**Computer Organization**

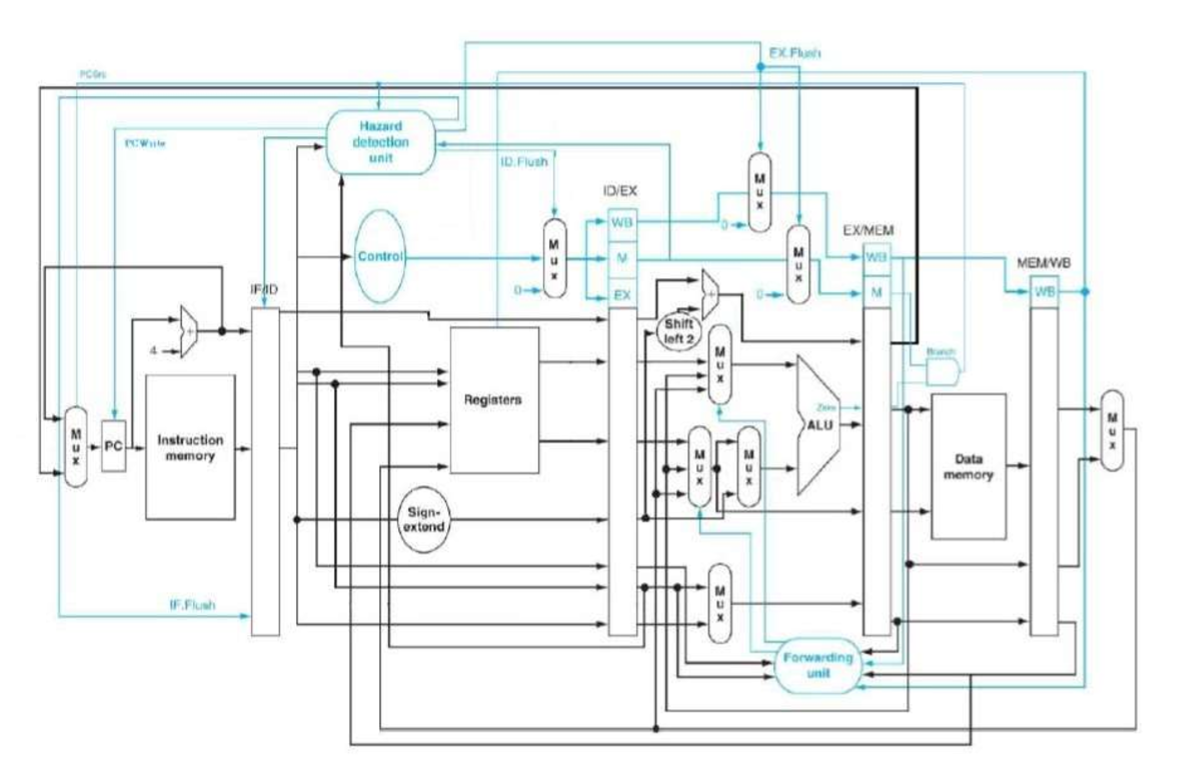
Student ID: Name:

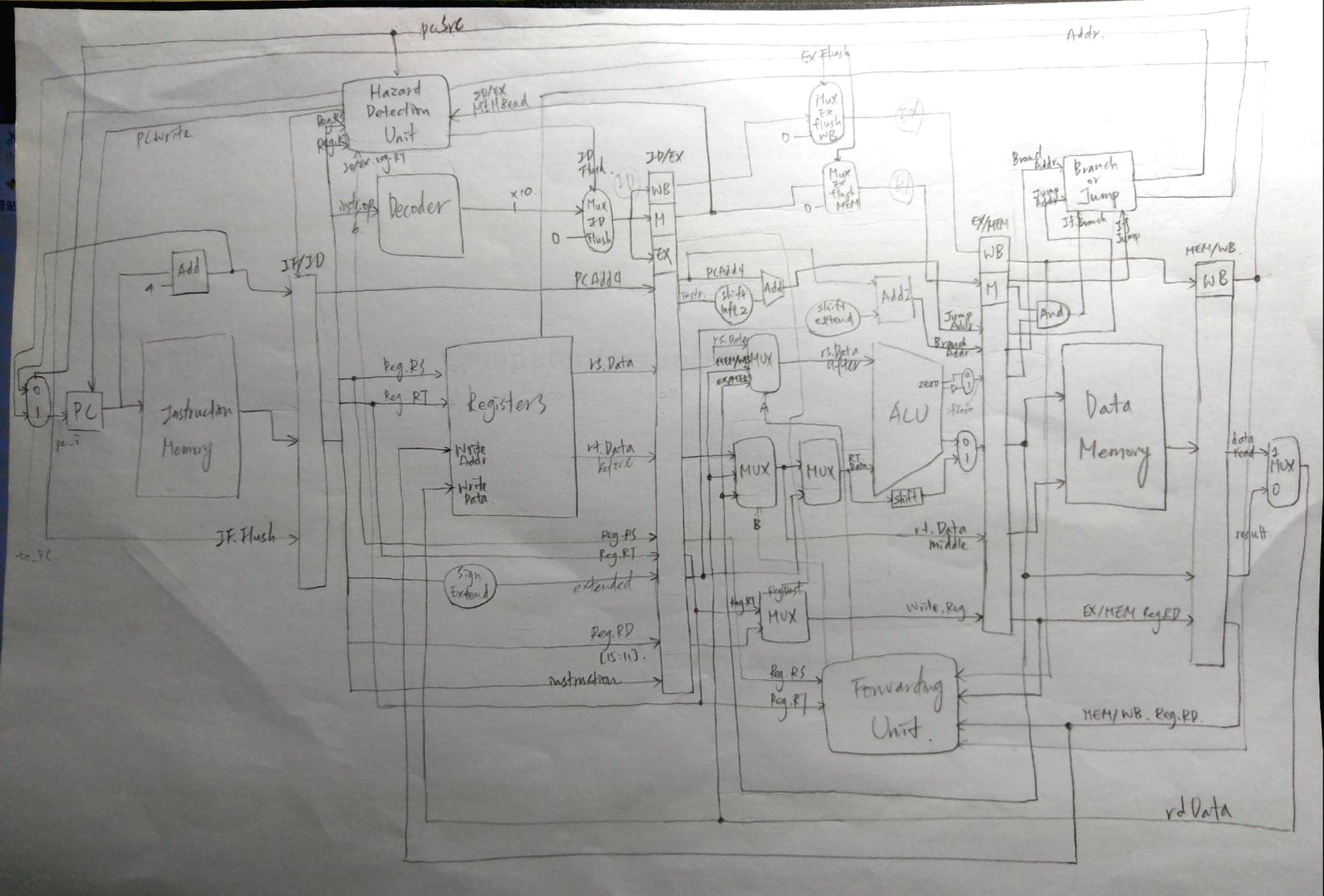
Finished part:

IF/ID, ID/EX, EX/MEM, MEM/WB, Hazard Detection Unit, Forwarding Unit

Architecture diagrams:

(Please write down or plot on the architecture diagram to show what did you do to improve in this lab.)





Hardware module analysis:

Pipeline CPU with data and control hazard detection.

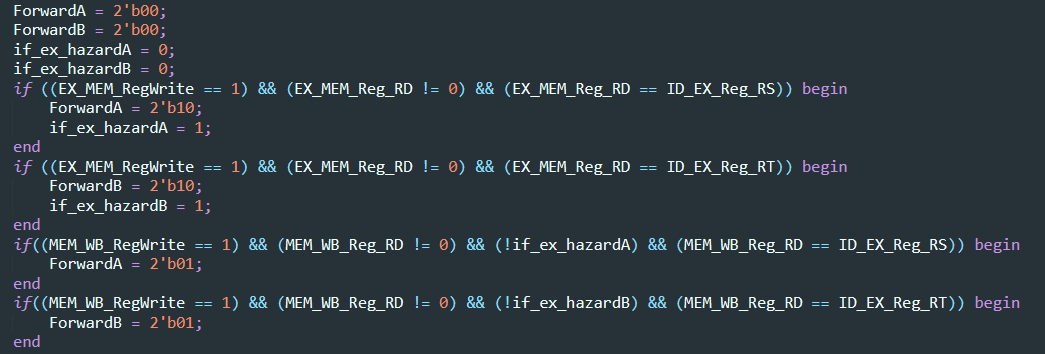
Each pipeline register is “positive-edge triggered” and has default value 0.

Explain how your **Forwarding.v** and **Hazard\_detection\_unit.v** to detect and set the signals to handle the hazard problems

(both test patterns CO\_P5\_test\_data1 and CO\_P5\_test\_data2 are needed)

Ex. The table is just an example format to answer the questions, and there is maybe more or less hazard parts.

* Load-use hazard: ID flush.
* Branch, jump hazard: IF flush, ID flush, EX flush.
* Data hazard:



|  |  |  |
| --- | --- | --- |
| Data/Control Hazard part | Forwarding | Hazard detection |
| 1 lw | no | ID flush |
| 2 branch | no | ID flush  IF flush  EX flush |

Problems you met and solutions:

Jump如何接線路想了很久，後來參考網路上的方法新增一個叫Branch Or Jump的module判斷jump或branch，再接到PC。

Bug很難找，第一次是遇到pipeline register 傳過來的資料有誤，導致寫回register時，write address與 write data不同步。後來找到是在ID/EX的register接錯instruction，應該接IF/ID送過來的instruction，卻接成instruction memory送出的instruction。

之後還遇到branch失敗的問題。也是因為一個地方不小心寫錯，將branchType寫成branch。

後來發現一個不錯的debug的方法：根據錯誤去回溯所有會觸及到的線路與module，再一個一個印出來看，如此逐漸縮小範圍，會比較容易發現bug所在。

Summary:

這次作業的線路跟模組很多，要先畫好完整的線路圖，有條理地增加module後，再接上所有電路。名稱的格式必須有一定的規則，否則容易錯亂。

做完這次作業，對於pipeline如何處理hazard更清楚了。雖然debug很累，但是很值得，獲益良多！