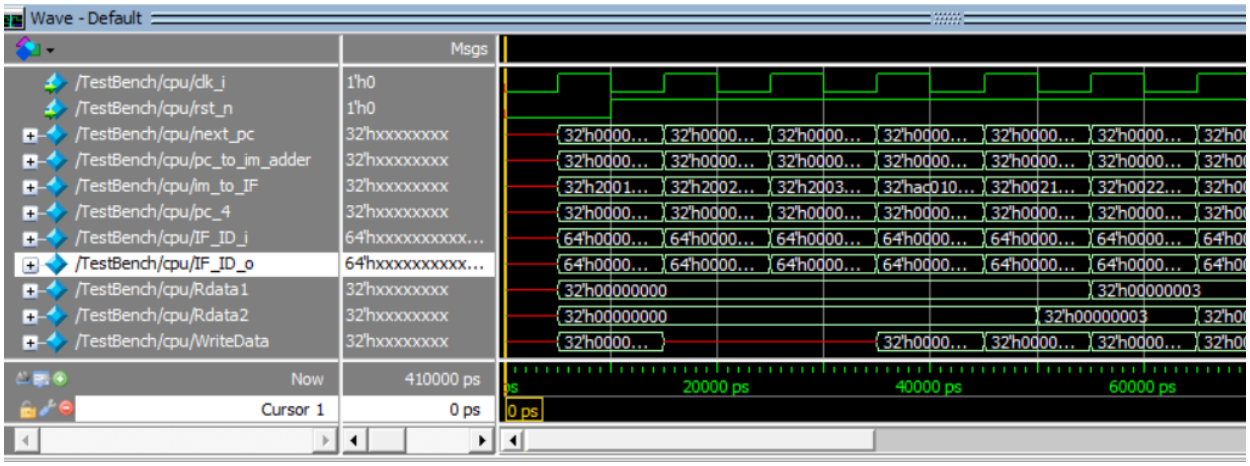
4. 作業心得討論：

看波形圖的時候才深刻感受到 pipeline 跟沒有 pipeline 的差別：

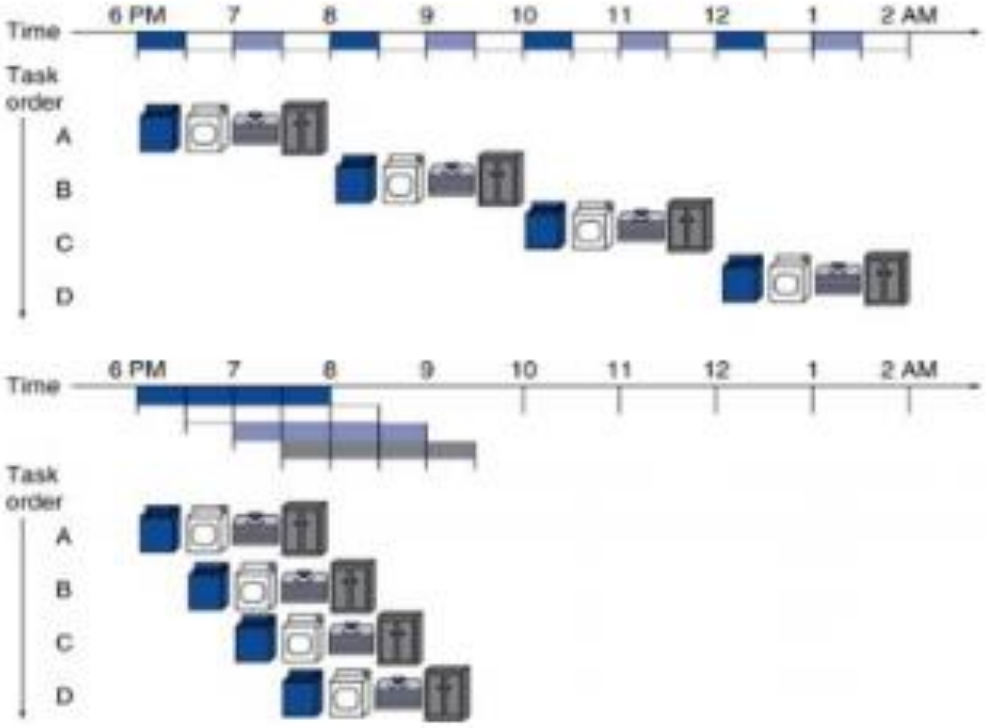
沒有 pipeline：



有 pipeline：



有 pipeline 的速度比較快，可以清楚的看到每個 state 的都在執行不一樣的 instruction。沒有 pipeline 就是規規矩矩的一個一個執行。


Time
6 PM
7
8
9
10
11
12
1
2 AM
Task order
A
B
C
D
Time
6 PM
7
8
9
10
11
12
1
2 AM
Task order
A
B
C
D