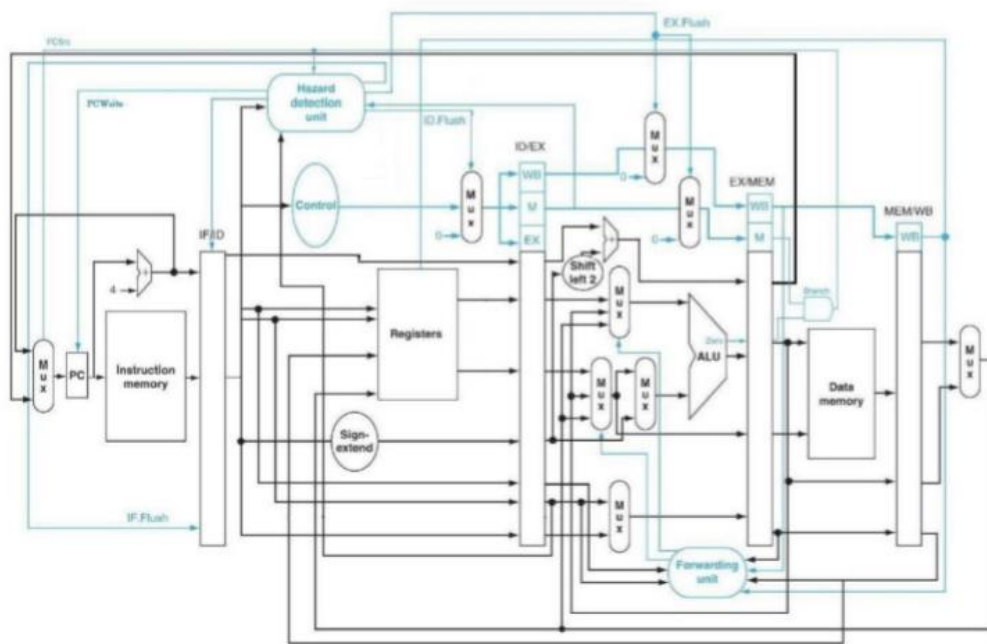


My architecture:



跟上圖一樣

Hardware Module Analysis :

這次新增了兩個 module 分別是 forwarding unit 和 hazard detect unit，還有微調了 pipe_reg 裡面的線。

Forwarding unit: 用來將一些可以在寫回 reg 前就知道的值直接接到下一個 instruction，可以減少 stall 的發生。

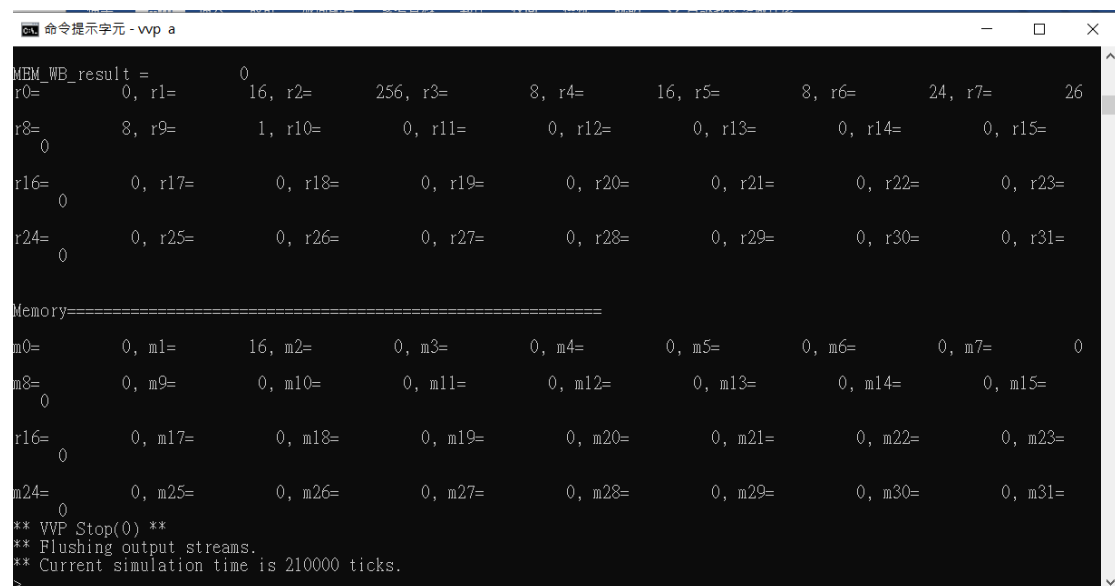
Hazard detect unit: 用來判斷是否有 data hazard 或者是 branch hazard 的發生，再將判斷的結果傳到 forward unit 來 forwarding。

Problems You Met and Solutions :

這次遇到的問題主要還是跟往常一樣，是在接線的部分上，新增的兩個 module 都不是太難寫，只要把 conditions 照講義上寫好就

好了，因此我卡最久的地方還是接線，還有一個是為 **pipe reg** 新增了一個 **flush** 的指令，只要 **flush** 是 **1** 就清空資料，這個我也是想一陣子才想到要如何很好的時做出 **flush** 這個功能，還有最大的難關是在加分題的部分，因為又多了 **4** 種新指令，因此整個設計又更繁瑣了一點，我因此新增了 **4to1** 的 **mux** 來解決，在想出解決方法的時候遭遇了滿多困難的，但還好線並不是很難接，因此有了方法很快就做完了，但因為不知道有沒有成功解決 **branch hazard**，因此我又自己設計了一套 **instruction** 來檢查，檢查結果是對的時候十分開心，希望助教測的時候也是對的~

Result :



```
命令提示字元 - vvp a
MEM_WB_result =      0
r0=      0, r1=     16, r2=    256, r3=      8, r4=     16, r5=      8, r6=     24, r7=     26
r8=      0, r9=      1, r10=     0, r11=     0, r12=     0, r13=     0, r14=     0, r15=
r16=     0, r17=     0, r18=     0, r19=     0, r20=     0, r21=     0, r22=     0, r23=
r24=     0, r25=     0, r26=     0, r27=     0, r28=     0, r29=     0, r30=     0, r31=

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=
r16=     0, m17=     0, m18=     0, m19=     0, m20=     0, m21=     0, m22=     0, m23=
m24=     0, m25=     0, m26=     0, m27=     0, m28=     0, m29=     0, m30=     0, m31=
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 210000 ticks.
```

如上圖，與正確答案一樣。

Summary :

這次的 lab 是上一次 lab 的延伸，主要是要我們實作解決 **hazard**

的方法，而因為只多了解決 hazard 因此需要改的地方並不多，就新增兩個 module 而已，但接線的部分就有點頭疼了，新增了滿多的 mux 和一些 module，整體的複雜度又更提升了，因此在接時會有點眼花撩亂，變數的名稱和數量也變多，我剛開始接的時候還忘記我上個 lab 的變數名稱是甚麼意思，因此花了好一功夫才理解，早知道上次不要亂取變數名子，總體來說這次的 lab 也是很有收穫的，讓我寫 verilog 的技術又提升了不少，寫完後有點後悔應該期末考前寫的，因為寫完後我對整個 pipeline cpu 的理解又更上一層樓了，以前懵懵懂懂的部分也因為要實做出來必須要搞懂，所以觀念清楚了，又是十分有收穫的一次 lab!