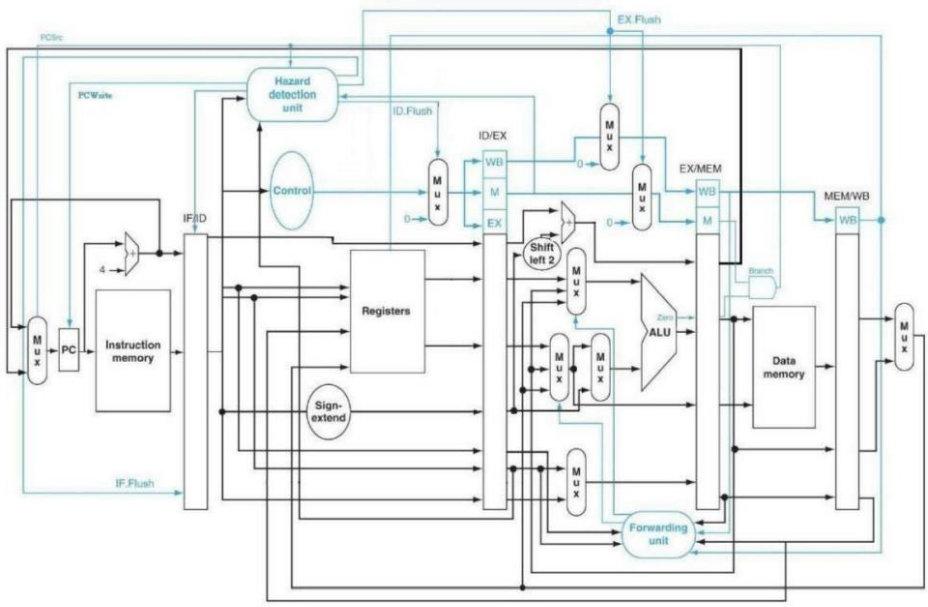
Computer Organization Lab5

Name:黃芷柔

ID:110550142

Architecture diagrams:



Hardware module analysis:

在 lab5 中，使用了 lab4 的 Adder, MUX_2to1, MUX_4to1, Decoder, Sign-extend, Shift left 2, ALU control, ALU，新增了 Hazard Detection 跟 Forwarding Unit。

Hazard Detection:

偵測是否有 Hazard。如果 branch 有 taken 的話，就把 IF/ID, ID/EX, EX/MEM 的 pipe register 全部 flush；如果 D/EX.MemRead and ((ID/EX.RegisterRt = IF/ID.RegisterRs) or (ID/EX.RegisterRt = IF/ID.RegisterRt))，就 flush ID/EX 的 pipe register，也就是 stall 一個 clock cycle。

Forwarding Unit:

If (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd = ID/EX.RegisterRs))，forwardA=10

if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd = ID/EX.RegisterRt))，forwardB=10

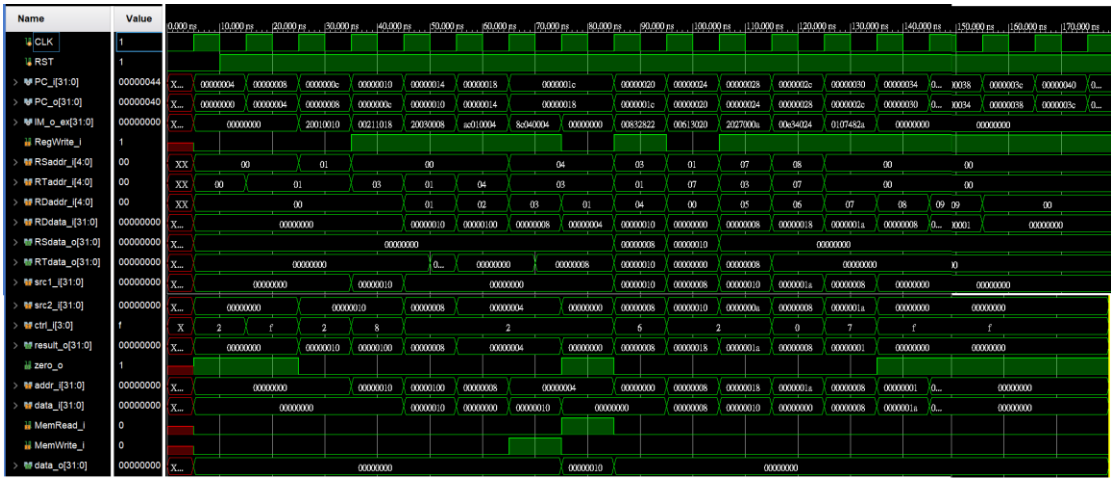
if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and (MEM/WB.RegisterRd = ID/EX.RegisterRs))，ForwardA=01
if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB = 01
其他情況下 Forward=00
在 EX，如果 Forward 是 10，就輸入上個 ALU 的結果(在 MEM)；如果 Forward 是 01，就輸入 WB RegisterRd 的 data。

優點:CPU 可以自己解決 hazard 的問題
缺點:由於 decoder 是沿用 lab3，有設計 jump 需要的輸出，在 lab5 不需要寫 jump，所以浪費了一些沒用到的 bit。

Finished part:

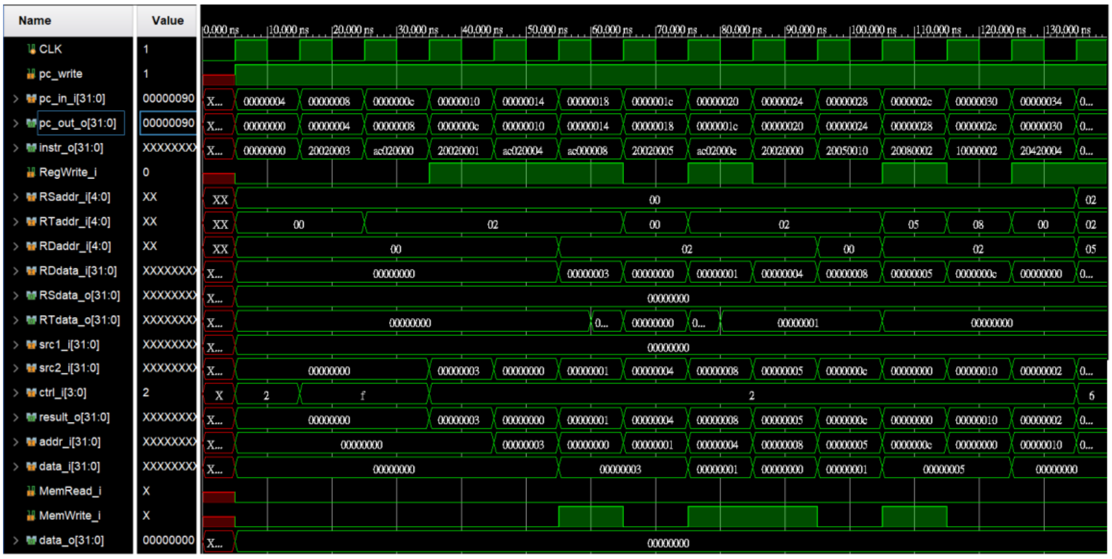
Testcase1:

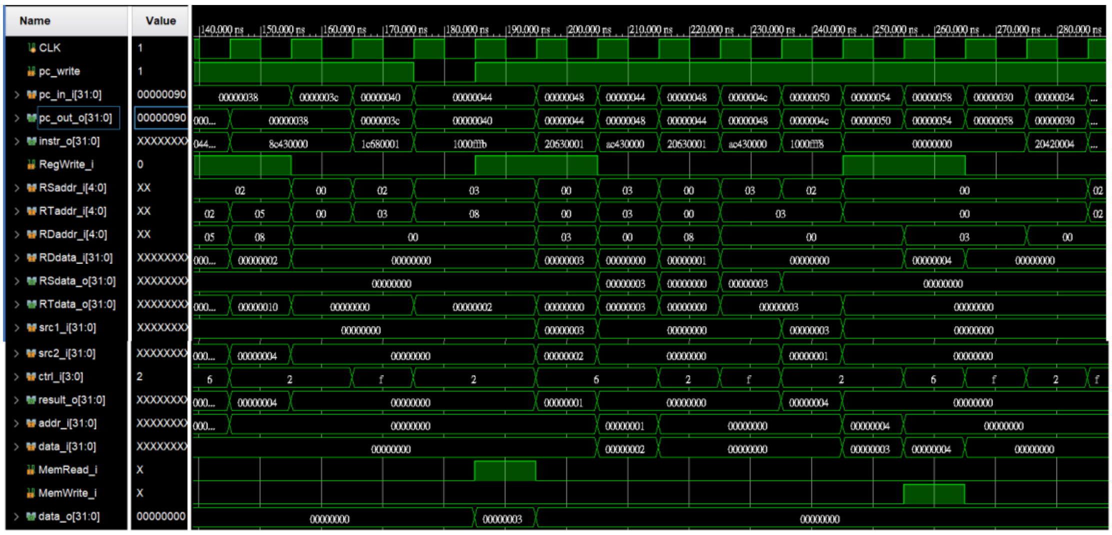
```
################################## clk_count = 17####################################
=====================================Register=======================================
r0 =    0, r1 =   16, r2 =  256, r3 =    8, r4 =   16, r5 =    8, r6 =   24, r7 =   26
r8 =    8, r9 =    1, r10=    0, r11=    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=    0, r30=    0, r31=    0
=====================================Memory=========================================
m0 =    0, m1 =   16, 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
```

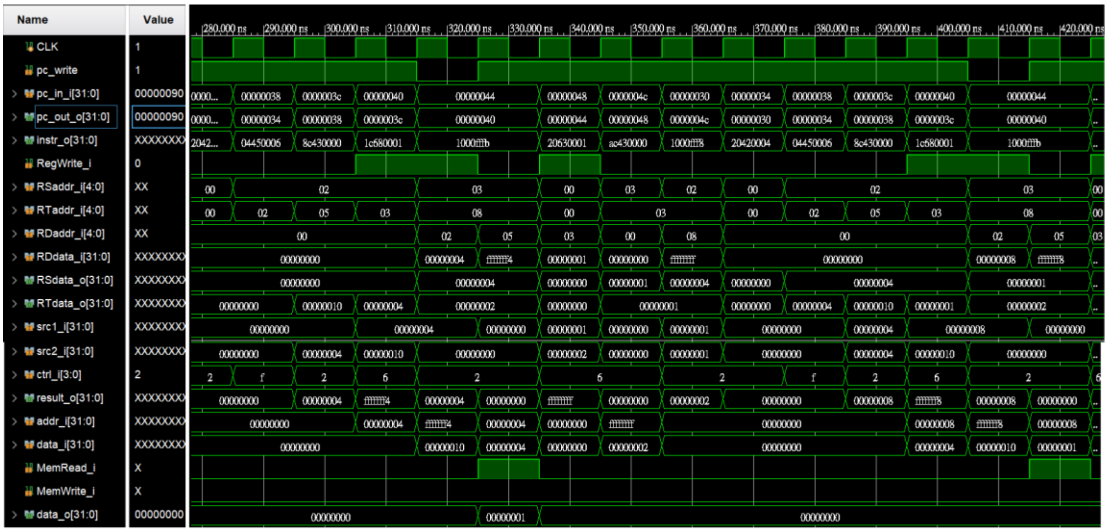


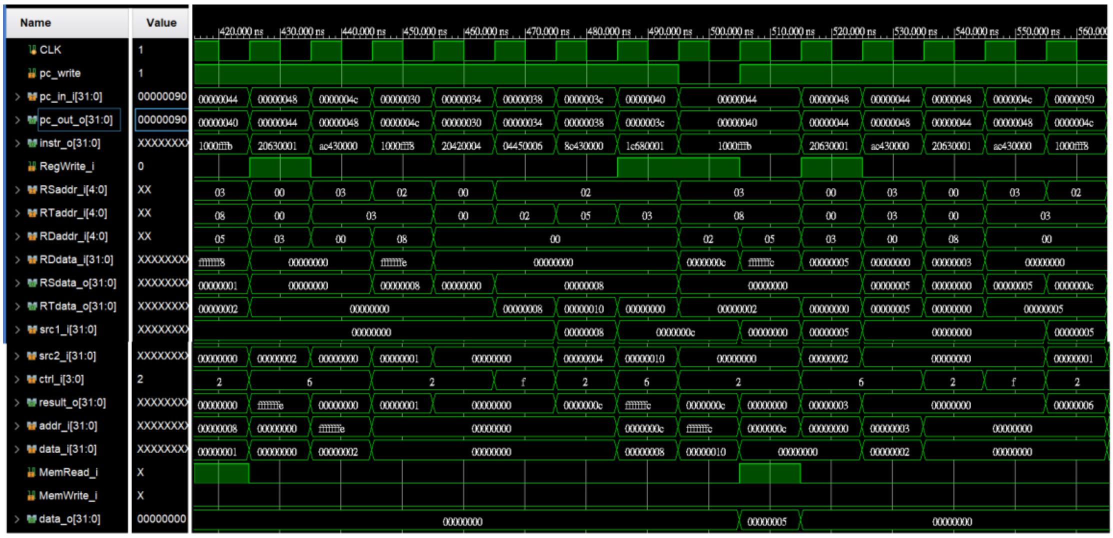
Testcase2

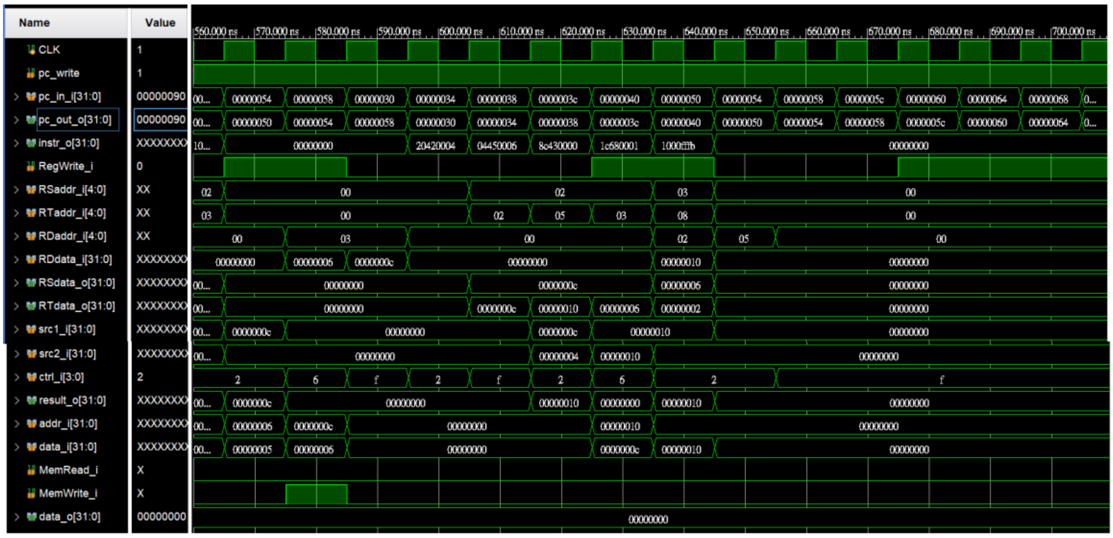
```
################################## clk_count = 70####################################
======================================Register======================================
r0 =    0, r1 =    0, r2 =   16, r3 =    6, r4 =    0, r5 =   16, r6 =    0, r7 =    0

r8 =    2, r9 =    0, 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=======================================
m0 =    4, m1 =    1, m2 =    0, m3 =    6, 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
```











兩個 testcase 都跟提供的解答一樣。

Problems you met and solutions:

在測試 testcase2 時，遇到 r2 大於 r5 還一直累加的問題，後來發現問題出在 hazard 沒有把 branch 跟 load-use 分開。除此之外還發現 decoder beq/ bne/ bge/ bgt 沒有寫得很完整，透過這次 lab 修正。

Summary:

實做過 lab 後，我對 Pipelined CPU 解決 Hazard 的方式更加熟悉。