1043335\_hw3\_實驗報告

## Source Code

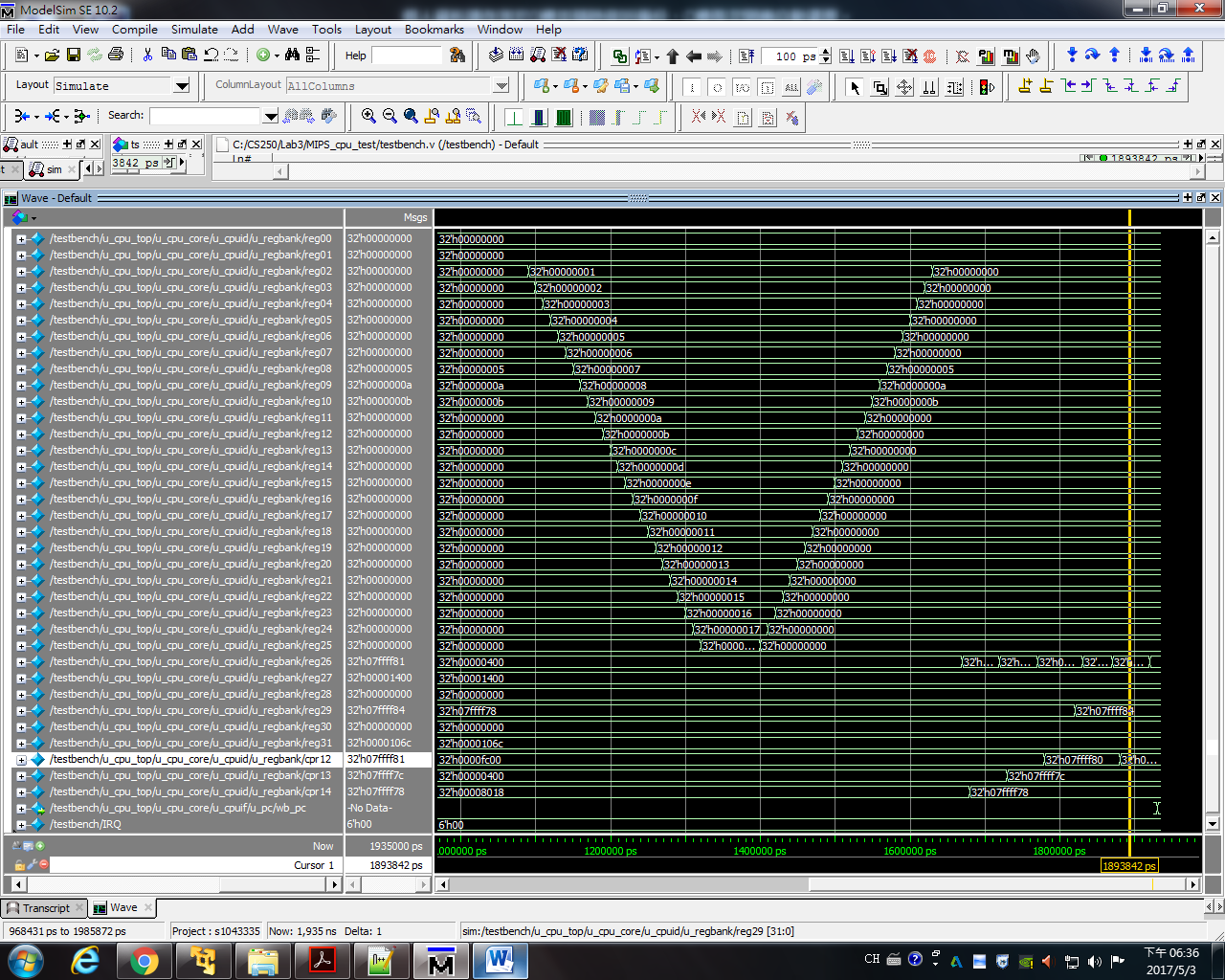
* inthandle.s

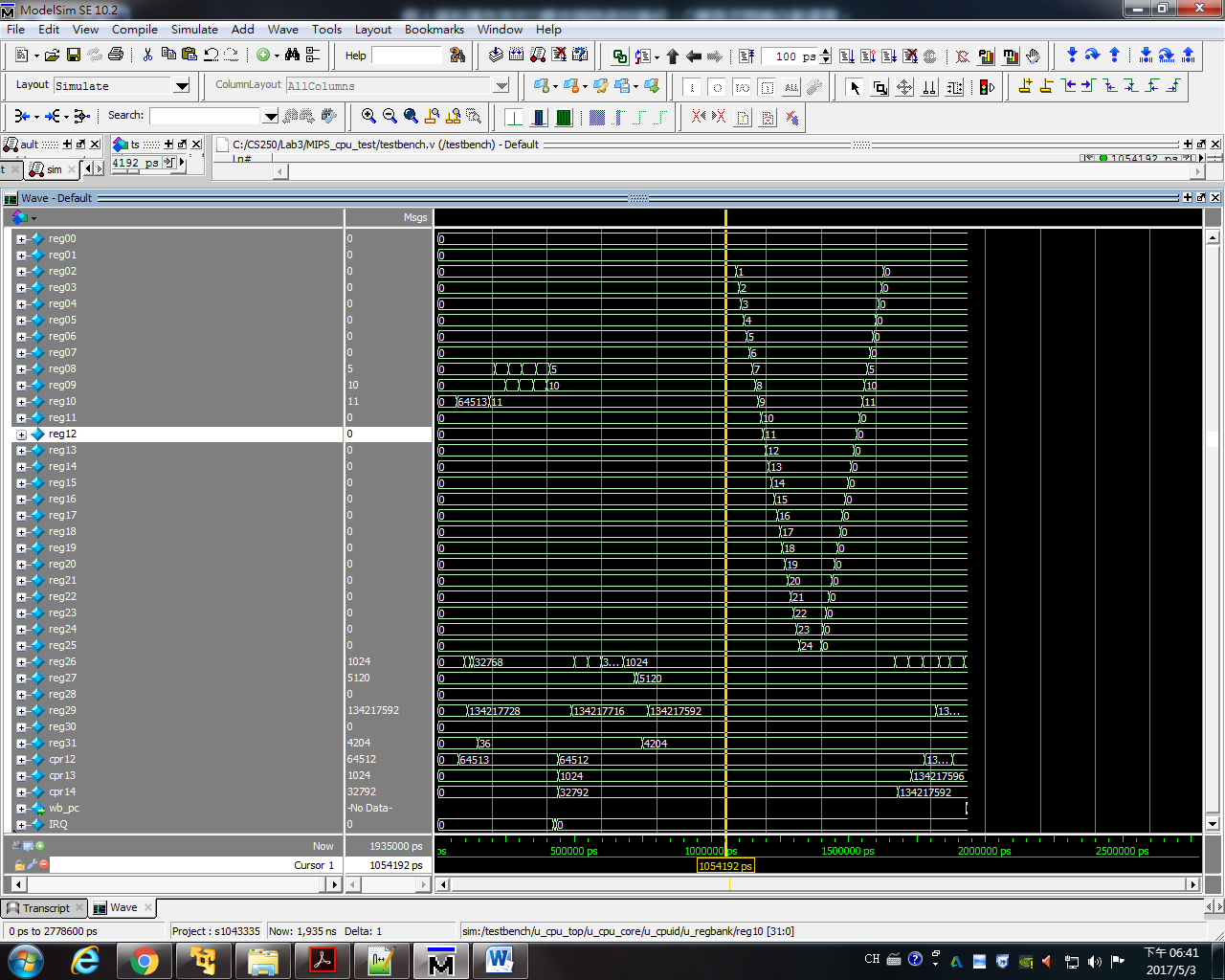
|  |  |
| --- | --- |
|  |  |

* intprocess.s

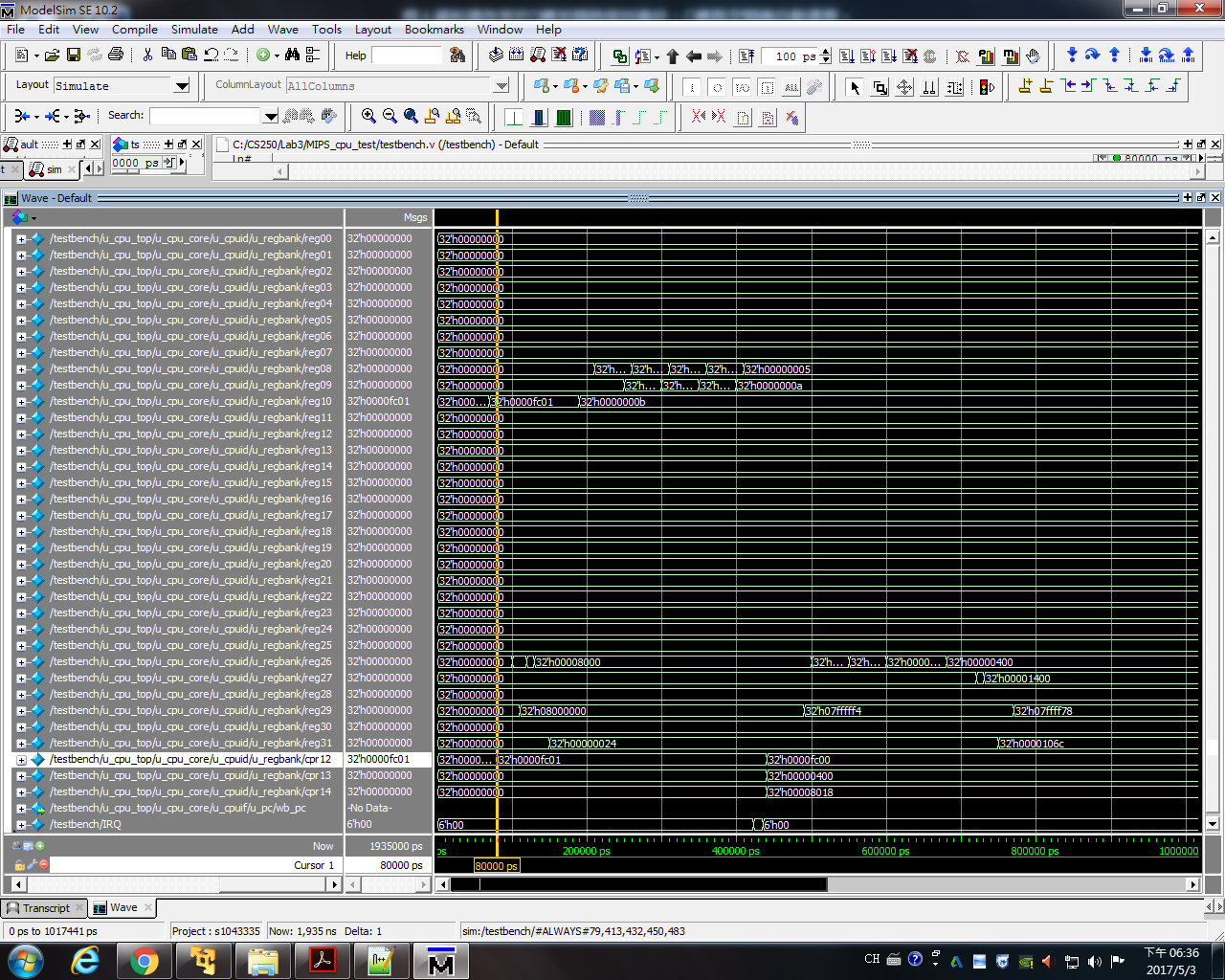
|  |  |
| --- | --- |
|  |  |

## 實驗結果

* 波形圖



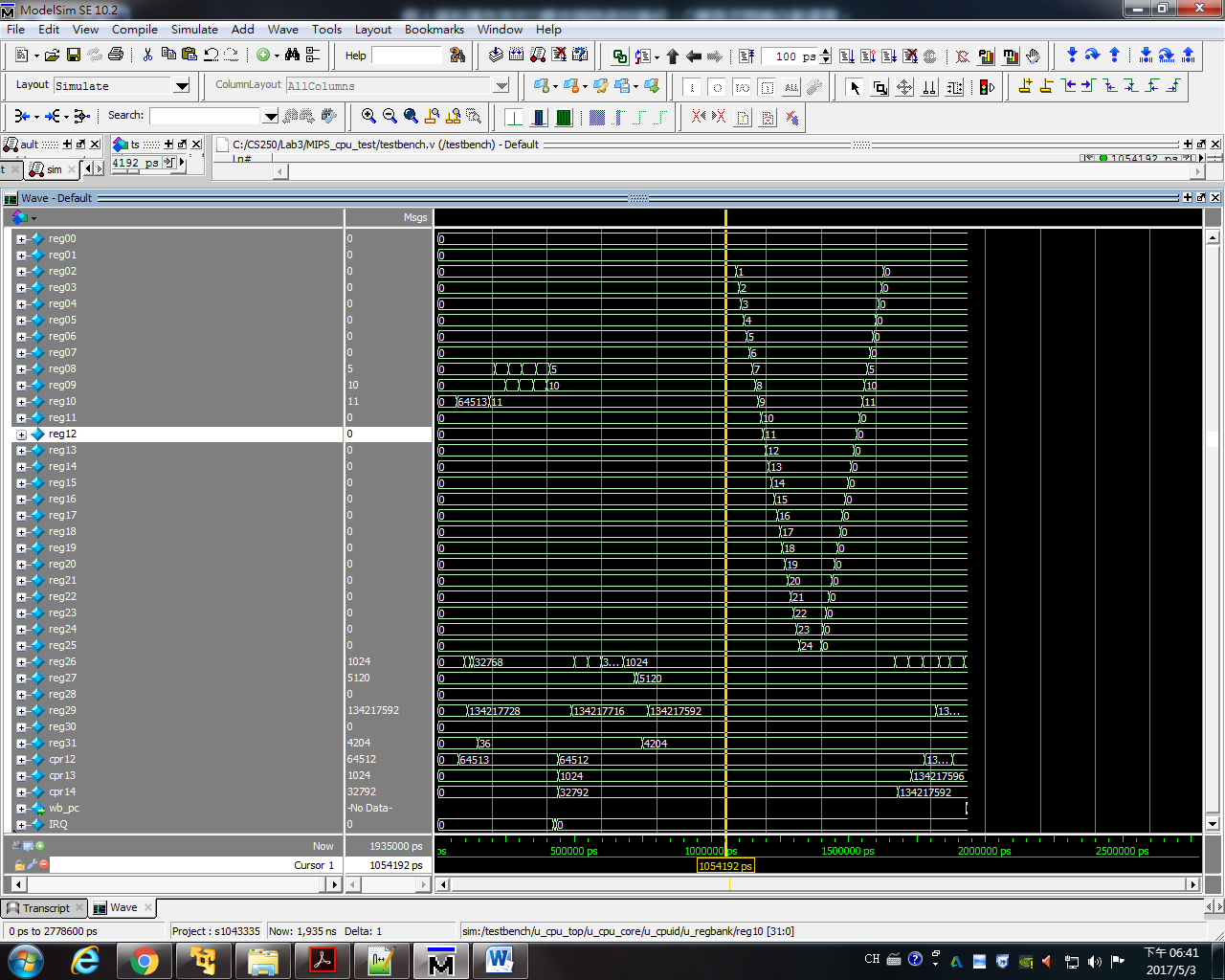
* 將數值轉成十進位
* interrupt流程



**2.**

**1.**

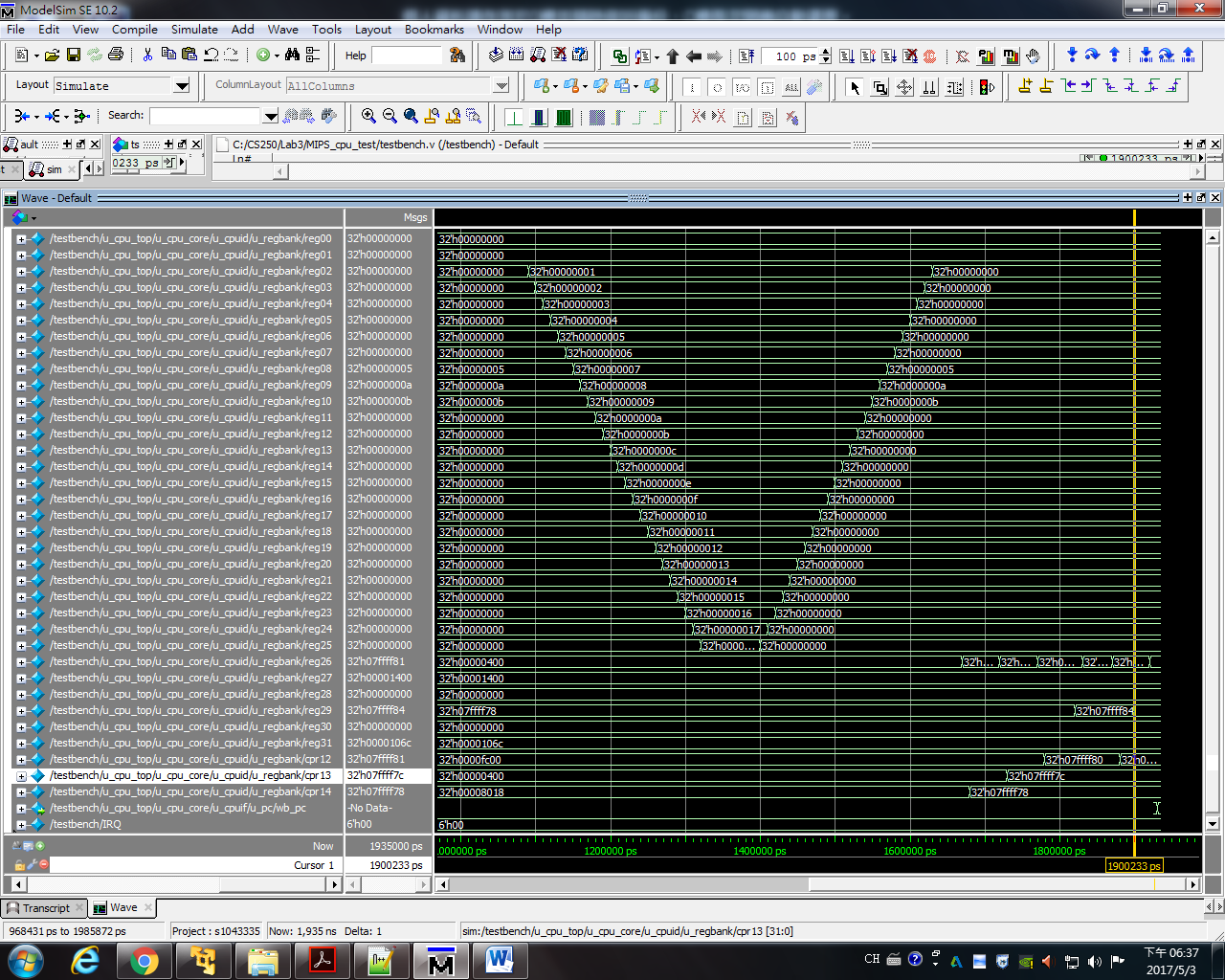
* 在編號1.的紅色框，大約為420000ps那可以看到interrupt enable的值變成0，代表我們已經收到interrupt的要求且進入了kernel mode，因此會跳去exception handler address，並且執行ISR。
* 在編號2.的紅色框，可以看到reg26( $k0 )，將 $12，$13，$14的值依序存入stack。
* 在白色箭頭那可以看到 $k0取到 $13的值，與 $k1的值相加後，為要跳去的位置(黃色箭頭)，0x0001400。



**4.**

**3.**

* 跳去對應的服務程式後，會把 $1~$31都存進stack，可以看到在編號3.的紅色框中 $8~$10有值。
* 再來可以看到暫存器的值被亂改成1~24，但是因為剛剛已經存進stack中了，所以可以把他還原回來，注意編號4.的紅色框中 $8~$10還原的是原本的值，其他的也是他們自己原本的值，代表還原成功。



**6.**

**5.**

* 跳回program被中斷的位置後，在編號5.的紅色框中，可以看到$k0取出之前存進stack的值放回 $12，$13，$14。
* 最後，把 $12 的enable和M2~M7設成1，編號6.的紅色框為圖上黃線那個時間點的值，可以看出值真的設成1了。
* 驗證項目:
* 1) GPRs是否成功備份、還原
* 2) EPC, Status, Cause是否成功備份、還原
* 3) Interrupt handler & ISR是否成功執行
* 4) 離開中斷程式之前,是否 enable interrupt

## 心得

* 實驗心得:

在上作業系統時，楊正仁老師就已經為我們詳細的介紹過interrupt的所有流程了，但是即使如此，在這次的實驗課中，才發現雖然已經學過，但是沒有實作經驗還是會有不懂的地方。

而且懂和會做真的是兩回事，在開始寫程式以後，雖然要寫的部分不多，但是還是很猶豫不知道這樣寫對不對，最後等到都寫完了，也編譯完做好波形後，發現有疏漏的地方，因此只好又再回去改。

等到終於都完成以後，卻發現觀察波形又有問題了，不過因為時間快到了所以只好先截好圖，再回家慢慢分析。

果然，做實驗最重要的一步就是事後檢討分析，原本猶豫不確定的地方，再我重新分析程式碼和我的波形以後，終於能順利將所學融會貫通了，不只我自己寫的程式，還有助教幫我們填好的那一部分程式碼也是。

所以我想，現在叫我再自己寫一份interrupt的code出來，應該是沒問題了。

* 對本次Lab的改進建議:

我覺得這次Lab很好玩，雖然剛開始有點緊張怕完成不了，而且過程中也出了很多錯，但是正是因為這樣，也學到很多而且在跌跌撞撞中更能釐清自己原本所模糊的地方。

雖然每次實驗都覺得時間有點不夠，但是不多不說，是真的很充實刺激。很謝謝助教可以幫我們填好一部份的程式碼和做好Makefile，讓我們能夠更快的完成實驗。

因為即使事後寫完分析報告以後都瞭解了，但是在還沒接觸過的時候，真的需要一段時間去摸索一下，所以有這些幫助的程式碼真的太讓人感動了。