Computer Organization

0516013 吳泓寬

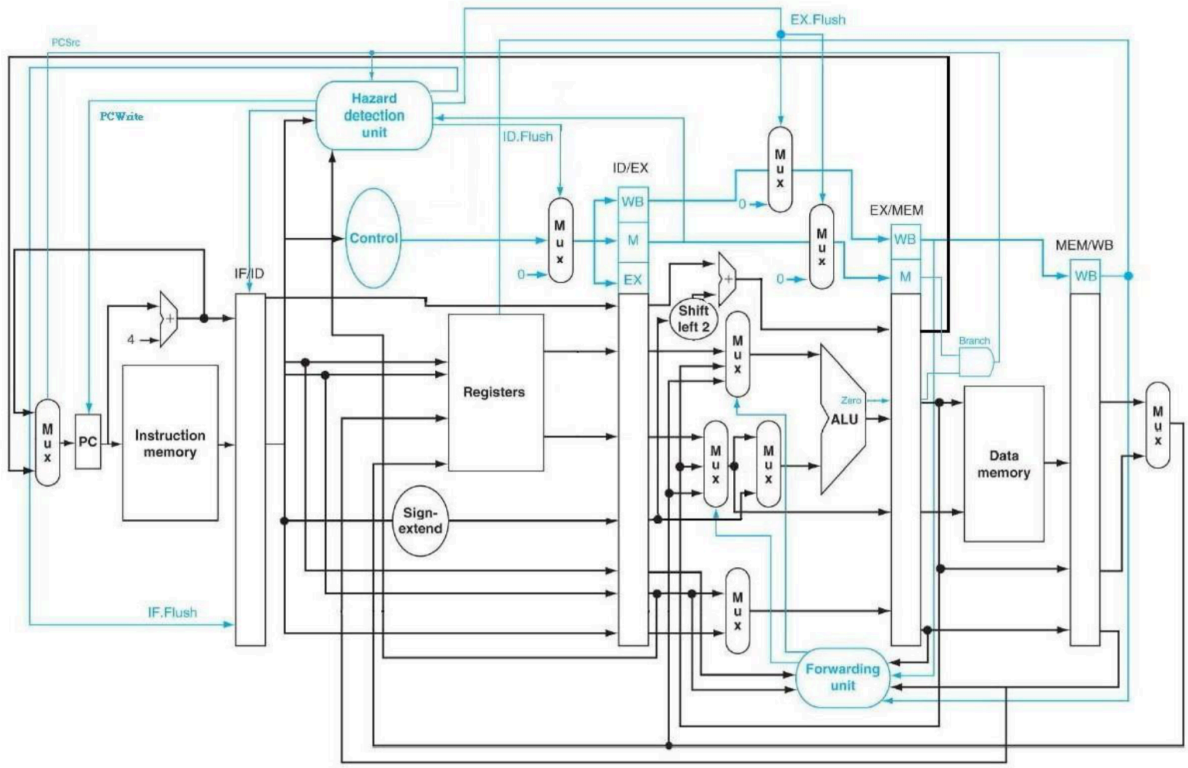
1. Source code and the note

基本上我覺得這次跟上次lab比起來就是加上forwarding和hazard detection 也就是讓pipe register flush和插入bubble[讓PC不要讀下個指令]。

```
if(RegWrite_exmem == 1'b1 && rd_exmem != 5'b00000 && rs_idex == rd_exmem)
    forwardA <= 2'b01;
else if(RegWrite_memwb == 1'b1 && rd_memwb != 5'b00000 && rs_idex == rd_memwb)
    forwardA <= 2'b10;
else
    forwardA <= 2'b00;
```

大概說明一下forwarding判斷的部分，這部分跟講義差不多，基本上就是前個或前前個指令的rd剛好是目前的rs或rt就要forward，至於先判斷exmem再看memwb是為了防止double-data-hazard的情形。

2. Your architecture



(取自於CO_Lab5.pdf)

3. Hardware module analysis

這次加入了兩個module Hazard_Detect_Unit 和 Forwarding_Unit ， 前者是負責load-use時要insert一個bubble，branch時要flush掉後面的指令，而後者如同前面的說明，是用來判斷前一個或前前個指令的rd是否為目前這個指令的 rs 或 rt。

剩餘的部分就是在pipeline_cpu上根據上圖把它接起來。

4.Finished part

basic和advanced instruction set都有做完，基本上advance就是判斷branch然後把前面的幾個pipeline register清掉就完成了。

5. Problems you met and solutions

這次主要是觀念的部分，一開始有點搞錯要flush的pipeline register所以回去看了一下先前的講義，才弄清楚load use的部分。

6. Summary

這應該是這學期最後一次寫verilog了，感覺CPU完成的差不多了，不過這次作業做得蠻順利的，快速解完編譯器出來的error就成功了，不然接成pipeline之後要debug真的很傷腦筋，而且我們才分5個stage而已，如果stage一多起來，用$display來debug感覺真的太難了。