Computer Organization

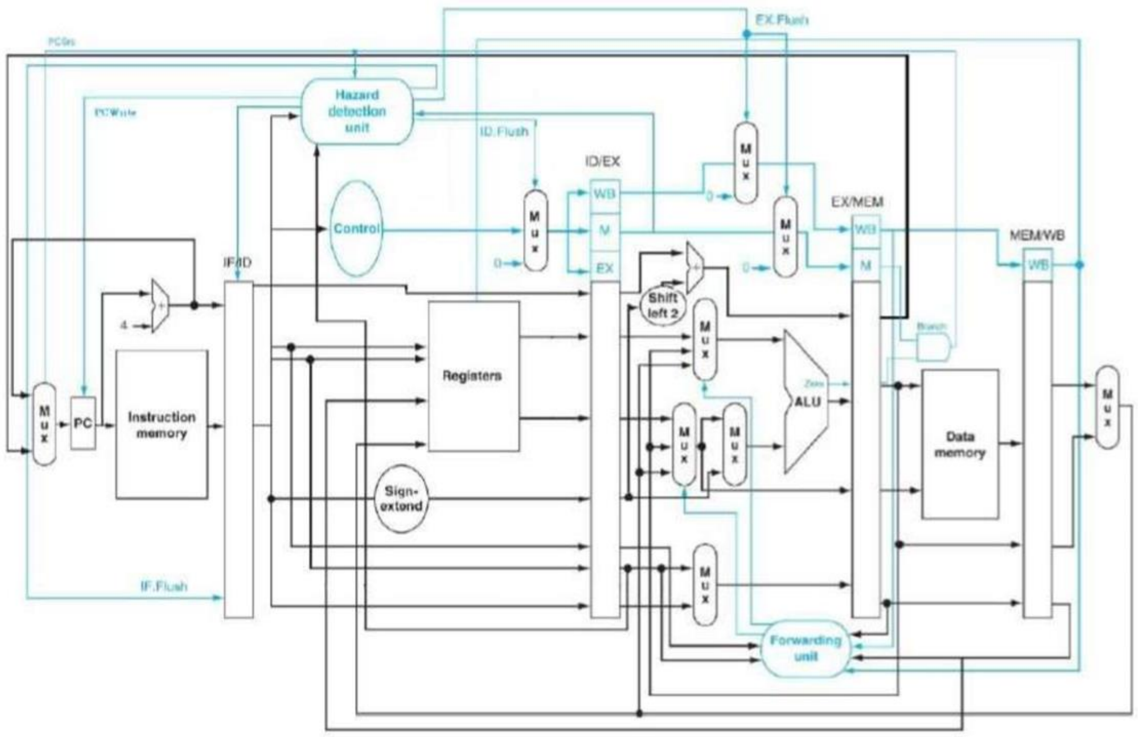
Student ID: Name:

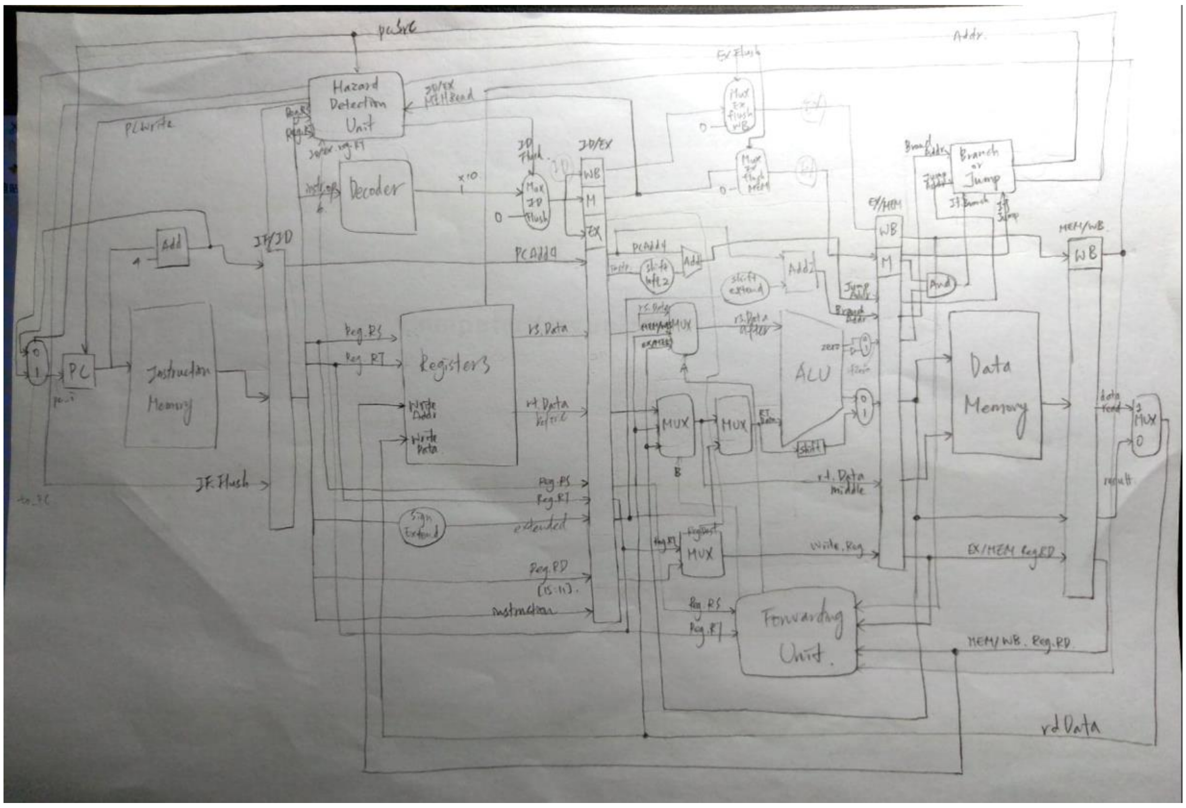
Finished part:

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.)



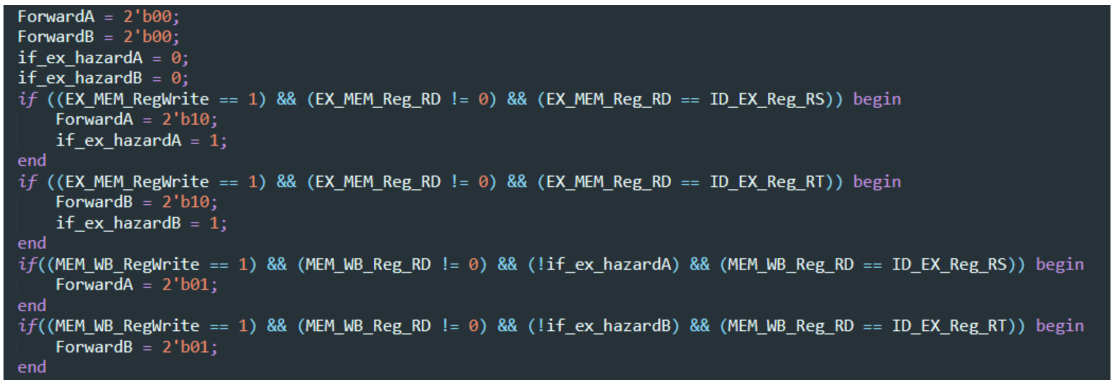


Hardware module analysis:

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:



```
ForwardA = 2'b00;
ForwardB = 2'b00;
if_ex_hazardA = 0;
if_ex_hazardB = 0;
if ((EX_MEM_RegWrite == 1) && (EX_MEM_Reg_RD != 0) && (EX_MEM_Reg_RD == ID_EX_Reg_RS)) begin
    ForwardA = 2'b10;
    if_ex_hazardA = 1;
end
if ((EX_MEM_RegWrite == 1) && (EX_MEM_Reg_RD != 0) && (EX_MEM_Reg_RD == ID_EX_Reg_RT)) begin
    ForwardB = 2'b10;
    if_ex_hazardB = 1;
end
if((MEM_WB_RegWrite == 1) && (MEM_WB_Reg_RD != 0) && (!if_ex_hazardA) && (MEM_WB_Reg_RD == ID_EX_Reg_RS)) begin
    ForwardA = 2'b01;
end
if((MEM_WB_RegWrite == 1) && (MEM_WB_Reg_RD != 0) && (!if_ex_hazardB) && (MEM_WB_Reg_RD == ID_EX_Reg_RT)) begin
    ForwardB = 2'b01;
end
```

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！