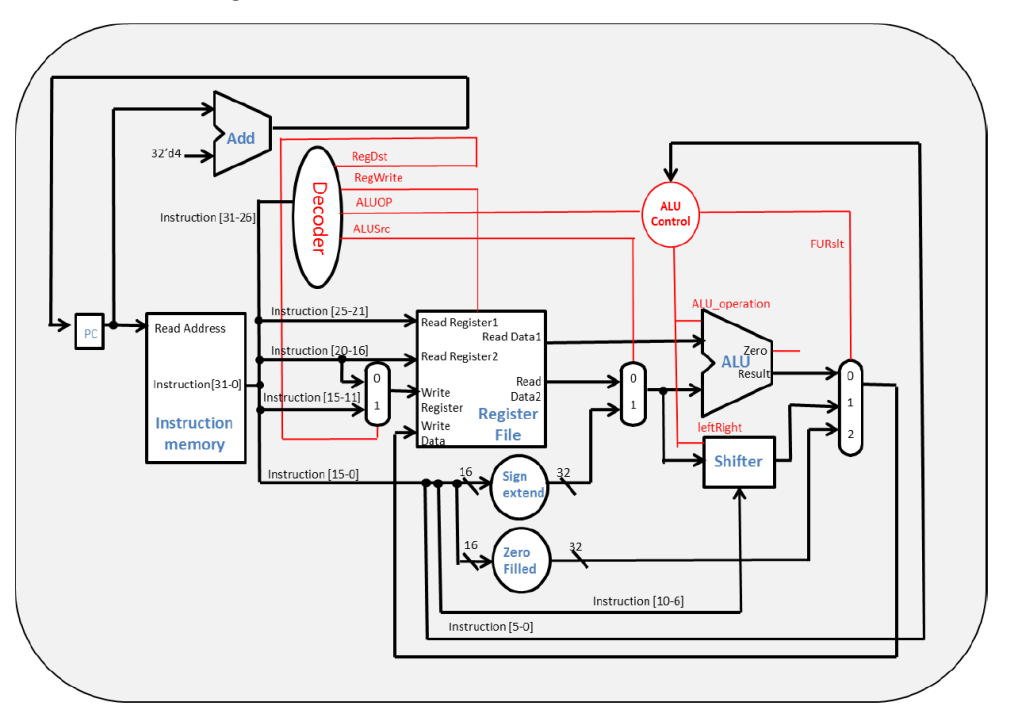
**Computer Organization**

**Architecture diagrams:**

****

**此圖為single cycle CPU的電路圖，用此來實作。**

**Hardware module analysis:**

**Decoder: 看op code來判斷出各種type的RegDst、RegWrite、ALUOP、ALUSrc並且讓每個mutiplexer判斷要選擇哪一條路走。**

**Sign extand:讓I type 原本的16 bits擴充成32bits來讓ALU 計算。**

**Zero Filled:當遇到LUI指令時，乘上再繼續去計算**

**Shifter: 用來乘或除2的倍數，向左或向右移動幾個bits**

**ALU:用來進行加減 and or slt之類的運算**

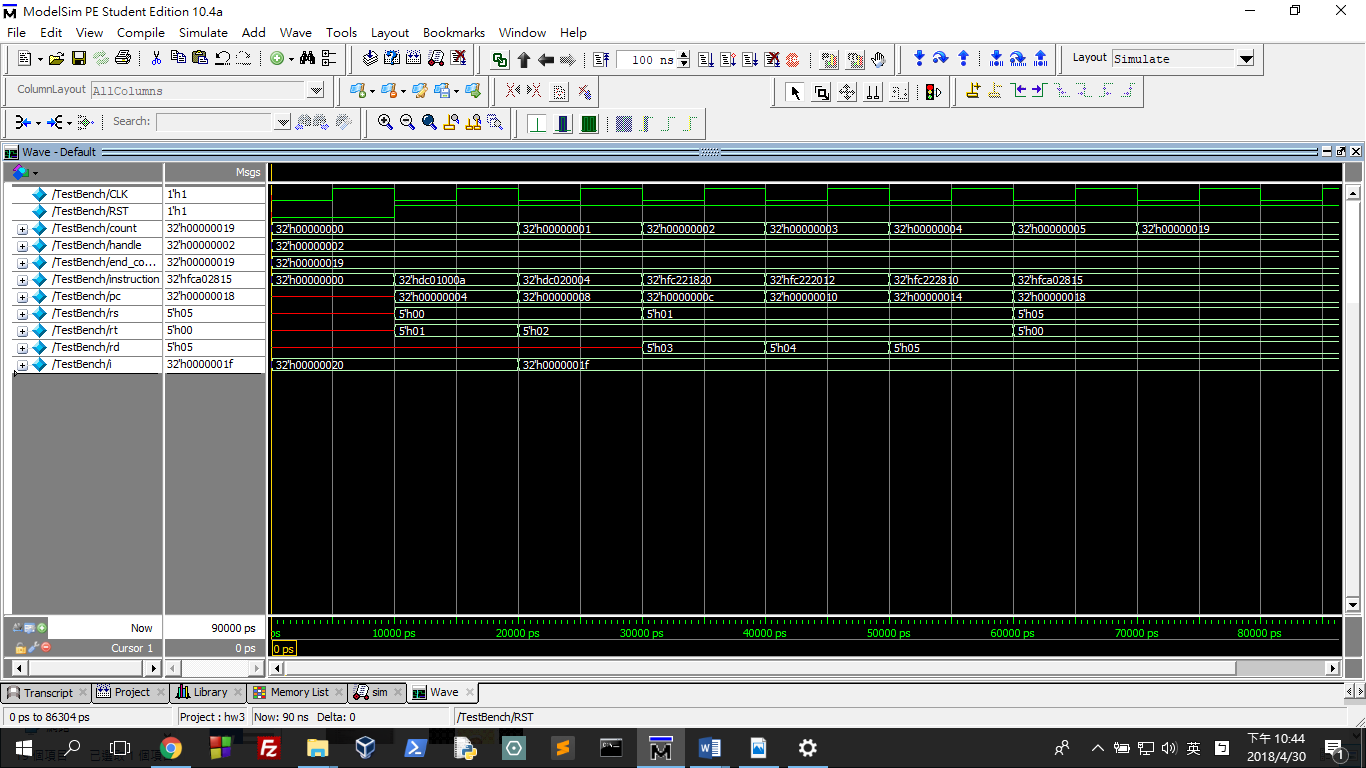
**ALU control:會根據ALU\_operation對不同指令給ALU不同動做ALU\_operation、FURslt、leftRight**

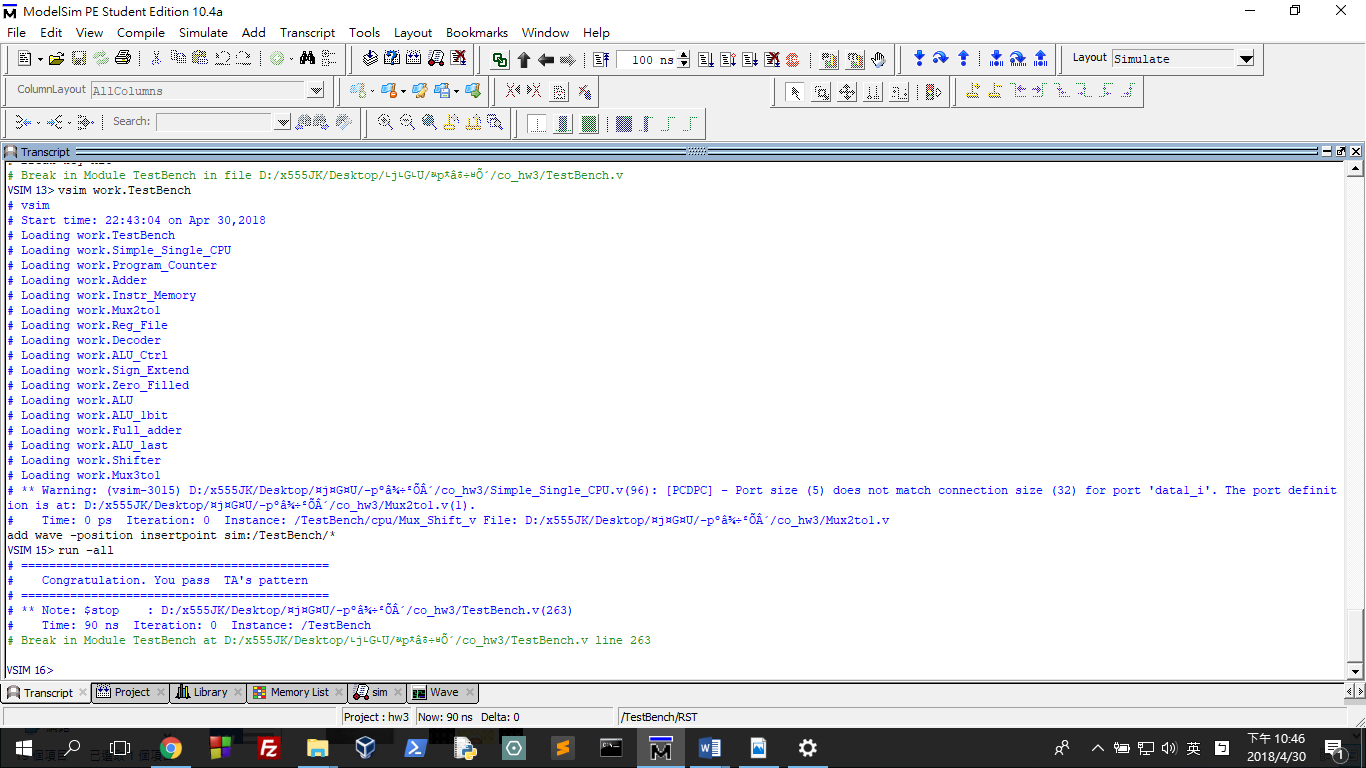
**Mux2to1:種共有兩個，第一個是讓Decoder給的RegDst來判斷目的是rs還是rd，第二個是讓Decoder給的ALUsrc 來判斷是由I type的sign extend還是Rtype 的rs 來的。**

**Mux3to1:從FURslt來看result是從ALU 或shifter或 zero filled來的。**

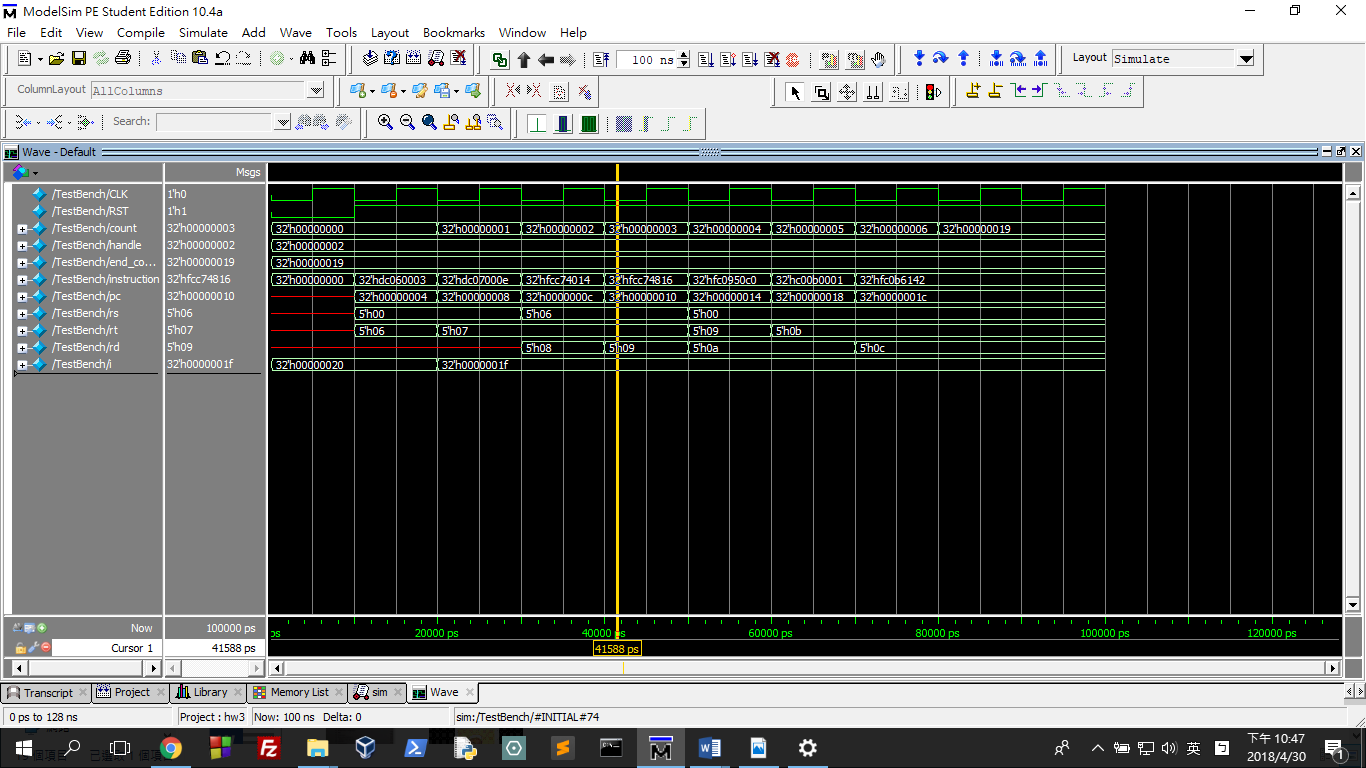
**Finished part:**

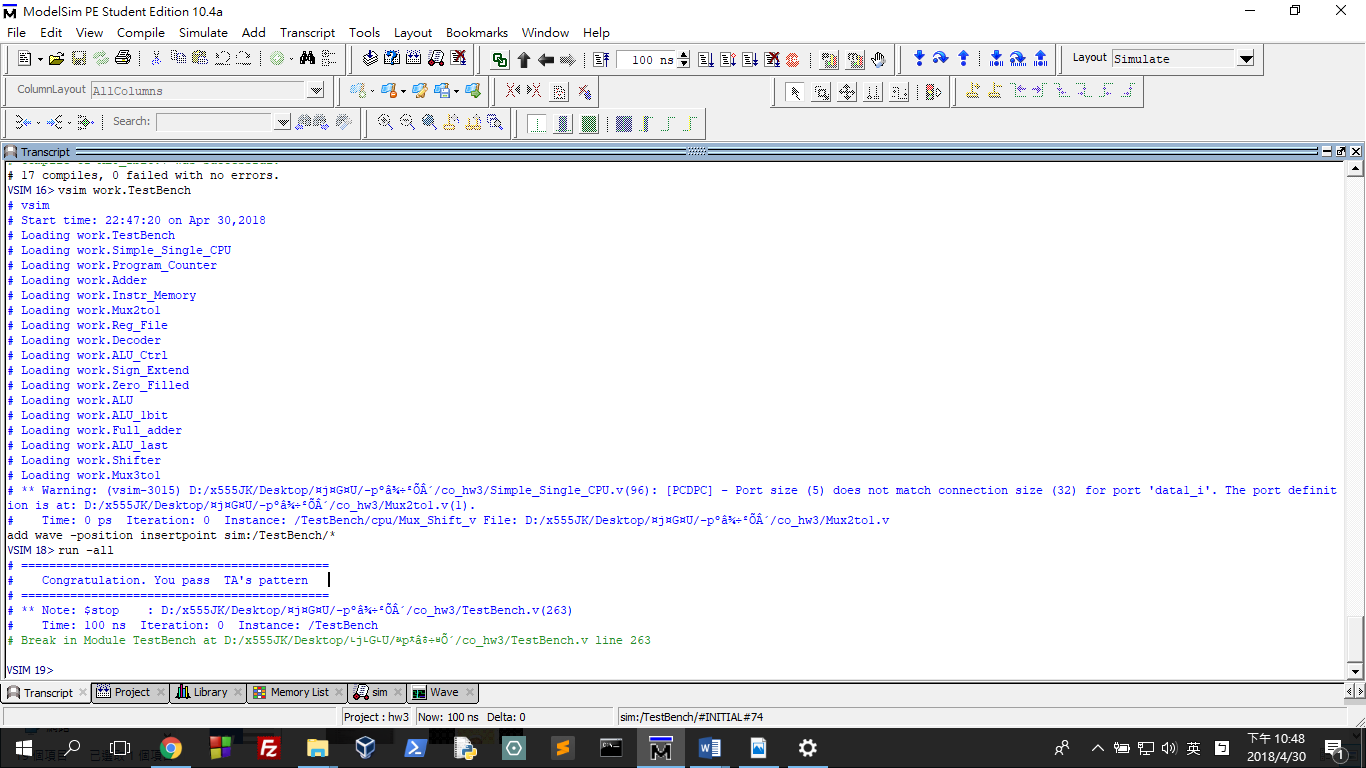
**Test1:**



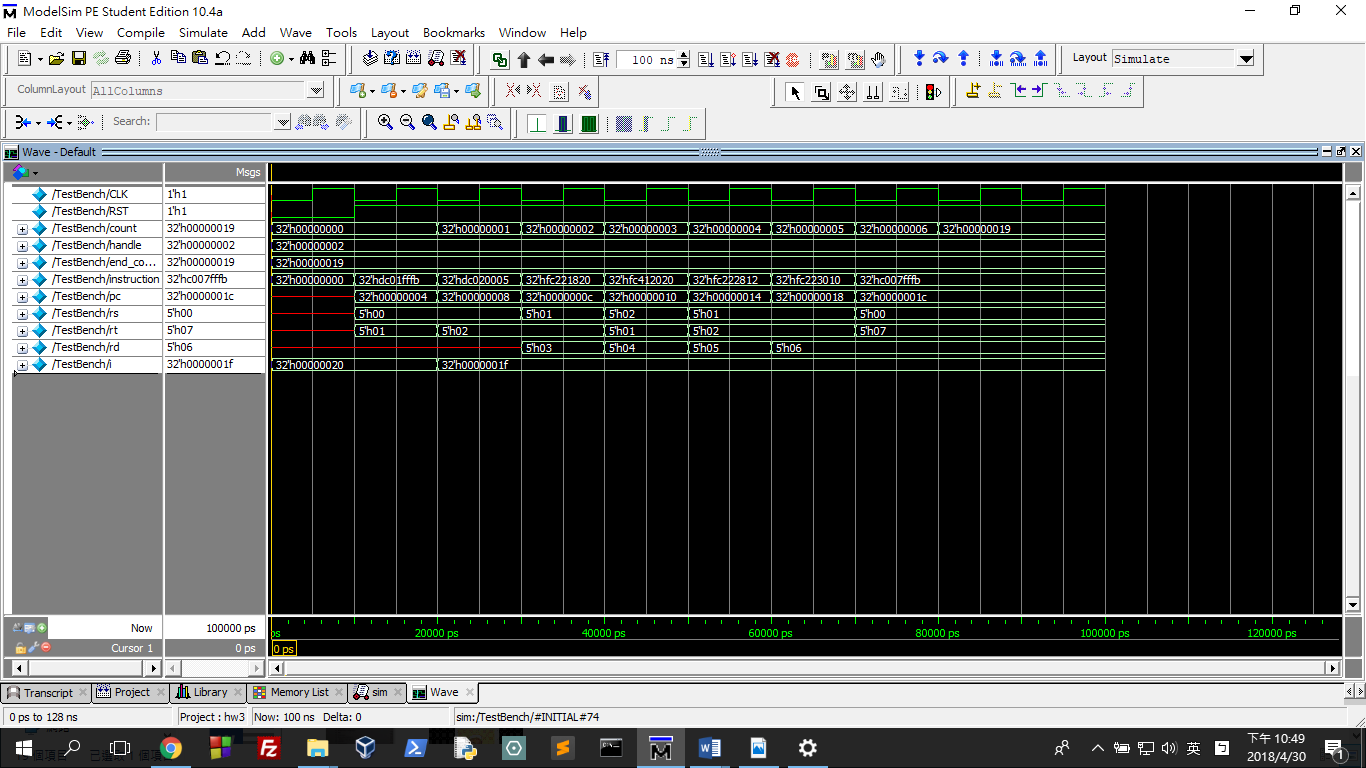


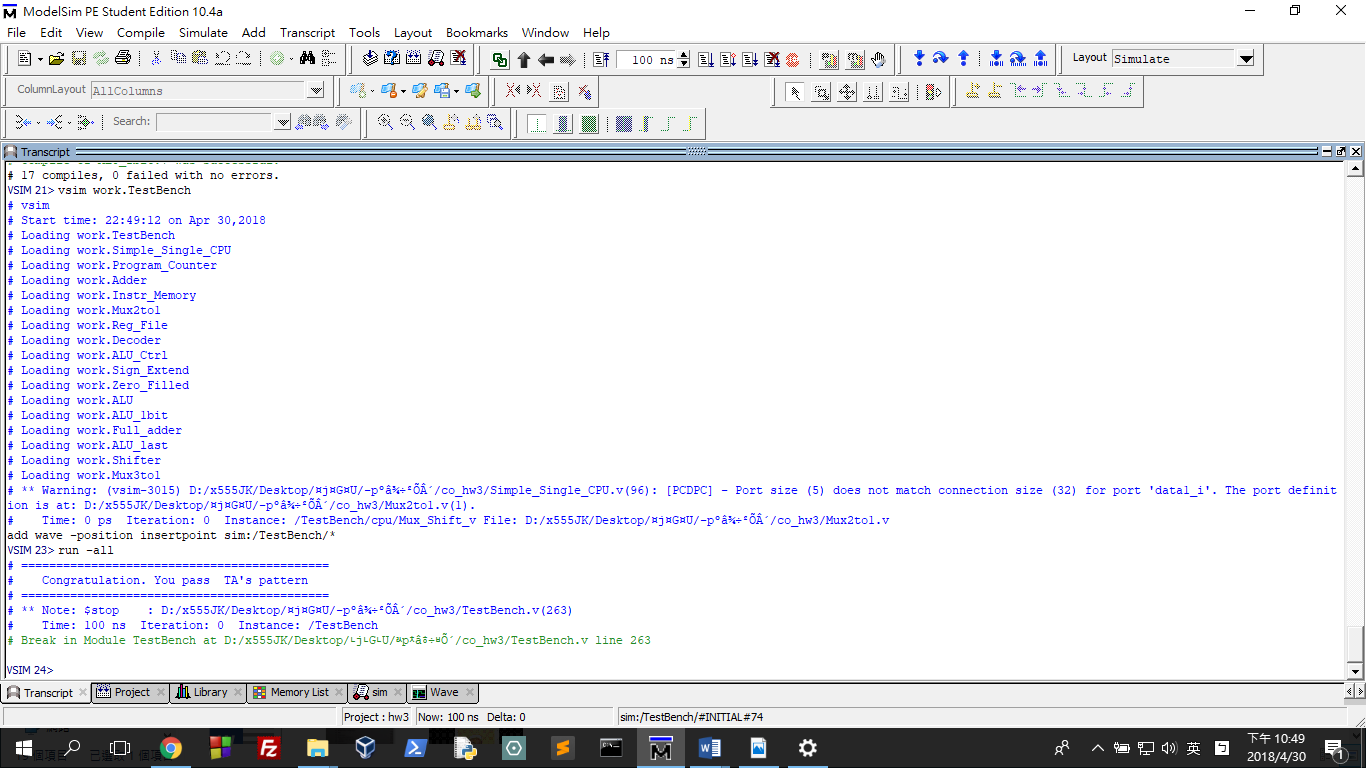
**Test2:**





**Test3:**





**Problems you met and solutions:**

**首先要解決的問題是要知道各種type的Decoder 和 ALU control的路線要怎麼跑，從先理解到整個店路跑的流程，再去用code去把每個gate串起來，我覺得再區分並且熟悉不同type指令的跑法讓我花了一點時間。**

**而大部分電路接好之後比較難的是debug，因為以前比較少用verilog debugn所以比較不知道要用甚麼方式看自己的錯誤在哪 code，之後再simulate有問人知道原來也可以用看wave的方式去判斷，在甚麼值的情況下會有問題，讓我可以去改code的邏輯，看wave是一個還不錯的debug方式。**

**Summary:**

**透過這次的lab讓我了解到了single cycle cpu的運作，以及各個線路的接法，還有不同type的指令，應該要跑到哪條對應的電路，先理解整個大架構的電路，再去看細節要如如何去要去如何連接，並且透過這次的lab也多練習到了一些以前比較不會用的verilog語法，希望透過這次的lab可以幫助我在之後寫更複雜cpu時更容易上手。**