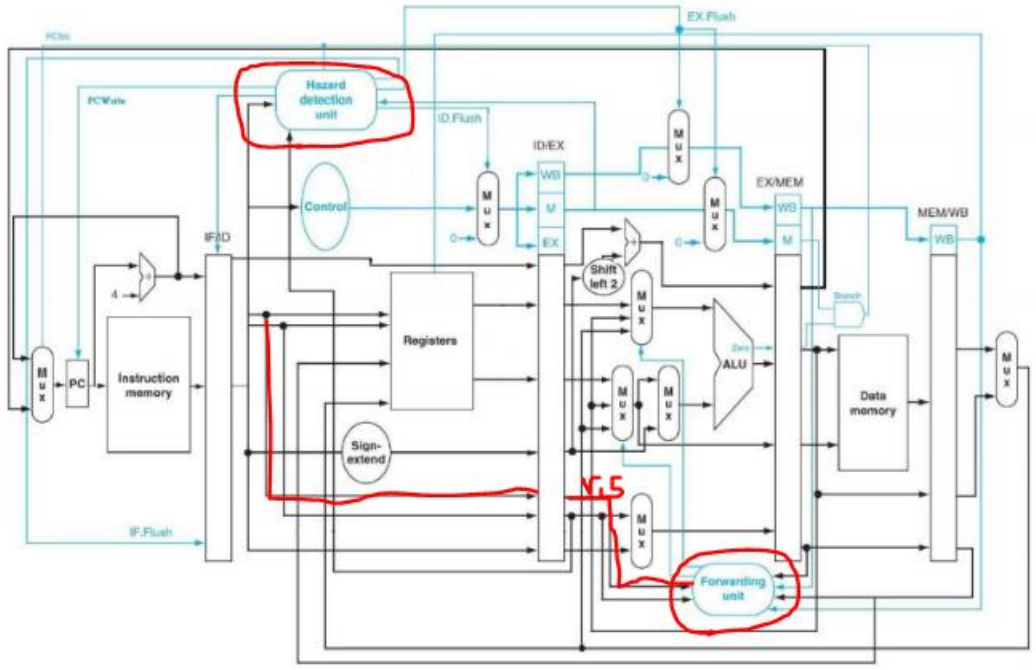
Computer Organization - Lab5

1. 系統架構：

以 lab4 為基礎下，新增 Forwarding Unit 和 Hazard Detection Unit，來解決 Data Hazard 的問題。並讓 Pipeline Register 多存 rs 的 address(用來判斷 Data Dependency)



2. 設計模組分析、設計結果：

Forwarding Unit：

Input：

Address：EX_rs, EX_rt, MEM_rd, WB_rd

Control：MEM_RegWrite, WB_RegWrite

Output：

Control：A_o, B_o

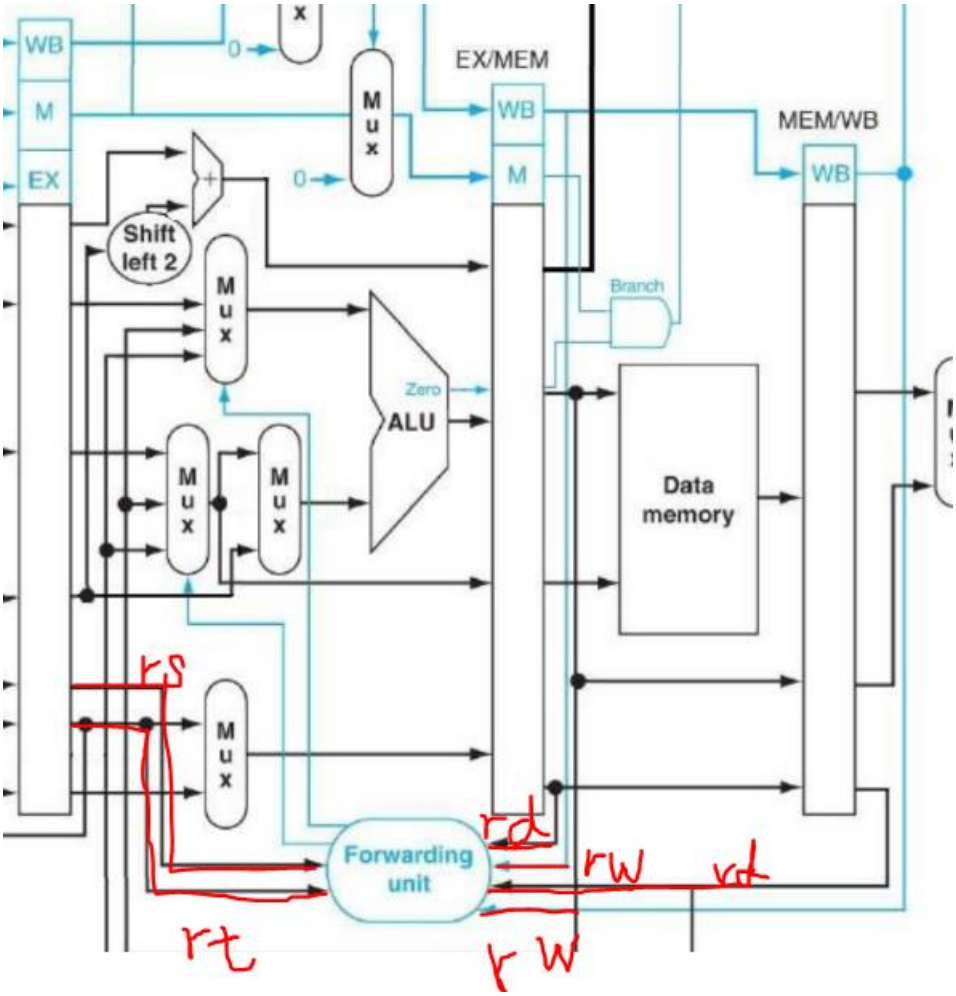
概念：

A_o：

1. 需先判斷 MEM_RegWrite 和 WB_RegWrite 是否開啟，如果沒開，data 就不會被更新，故不必更換 data。
2. 若 EX 層的 rs_address 和 MEM 層的 rd_address(WB 層的 rd_address) 相同，就把 EX 層的 rs_data 改成 MEM 層的 ALU_result(WB 層的 Writedata)。

3.若 MEM 層和 WB 層都符合上述條件，選 MEM 層(data 最新)。

B_o：把上述之 rs 改成 rt 即可。



Hazard Detection Unit：

Input：

Address：ID_rs, ID_rt, EX_rt

Control：EX_Memread, PCSrc

Output：

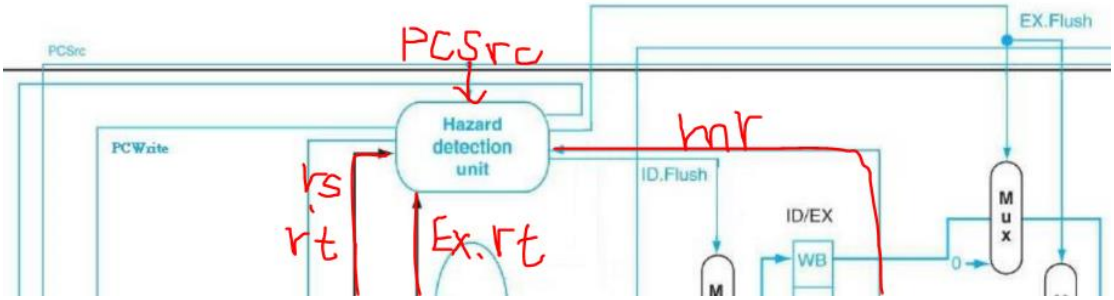
Control：IF_ID_write, PCWrite, IF_ID_flush, IF_flush, ID_flush, EX_flush

概念：

當 Load 指令後面緊跟著一個指令，裡面恰好需要用到 Load 指令中的 Register。這種情況是 Forwarding Unit 無法解決的，因為 Forwarding Unit 判斷 MEM 層的時候，還未從 Memory 中讀取 data，故需要用 Hazard Detection Unit 來 Stall 一個 cycle。

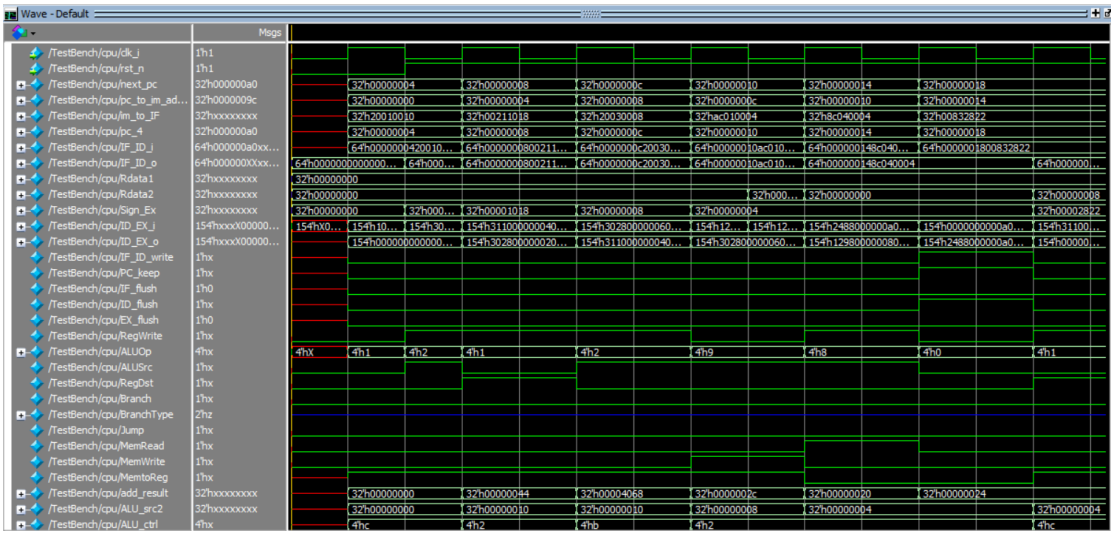
1. 先判斷 EX 層的 MemRead 是否開啟，只有 lw 指令會開啟，並造成 Load-Use Data Hazard。

2. 判斷 ID 層的 rs 或 rt 是否跟 EX 層的 rt 一樣。
3. 若符合上述條件，就讓 Output 的 Control 讓 PC、IF_ID_Pipeline_Register 維持原狀，並把所有的 Control 都關閉。



設計結果：

Test：




```
# 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=        128, 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
#
# 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
```

3. 遭遇困難與解決方法：

第一次寫出 Forwarding Unit 的時候，A_o 和 B_o 一直維持原狀，後來重打一次之後，然後把 Pipe_CPU 的線重新檢查一次，發現是 rs_address 錯誤，因為一開始 ID_EX_pipe_reg 就沒有存 rs_address 的值，所以就新拉一條線儲存 rs_address。

跑測資的時候，遇到 lw 就會直接跳過，而不是等他跑完在繼續下一個指令，後來再把 Hazard Detection Unit 裡面的線一條一條檢查，後來才看到是沒有把之後的 control 都變成 0。

4. 作業心得討論

在考第二次期中考的時候，就對 Hazard Detection Unit 有點陌生，不太知道他的用法。打完這次的 lab 真的很清楚 Hazard Detection Unit 和 Forwarding Unit 是怎麼合作解決 Data Dependency 造成的 Data Hazard。