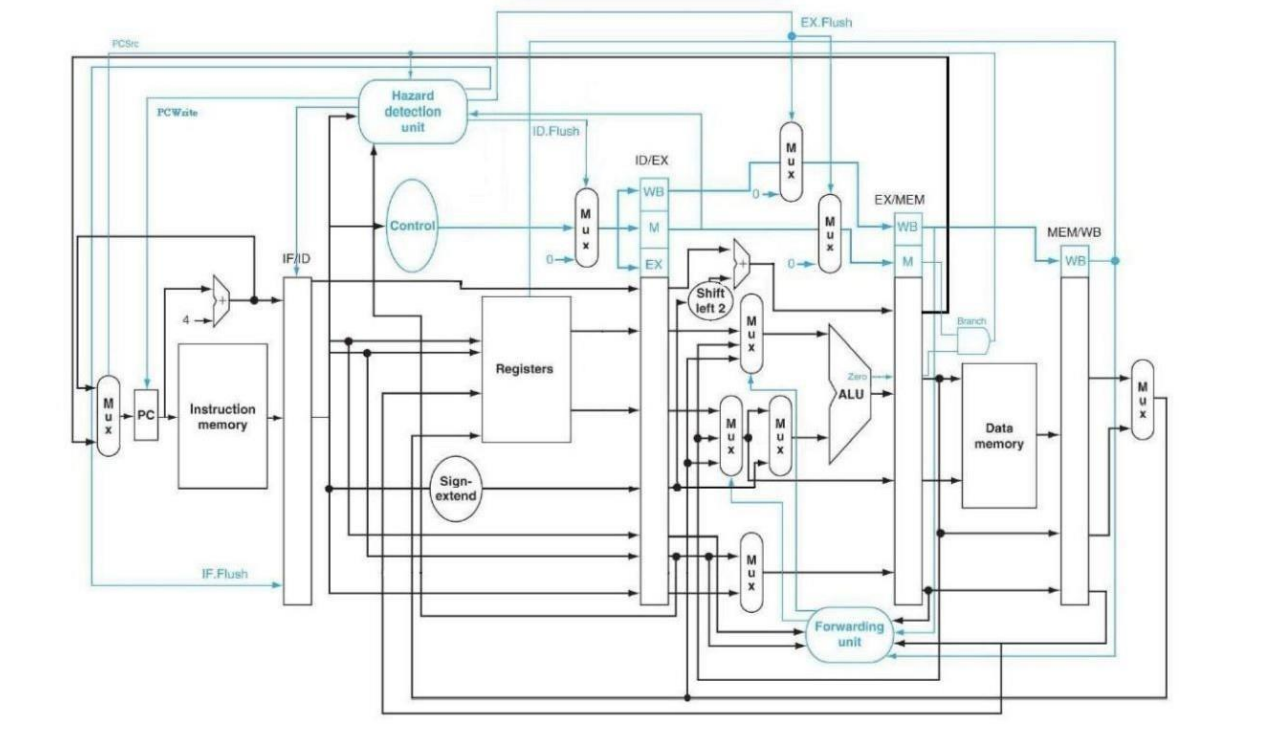
**Computer Organization Lab5**

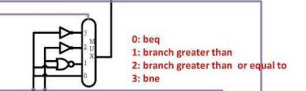
**Name:尤茂為**

**ID:110550065**

**Architecture diagrams:**

****

加上LAB3有提供但沒用到的部分，將其放在branch的其中一個輸入



**Hardware module analysis:**

**(explain how the design work and its pros and cons)**

跟上一次的lab相比，這次我多使用了MUX3to1跟MUX4to1來實作，在某些細節上需要更改，例如要從EX傳到MEM的ReadData2，原先是直接將ID得到的直接傳過去，但這邊需要用的是經過MUX3to1處理後的，也就是還要考慮是否有做forwarding。

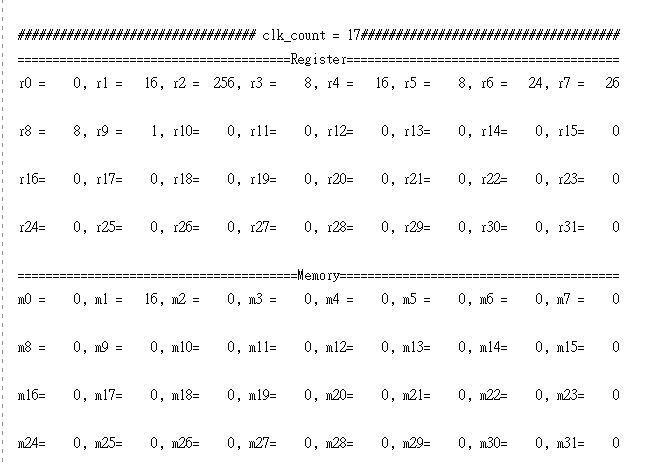
這次的lab多了hazard\_detection與forwarding的功能，能使CPU直接處理有關load-use的問題並利用forwarding的功能，減少需要stall的情況，因此，需要考慮的事情也變得比較多，有關flush與PC的更新與否等，參考了講義第四章裡給的條件判斷式子來產生。

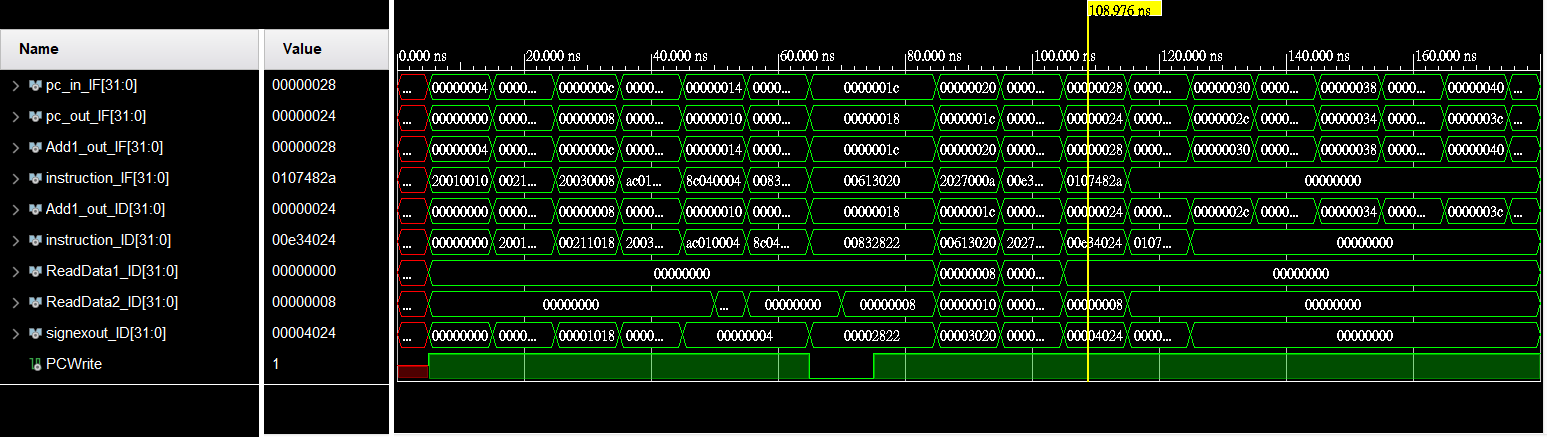
此外，我也更新了ALU、ALUCtrl與Decoder裡的內容，將一些不需要的instruction刪掉，避免發生不必要的錯誤。

**Finished part:**

**(show the screenshot of the simulation result and waveform, and explain it)**

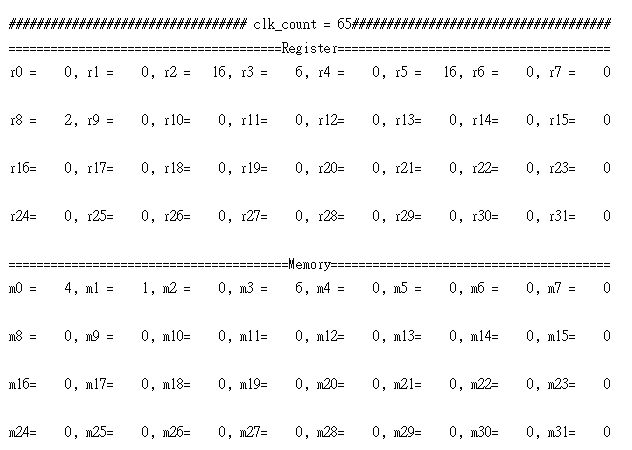
Part1:

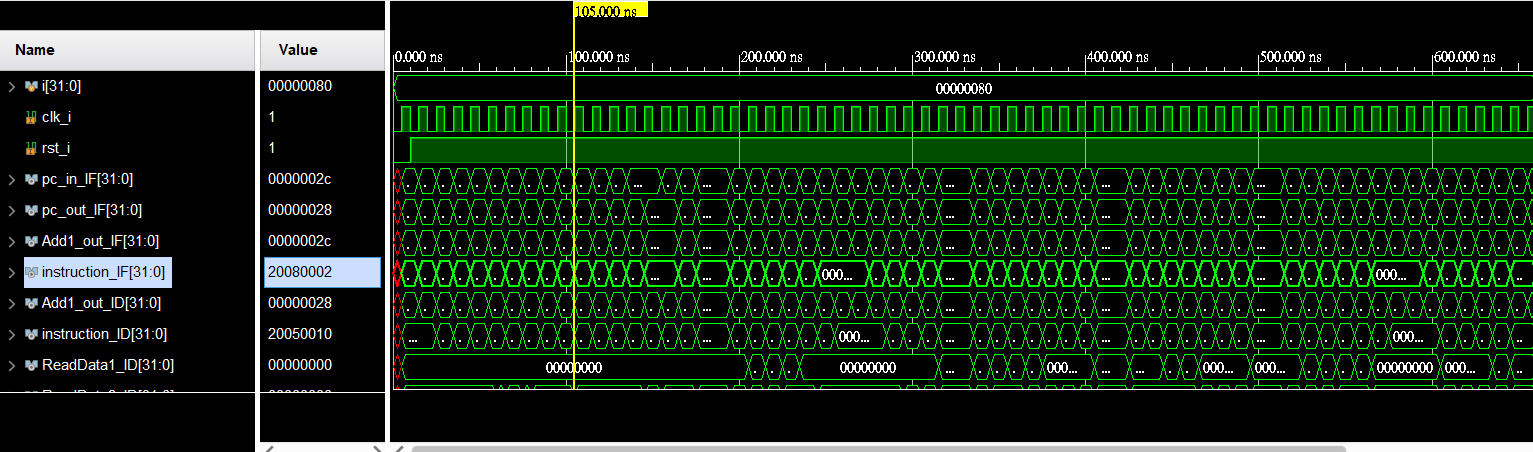




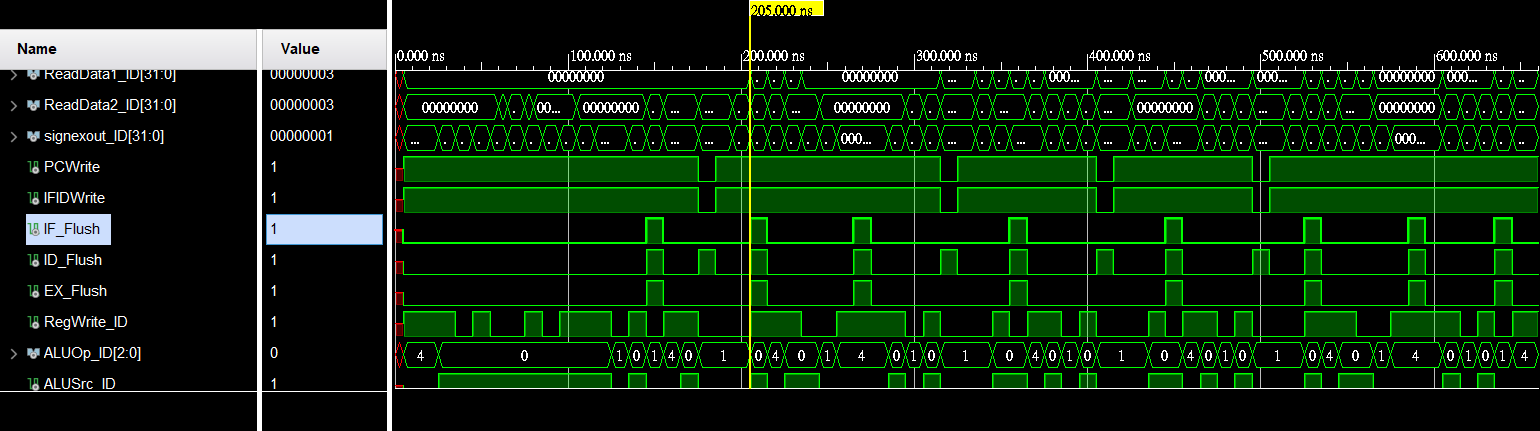
在這個時間點，IF stage正操作著I10: slt $9 $8 $7的部分，而ID stage正處理I9:$8 $7 $3，每個不同的stage分別處理不同的instruction，符合pipeline的特性。

Part2:





第二部分的測資就複雜許多，多了branch類的指令，但每個stage依舊正作著不同指令的工作。



我們能看到有三個Flush的產生，這有可能是因為前面的instruction被偵測到有branch的發生導致接下來要將Ripe\_Reg的內容清空。

若是單純load-use的hazard，只需將ID/EX的Pipe\_Reg內容清空即可。

**Problems you met and solutions:**

這次的lab基本上沒有容錯的空間，線一不小心沒接好就可能導致整個結果都是錯的，而在不確定lab4所繳交的作業是否正確的情況下，無法確切知道是哪部分或哪個功能出錯，需要根據波形圖來查看哪部分錯誤，但線的數量非常的多，常常會看到眼花撩亂，最後索性直接將Pipe\_CPU的線重新接過，幸好最後的結果是好的。

**Summary:**

這次lab的時間點恰好在期末周，在時間嚴重不足的情況下，還要處理那麼多變數並與加上hazard\_detection與forwarding這兩個功能，常常會因為急著想完成而粗心出錯，最後還是花了不少時間才完成此次lab。