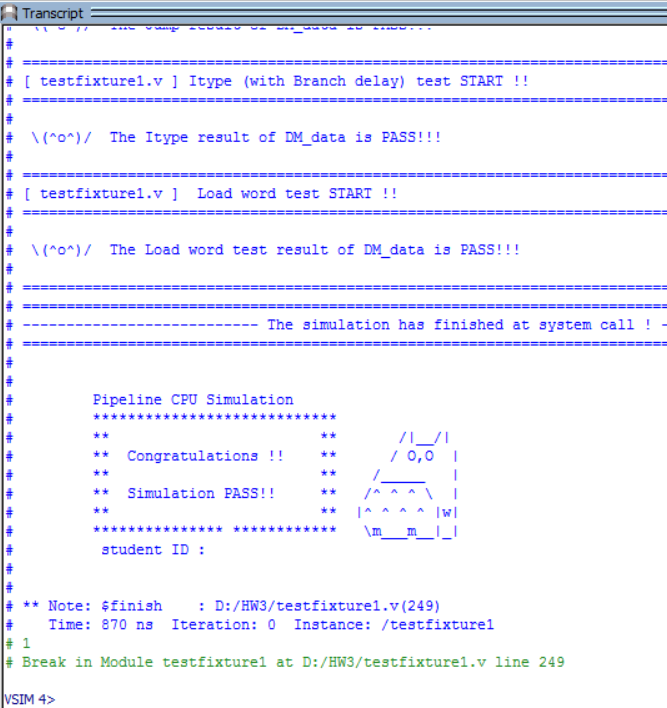
**Computer Organization 2016**

**HOMEWORK III**

系級: 107資訊系 學號: F74031051 姓名: 林允文

**實驗結果圖(snapshot of result)**

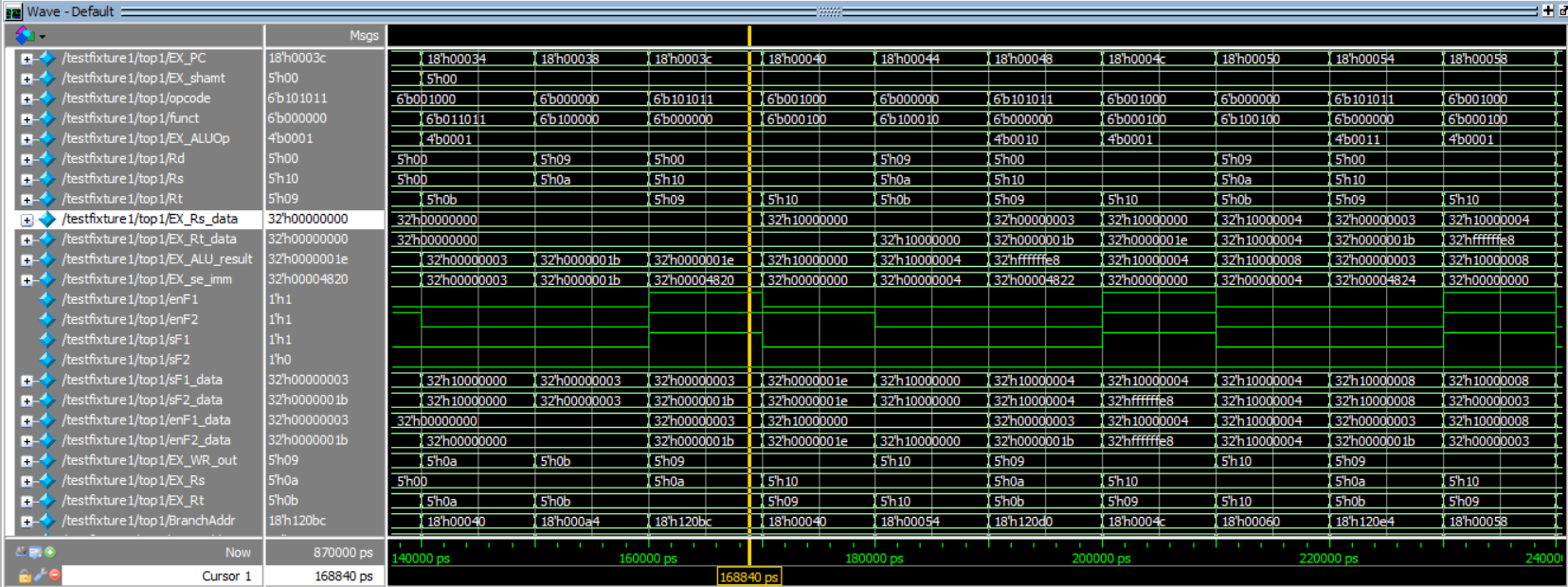


**指令波型圖( Instruction waveform of some situations)**

(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)

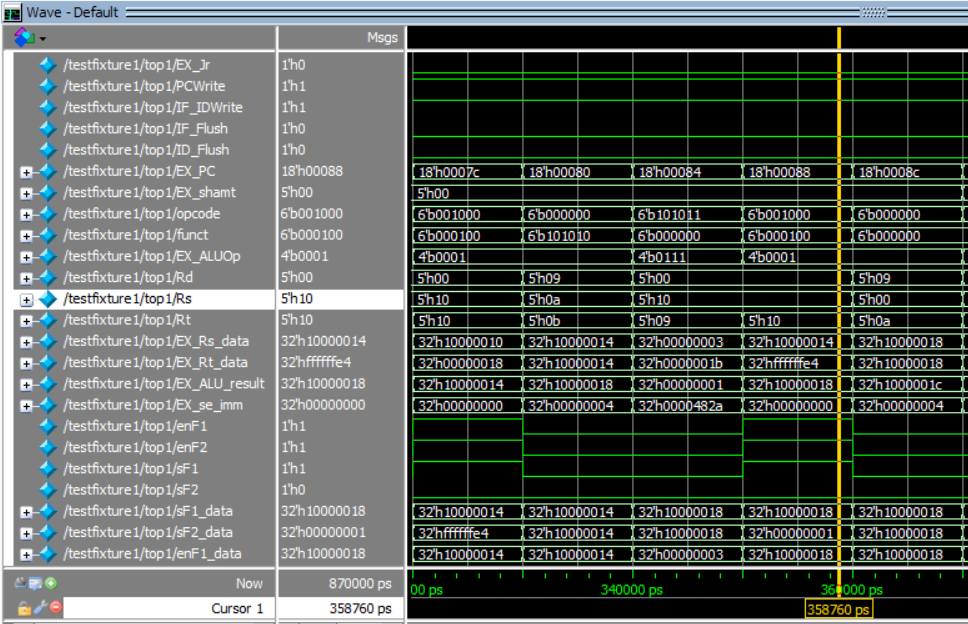
R-type:



Description:

因為opcode和funct是在第二個ID stage取到，所以當進入下一個階段也就是EX stage時就會執行，因此在紅色框框中(opcode:000000 funct:10000)add指令會在黃線的這個階段將Rs和Rt進行相加，但是發生兩者皆forwarding的情況，所以是將enF1 + enF2 (如藍色框框) ( 00000003 + 000001b = 000001e 如黃色框框的EX\_ALU\_result )，故答案是正確。

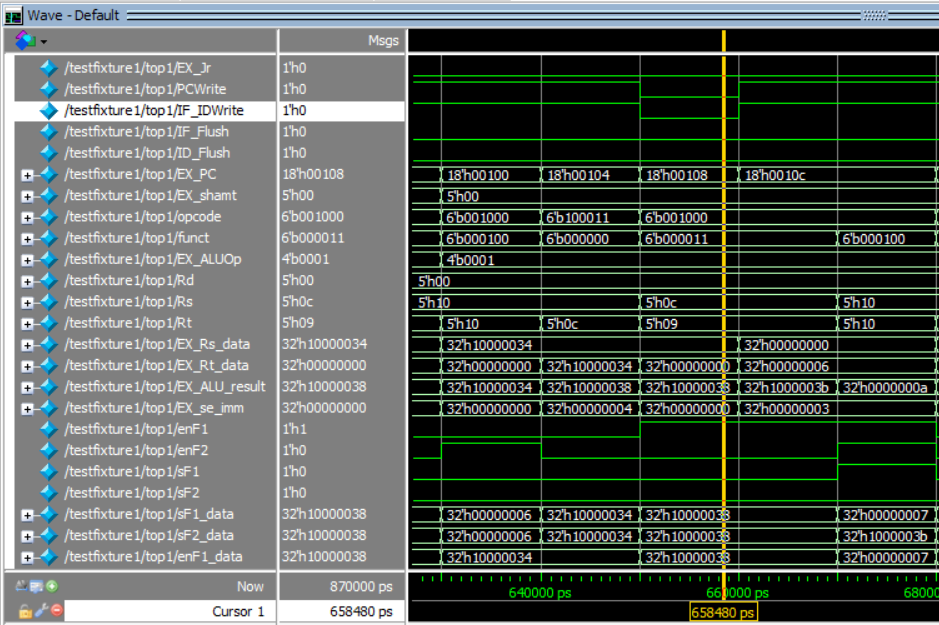
I-type:



Description:

在黃色線段前兩個stage是slt指令(紅色框框)，剛好他的Rd是09(看黃色框框)，正好是下一個指令sw指令(看白色框框)的Rt，所以當sw指令從ID stage進入EX stage時，會發生forwarding(正是黃色線段所在的階段)，可由enF2=1和sF2=0(如藍色框框)，得知Rs的值是從MEM stage所來。

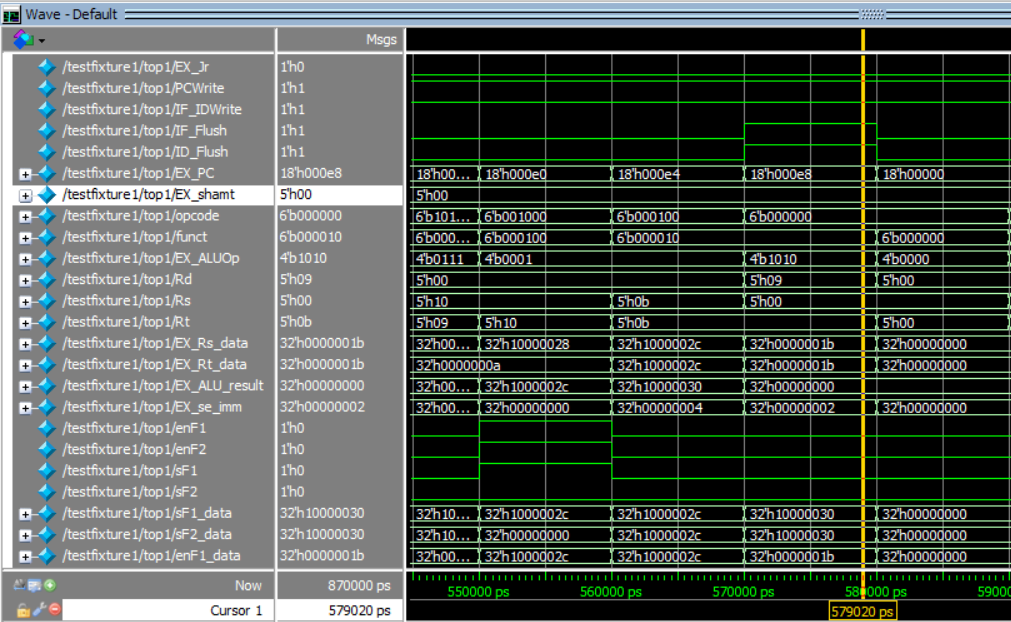
1. Load Stall:



Description:

因為紅色框框中的為load指令，因為他要存取的Rd是0c，正好是下一個指令(黃色線段所在)的Rs，因此在這邊發生stall，可以由藍色框框中的PCWrite和IF\_IDWrite為0可以得知，所以白色框框中的opcode和funct都沒有變。

1. Branch Delay (& Flush):



Description:

在ID stage知道是beq指令(如白色框框內)，所以在下個階段EX stage中(目前黃色線段所在)直接比較Rs和Rt的值，( 0000001b = 0000001b 如紅色框框中)，因此下一個PC值並不是原本的 +4 (如黃色框框)，另外要清除錯誤的指令，故IF\_Flush和ID\_Flush皆為1，如藍色框框。

**心得(Report)**

這次是要寫出pipeline 的 processor ，因為變數變得非常多，也有很多線段的連接，所以我覺得變得蠻複雜的，並不那麼容易，尤其在寫top的時候，必須要非常小心的一邊看圖片中線段如何相連，還要不斷檢查是否所連接的是變數是該階段而不是別的階段，常常一個不小心看錯，就要重頭檢查，不過也因此理解的東西比較深刻，完成時成就感也大增。