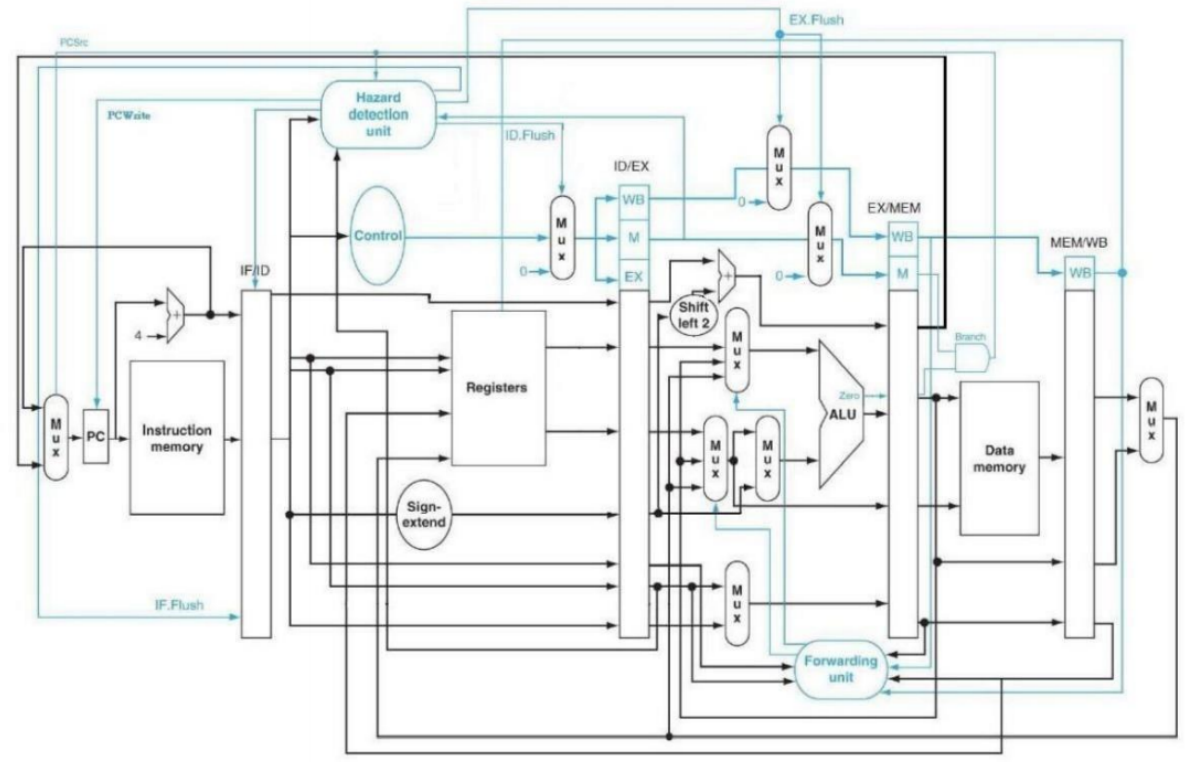
Computer Organization
COLab5 report

Architecture diagrams:



Hardware module analysis:
此次作業要解決data dependency的問題，多數module繼續沿用Lab4。
HazardDetection.v:圖上有五個output，當指令之間存在data dependence由PC_Write.IF_ID_Write.IF_Flush.ID_Flush_EX_Flush做以下控制:

1. load-use hazard: 當load指令後面緊跟著一樣的指令，所要讀取的Rs正好等於lw指令中的Rt，藉由(EX_MemRead) && ((EX_Rt == ID_Rs) || (EX_Rt == ID_Rt))這行判斷，當此情形發生時，避免資料來不及輸入到ALU，因此需要stall，暫停pipeline中前兩個階段的指令，使lw指令能在wb階段可以forwarding到所需指令。
2. PCSrc==1時表示做branch，利用Flush清除前三個指令。

Forwarding.v:根據講義ch4 p.77，forwarding unit控制ALU的兩個source，偵測目前指令之Rd與下一個指令及下下一個指令之Rs或Rt是否相同(data dependence)，當資料相依發生在EX/MEMRegisterRd階段forwarding=01，當發生在MEM/WBRegisterRd則forwarding=10。

Problems You Met and Solutions:
原本很苦惱hazardDetection要如何實作，透過上網找資料以及對照講義有想明白。

Result:
CO_P5_test_data1.txt

```
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

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
$stop called at time : 210 ns : File "C:/Users/GameToGo/Desktop/myLab5/CAhw5/CO_Lab5_v3/TestBench.v" Line 55
```

Summary:

了解當指令存在資料相依時，使用硬體的方式forwarding. hazard detection. stall(load-use hazard)來達成指令的正確結果，在還沒查資料前真的覺得好難喔，尤其作stall的部分，但當實際跑出正確值很欣慰。