**Computer Organization 2017**

**HOMEWORK III**

系級: 資訊108 學號: F74046242 姓名: 謝耀賢

**實驗結果圖(snapshot of the results)**

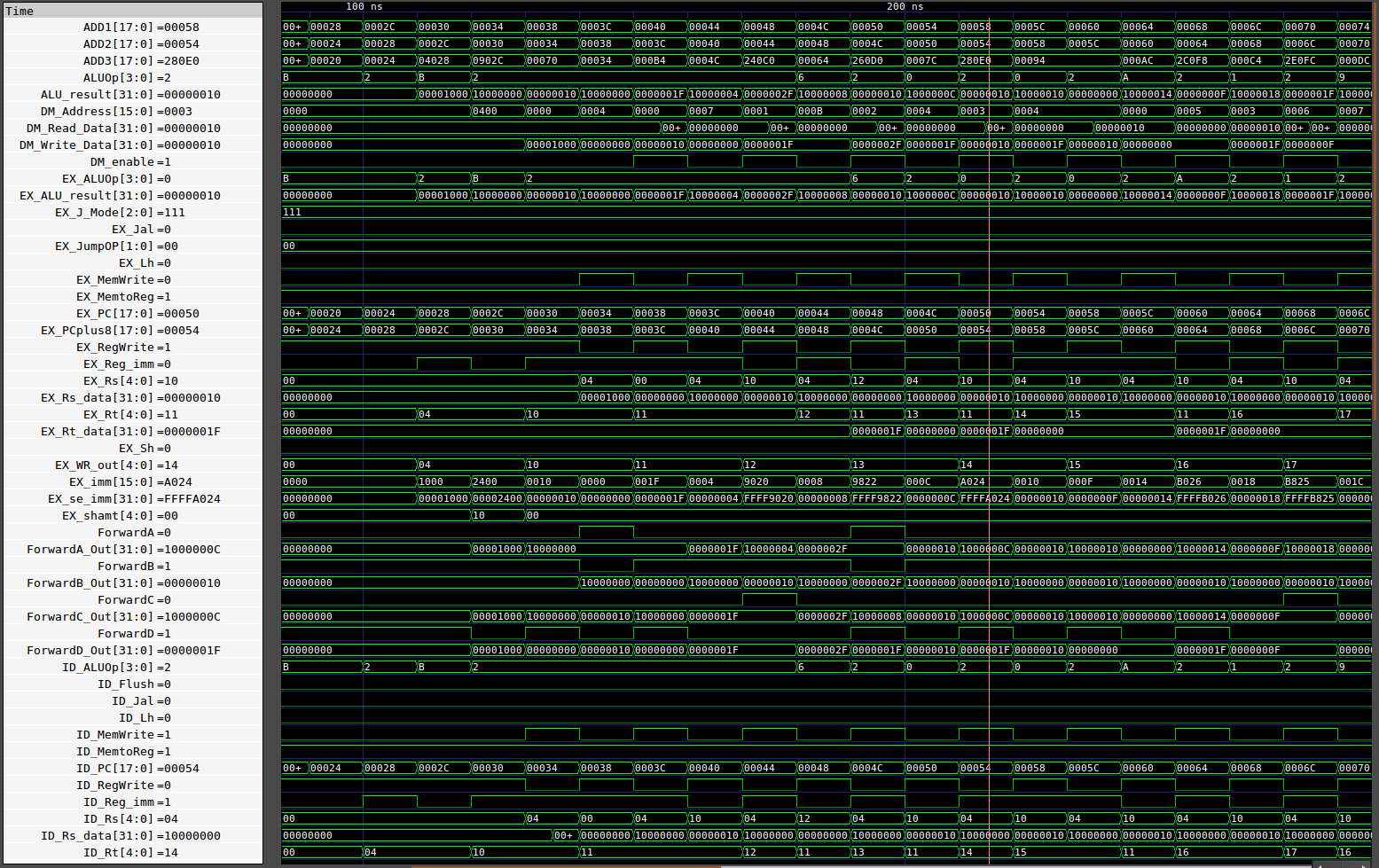
****

**指令波型圖(Snapshot of the instruction execution waveforms.)**

(Please explain why your snapshot is correct, including the wires, signals. The description should be as detailed as possible, e.g. why this situation occurs, and in waveform where does it occur?)

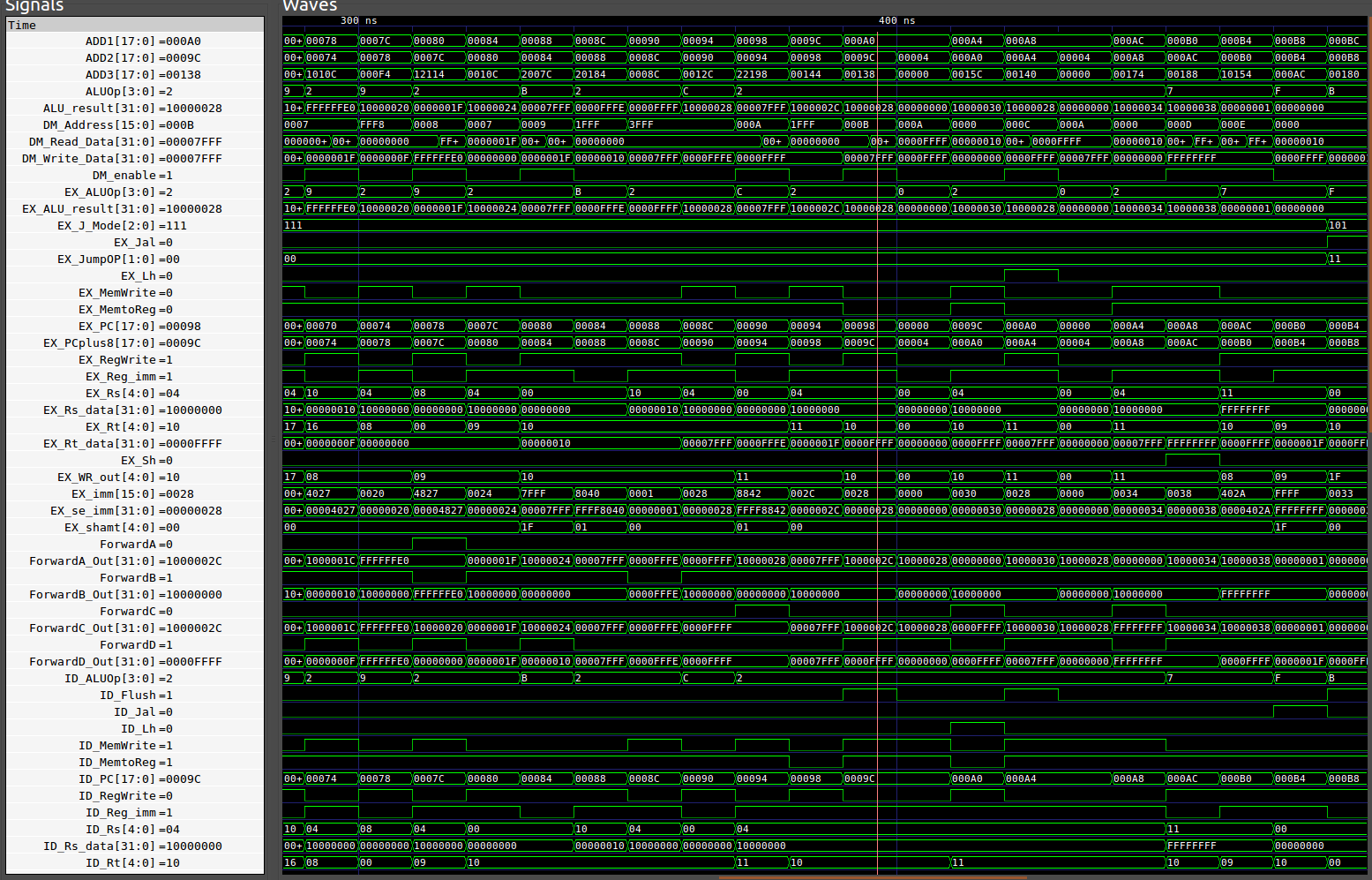
1. Instruction with Forwarding

R-type:

Description:

輸入R-type指令，會依序於5 step: IF, ID, EX, MEM, WB中執行，若有register值讀取時尚未Write back問題之時，會利用FU進行forwarding動作，會檢查此值是到了MEM階段或是WB階段，再將此值取過來直接扔進ALU中進行運算

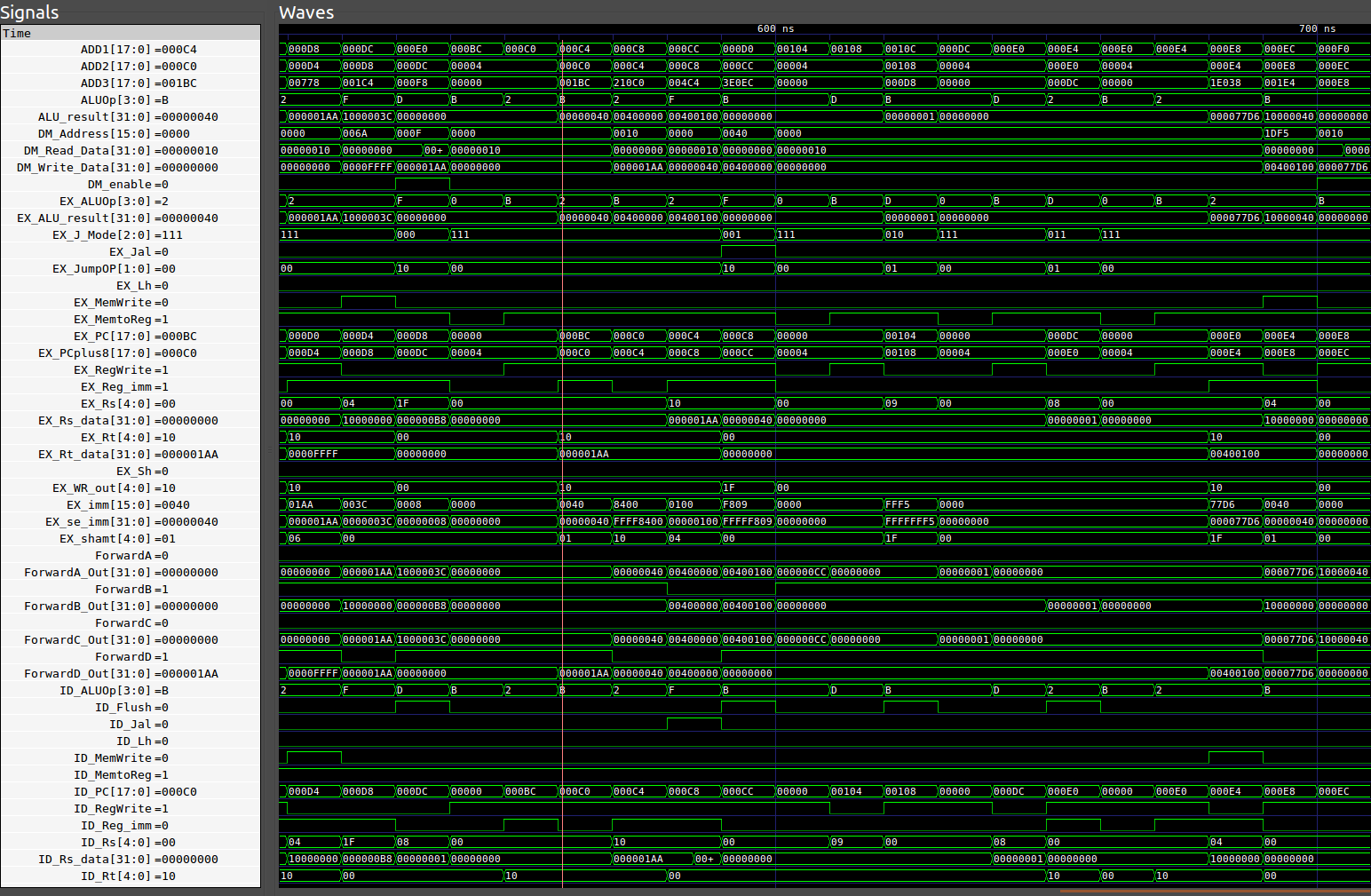
I-type:



Description:

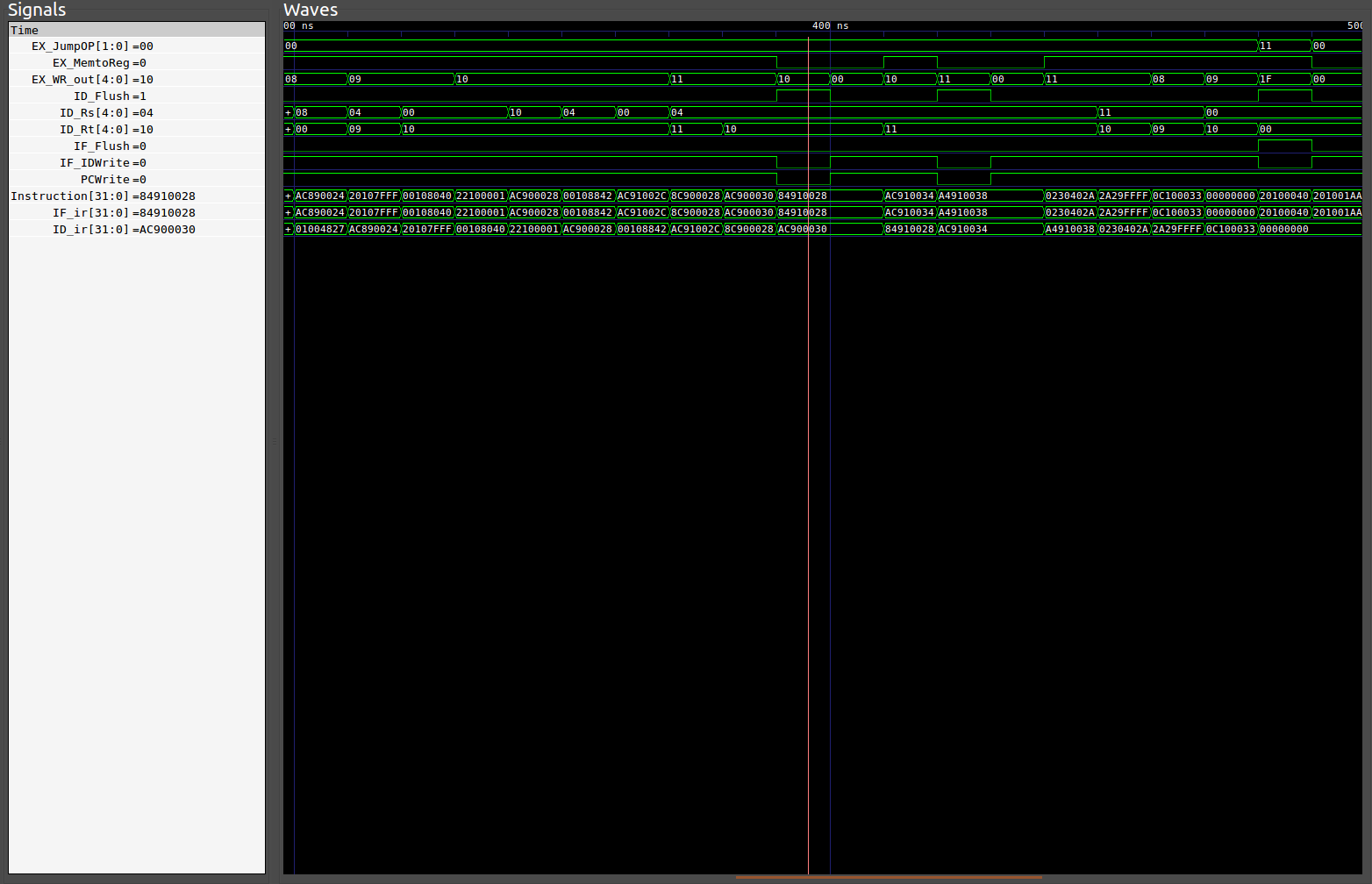
輸入I-type指令，會依序於5 step: IF, ID, EX, MEM, WB中執行，若有register值讀取時尚未Write back問題之時，會利用FU進行forwarding動作，會檢查此值是到了MEM階段或是WB階段，再將此值取過來直接扔進ALU中進行運算，比較需要注意的是lw指令，若lw所存放register與下一指令所用的register相同情況下，利用HDU再這兩指令中塞入nop指令之後再將值利用forwarding傳至EX階段進行運算(詳細於下方講解)

J-type:

Description:

將J-Type指令輸入，當遇到有branch的時候，則會依照branch delay的方式進行處理

1. Load Stall:

Description:

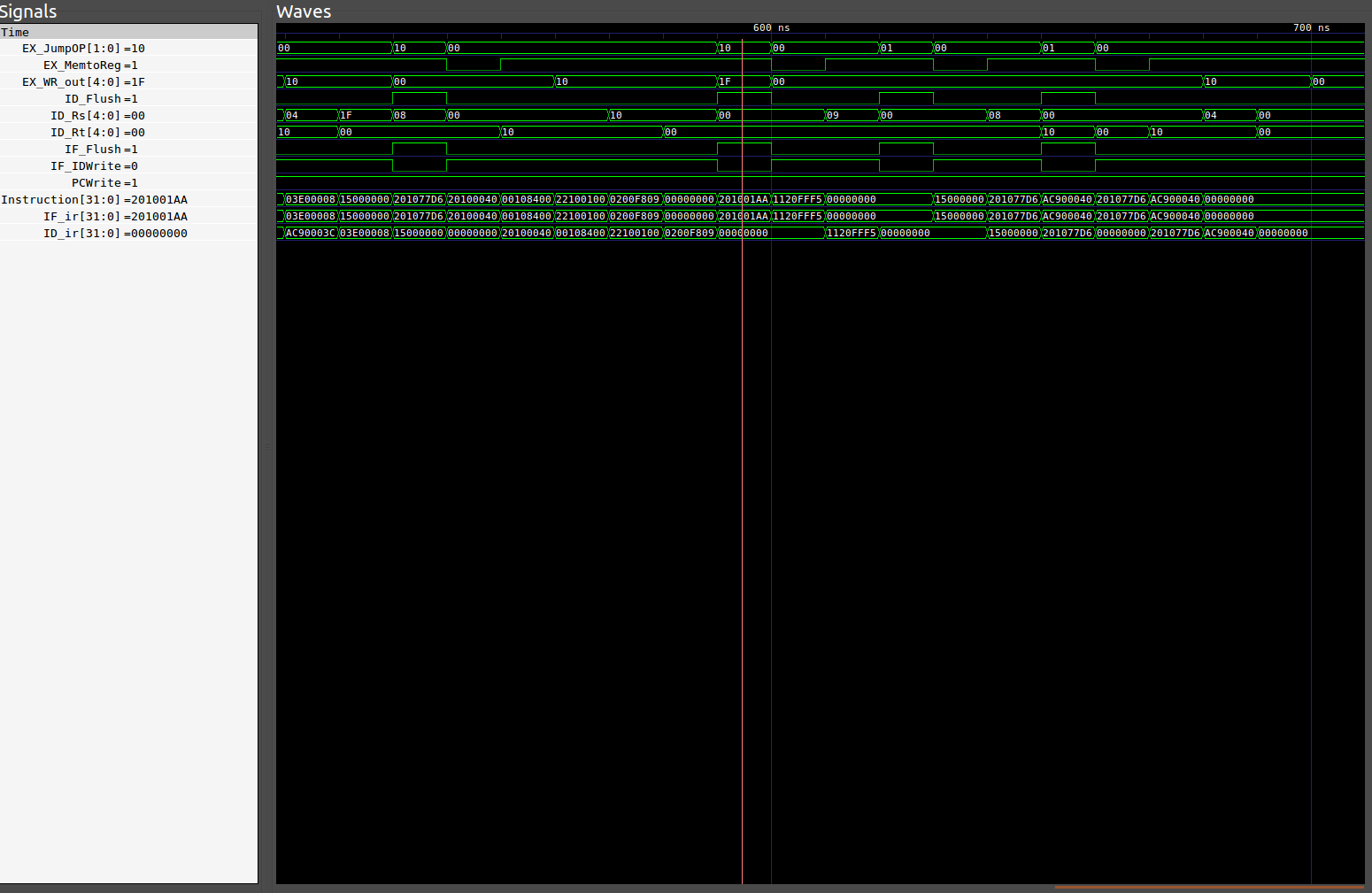
當輸入lw指令時，若lw存放之register位置與下一指令所讀取register位置相同之時，會再兩指令中插入一nop指令(因為lw在cycle4中執行，而下一指令卻在cycle3開始即需要資料運算，即便forwarding仍來不及)

插入nop指令方式如下:

1. 把IF/ID設為0(flush掉→ID\_Flush = 1)

2. 把PC的writeEnable設為0(不允許下一指令讀入)

1. Branch Delay (& Flush):



Description:

在cycle 2判斷是否要branch，若是需要branch，則將前一個instruction 給flush掉(錯誤的)

flush:

PCWrite = 1

IF/ID Write = 0

IF Flush = 1

ID Flush = 1

If you CPU data is from Figure 5 in the home, show your CPU datapath and explain why you want to design your CPU this way.

Reason: The same.

**心得(Report)**

這次的作業除了多了HDU、FU要寫之外，因為pipeline 將原本的1個cycle內要做的事情分成5個step執行，所以亦多了4個切換step的module要寫，所以在接線的時候(top.v)，多了一大堆的線要接，在DEBUG的時候，常常會有接錯線的問題。而且在弄懂HDU、FU上也花了很多時間。在這之中花最多的時間在於，不知道當同時出現WB以及MEM 需要forwarding的處理方式，而導致在DEBUG上花了很多時間，總而言之，在這份作業中，總算終於了解pipeline 如何處理hazard問題。