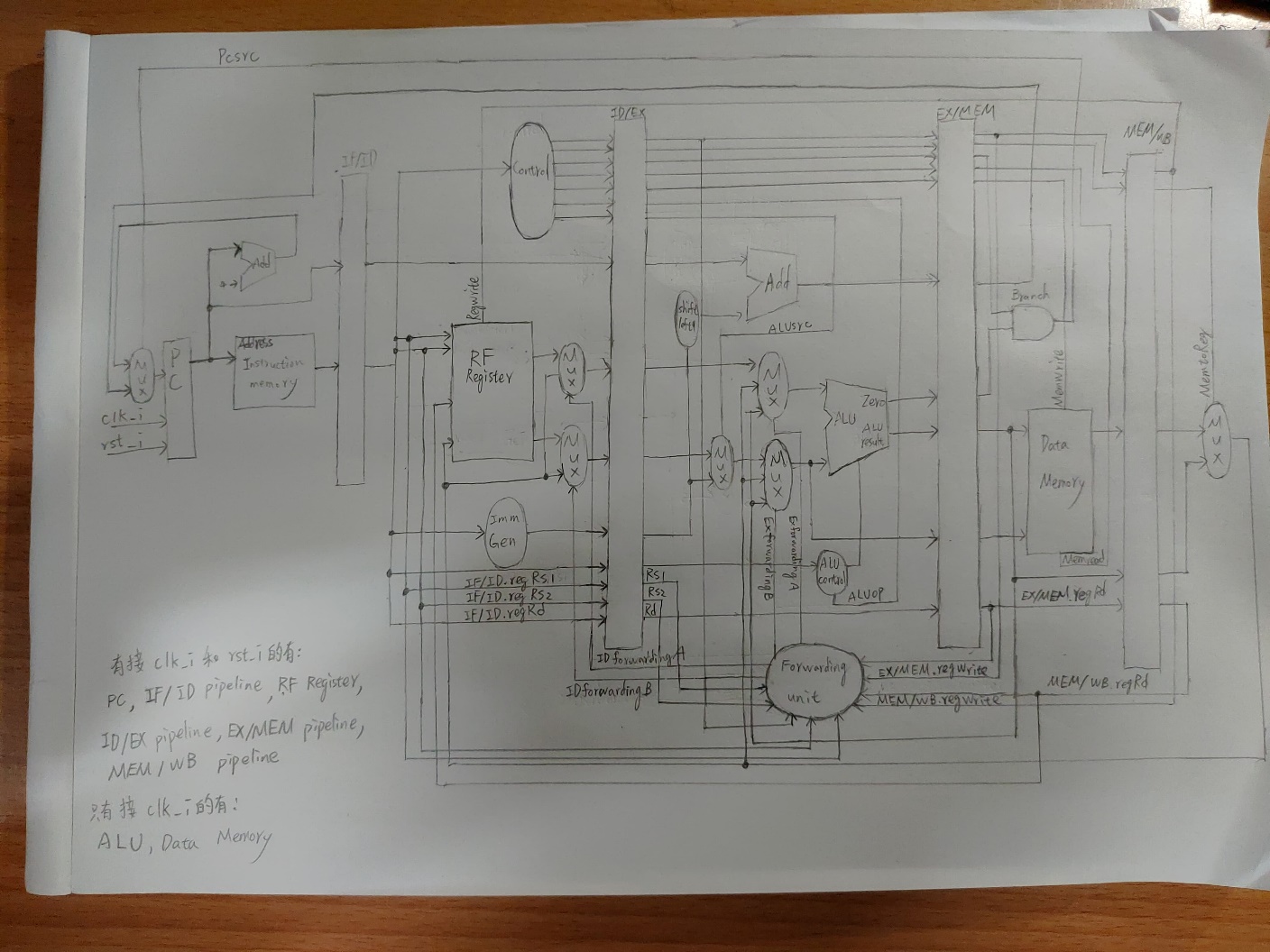
**Computer Organization**

**Architecture diagram:**

****

**Detailed description of the implementation:**

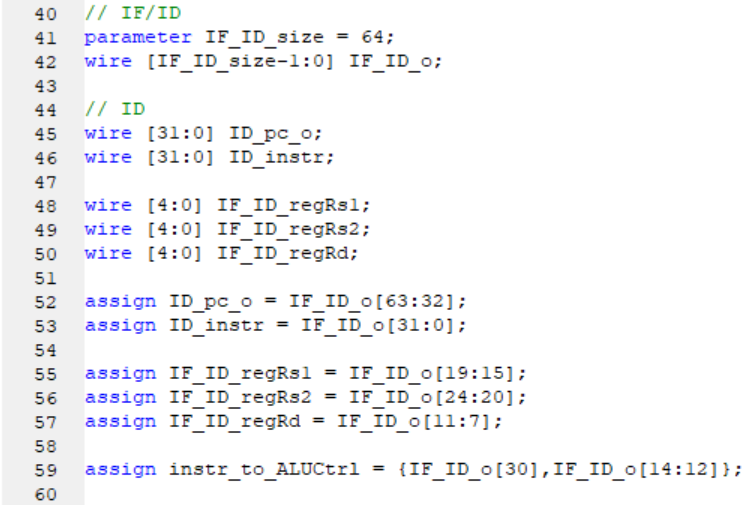
以下將著重在這次lab的重點，分為兩part，pipeline & forwarding unit

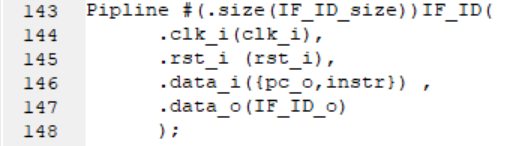
1. Pipeline:

根據課本的架構分為5 stages，分別為IF、ID、EX、MEM、WB

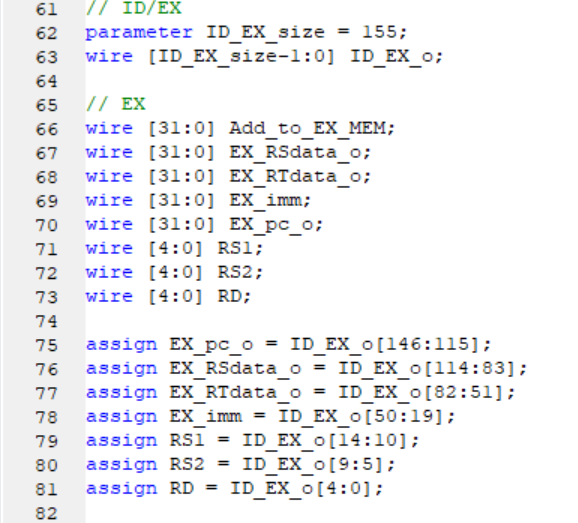
以此分為4個pipeline register : IF/ID、ID/EX、EX/MEM、MEM/WB

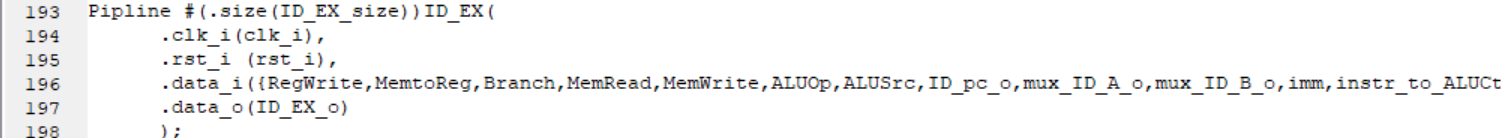
IF/ID: (IF\_ID\_size = 64)



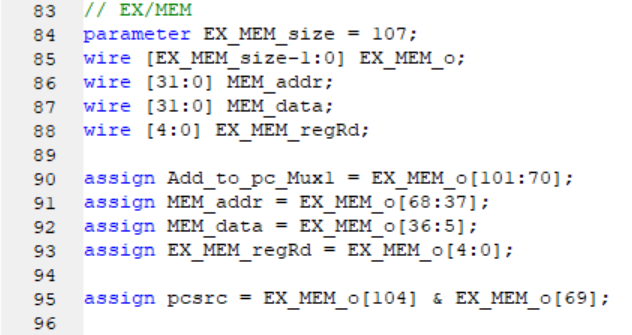


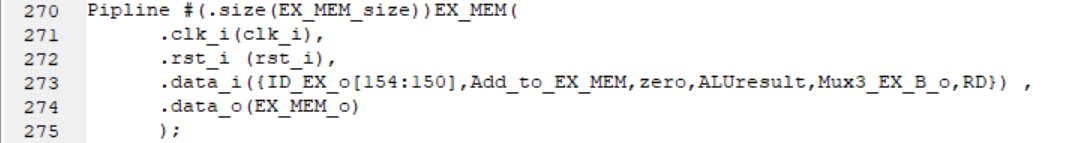
ID/EX: (ID\_EX\_size = 155)



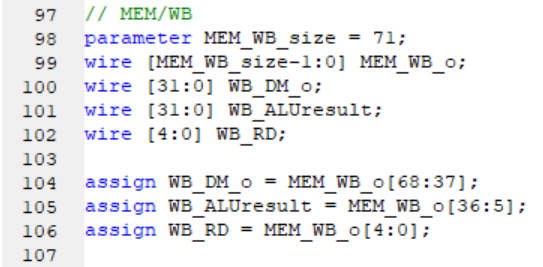


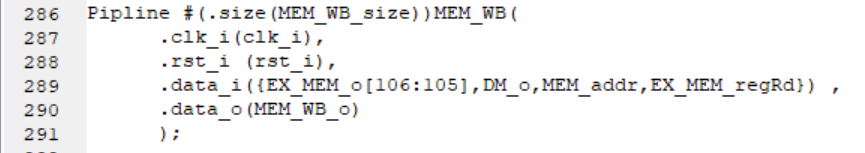
EX/MEM: (EX\_MEM\_size = 107)



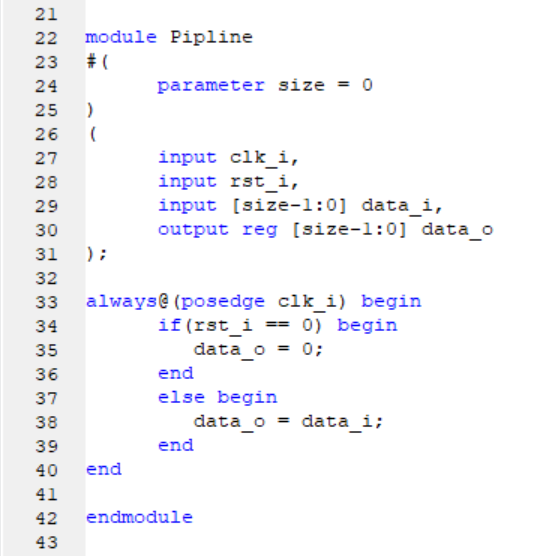


MEM/WB: (IF\_ID\_size = 71)



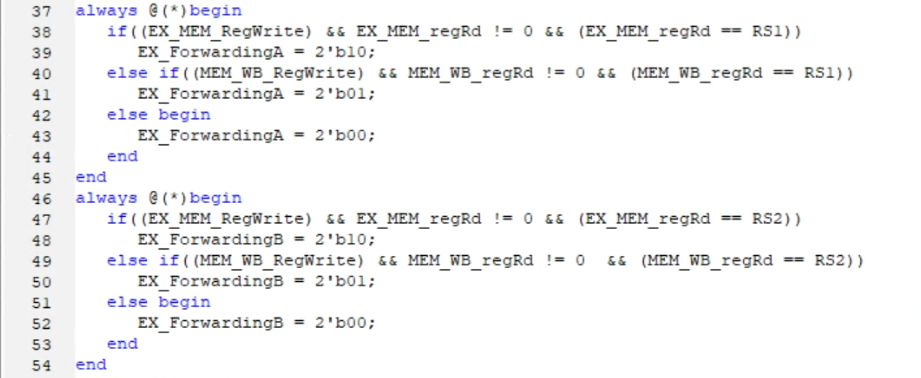


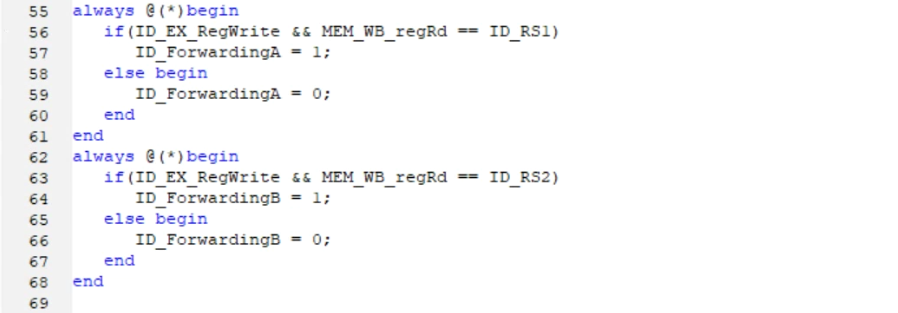
Pipline.v簡介:



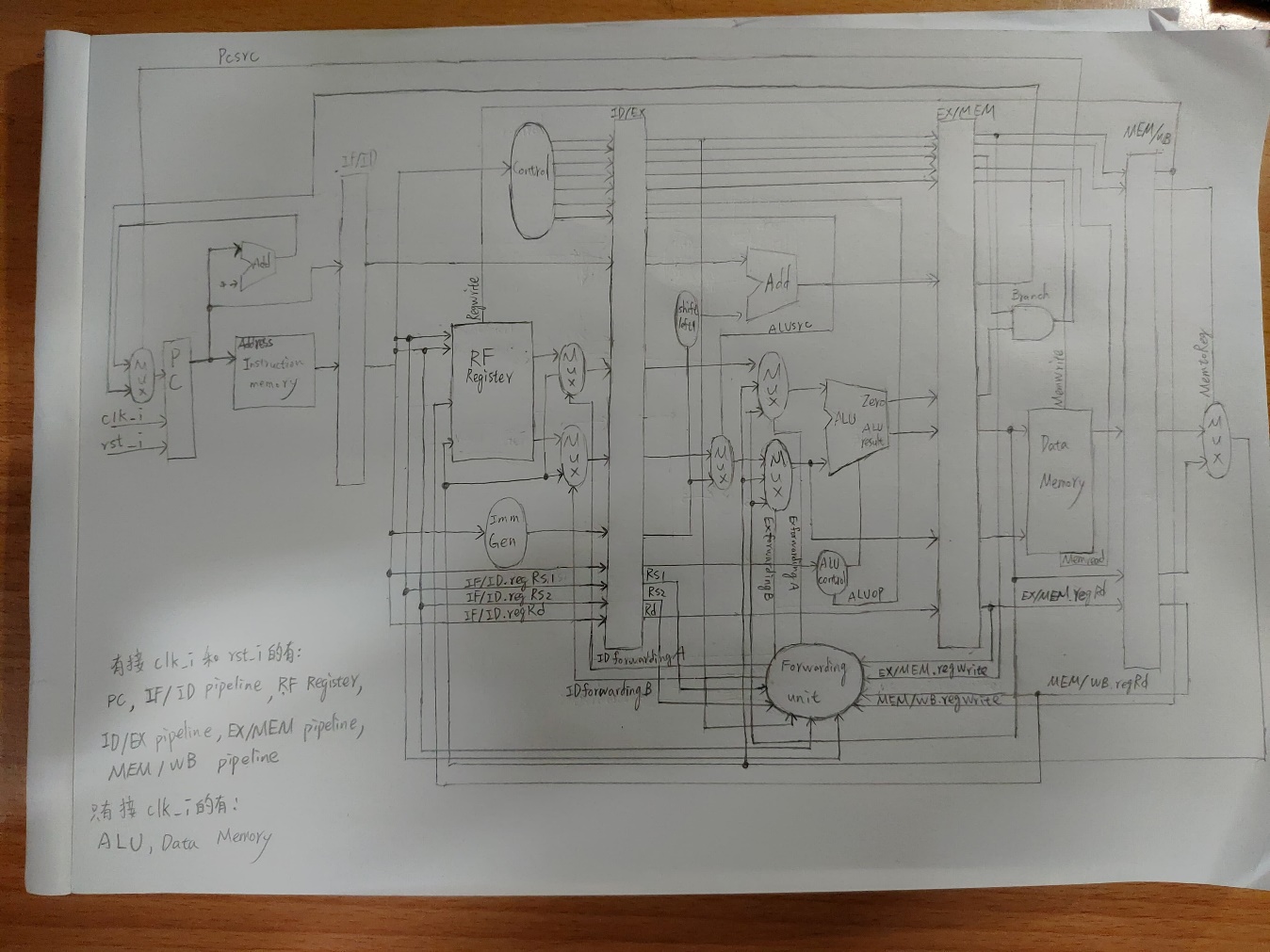
我們使用#()語法，使我們可以方便更改各個所需pipeline register的大小(parameter size)。該pipeline reg根據clk訊號，在每個positive clock edge triggered，以此把data寫入。

1. forwarding unit



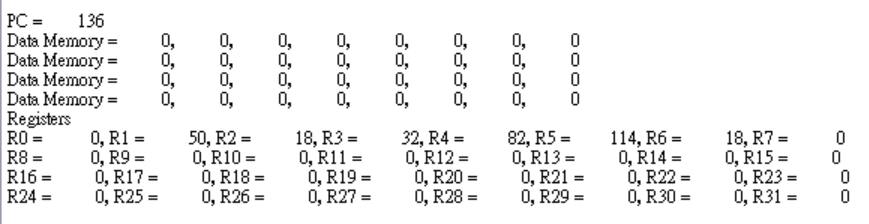


我們使用always blocks以區別4個forwarding的判斷。

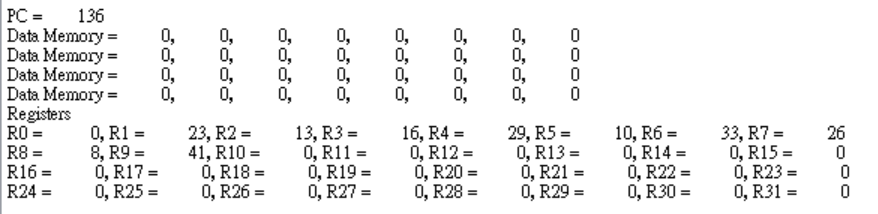
****

**Implementation results:**

CO\_test\_data1.txt

****

CO\_test\_data2.txt

****

**Problems encountered and solutions:**

cpu到了pipeline & forwarding unit的階段，過程中遇到的問題有 :

1. Pipeline register的部分，每個stage所需的大小並不相同，查詢後，知道有#()的語法得以使我們可以在top module的地方跟改size大小。
2. 比較困難的在forwarding unit的部分，要去思考if-else條件的判斷式要如何寫訊號才能做到正確的forward。最後版本，我們用了4個always block分開4個做forward的mux select訊號，不知道有沒有更簡潔的解法。

**Comment:**

在一個個verilog projects中，從當初光是要讀懂範例code就要花上不少力氣，到現在也算是慢慢掌握了一點點寫verilog的技巧，相對的寫作業的時間也越來越好掌握，bug會出現在哪裡也大致上越來越可以猜的到了。