**VLSISystemDesign**

**Homework-2**

**Part2**

**學號：N26991784**

**姓名：林宏達**

**1. (30 points) Write and verify SMILE CPU based on the specification. i. Implement the 19 instructions as listed**

**ii. Redesgin the FSM for the controller**

**iii. Register File size: 32x32bits**

**iv. Assume Instruction Memory and Data Memory with no delay**

**v. Instruction memory size:100Mx32bit**

**vi. Data memory size: 64Kx32bits**

**vii. timescale 1ns/10ps**

**viii. Clock period: 20ns**

**a. A summary in the beginning to state what has been done (such as SMILE CPU, synthesis, post-synthesis simulation, additional branch instruction with verification)**

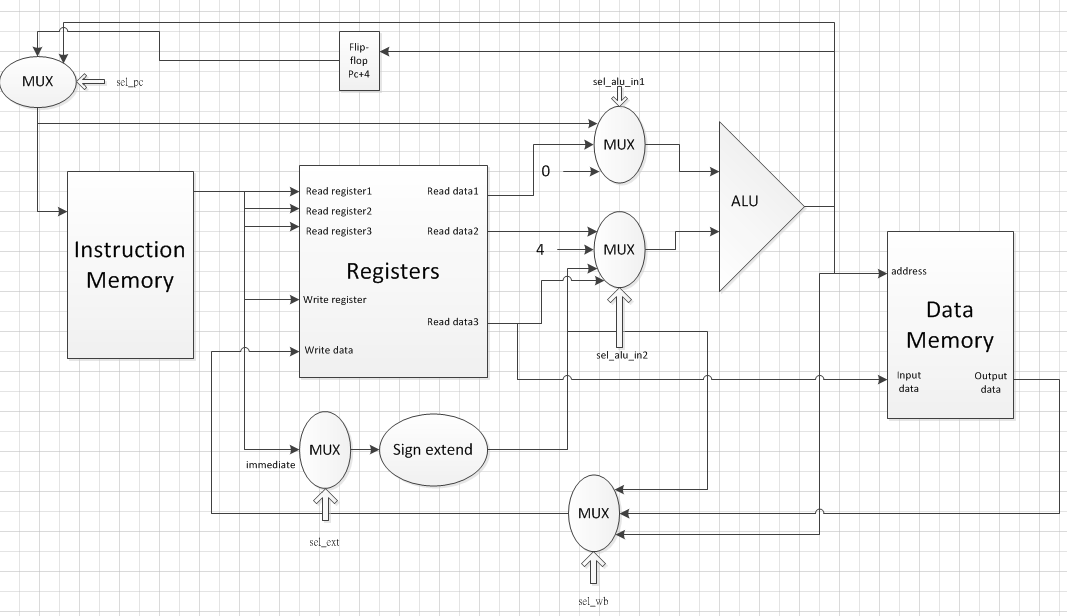
ANS：

這次的作業之前有些許的不一樣，除了跟之前一樣我做了ｐｒｅｓｉｍ的部分，這次因為作業需要加上了Ｂｒａｎｃｈ以及Ｊｕｍｐ的指令，另外我再狀態的部分修改成跟ＭＩＰＳ一樣，讓這顆ＣＰＵ更像ｍｕｌｔｉ－ｃｙｃｌｅ的ＣＰＵ（也就是Ｂｒａｎｃｈ與Ｊｕｍｐ都在３個ｓｔａｔｅ完成）。

**b. A block diagram for your completed SMILE CPU indicating all necessary components and I/O pins. Note please use MS Visio that is available in computer center in the university.**

ANS：

這次的圖跟上次沒有很大的分別，只是我在ｐｃ的部分多了選擇還有在ａｌｕ兩個輸入也都做了點改變，像是輸入１就多了多工器，而輸入２則是在多工器的選擇加了幾個選項，全部的ｍｏｄｕｌｅ除了ｃｏｎｔｒｏｌｌｅｒ沒畫上去外，我都是直接在ｍｕｘ的選擇打上選擇信號線，所以完整圖如下：



**C. Simulated waveforms with proper explanation**

ANS：

這次的TB分為兩部份，第一部分用的跟上次是一樣的數值，第二部分則是分為五個測試的地方，由自己寫的machine code分別去測試自己寫出來的CPU跑出來的結果是否正確。

以下為第一部分(a)TB的ｍｉｎｓ.ｐｒｏｇ裡面code：

0\_100010\_00000\_\_00000\_000000000000011 //MOVI R0=20'd3

0\_011100\_00000\_\_00001\_00001\_00\_00001010//SW M0=R0

0\_011100\_00001\_\_00001\_00001\_00\_00000010//LW R1=M0

0\_101000\_00001\_00001\_\_00000000000\_1001//ADDI R1=R1+4'b1001

0\_101100\_00000\_00000\_\_00000000000\_0100//ORI R0=R0|4'b0100

0\_101011\_00001\_00001\_\_00000000000\_1010//XORI R1=R1^4'b1010

0\_100000\_00000\_00000\_00000\_\_00000\_\_01001//NOP

0\_100000\_00001\_00000\_00001\_\_00000\_\_00000//ADD R1=R0+R1

0\_100000\_00001\_00001\_00000\_\_00000\_\_00001//SUB R1=R1-R0//R1=6

0\_100000\_00001\_00001\_00000\_\_00000\_\_00010//AND R1=R1&R0

0\_011100\_00001\_\_00001\_00000\_01\_00001010//SW M20=R1

0\_011100\_00010\_\_00001\_00000\_01\_00000010//LW R2=M20

0\_100000\_00010\_00001\_00000\_\_00000\_\_00100//OR R2=R1|R0

0\_100000\_00010\_00010\_00001\_\_00000\_\_00011//XOR R2=R2^R1

0\_100000\_00010\_00000\_00010\_\_00000\_\_01001//SRLI R2=R0 SRL(2)

0\_100000\_00010\_00010\_00100\_\_00000\_\_01000//SLLI R2=R2 SLL(4)

0\_100000\_00010\_00000\_11110\_\_00000\_\_01011//ROTRI R2=R0 ROTR(30)

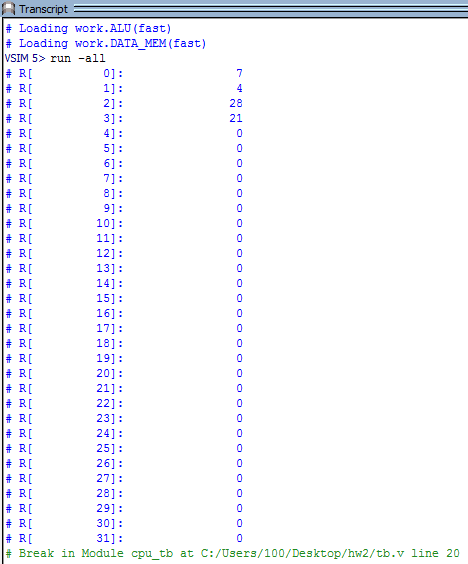
0\_011100\_00010\_\_00011\_00010\_01\_00001010//SW M56=R2

0\_011100\_00011\_\_00011\_00010\_01\_00000010//LW R3=M56

0\_100000\_00011\_00011\_00000\_\_00000\_\_00001//SUB R3=R3-R0

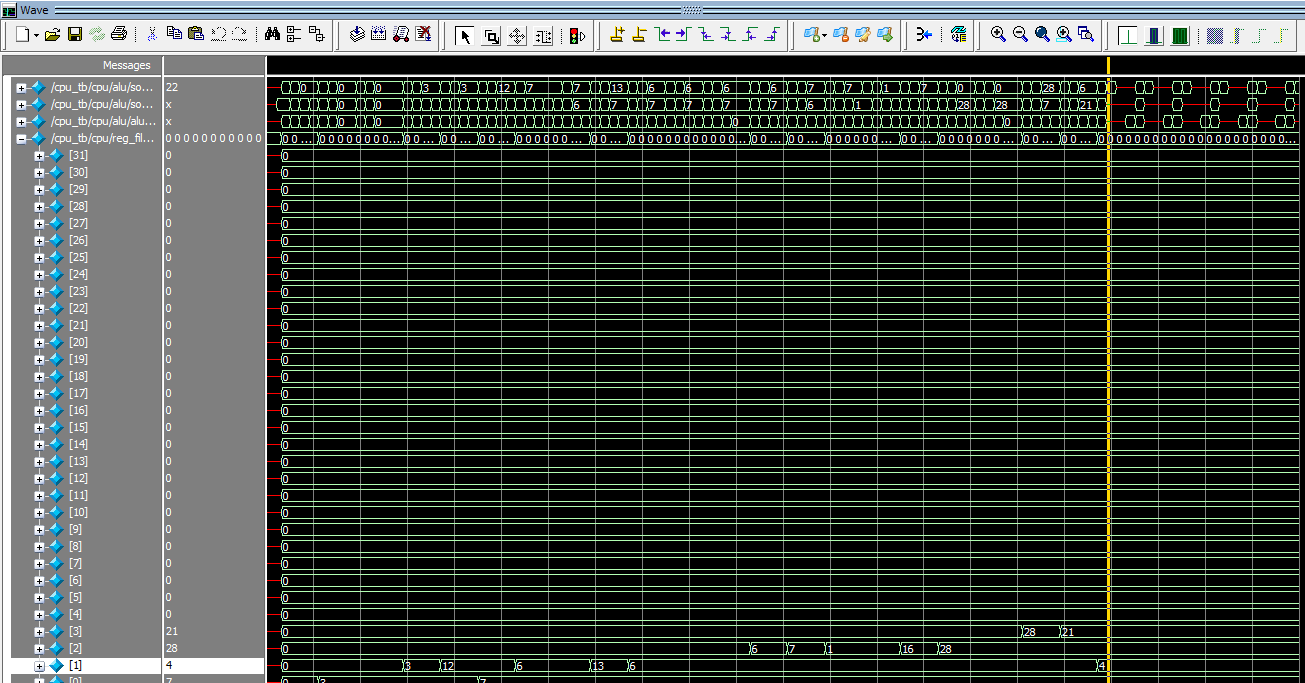
0\_100000\_00001\_00001\_00011\_\_00000\_\_00010//AND R1=R1&R3

我在TB的部分加了暫存器display出來，使我的程式觀看比較方便。如下圖：



Ｒ０＝7、Ｒ１＝4、Ｒ２＝28、Ｒ３＝２１。

而波形圖如下：



因為列出全部設計的線太多太雜，所以我在這邊只列出我暫存器裡面所存的值，以及ａｌｕ的ｉｎ與ｒｅｓｕｌｔ作為比較。

以下為第一部分(ｂ)TB的ｍｉｎｓ＿ｉ.ｐｒｏｇ裡面code：

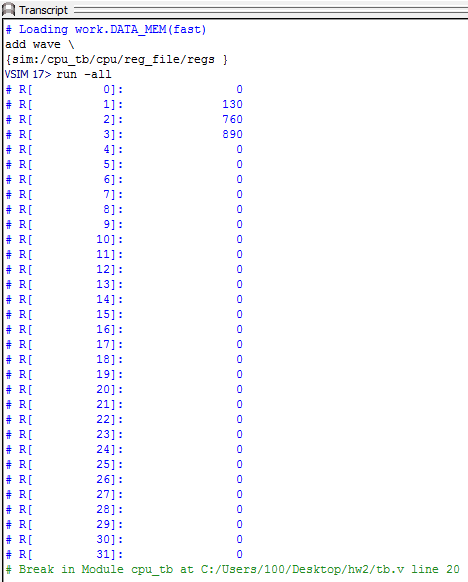
//130+760

0\_100010\_00001\_00000000000010000010 //R1 = 130

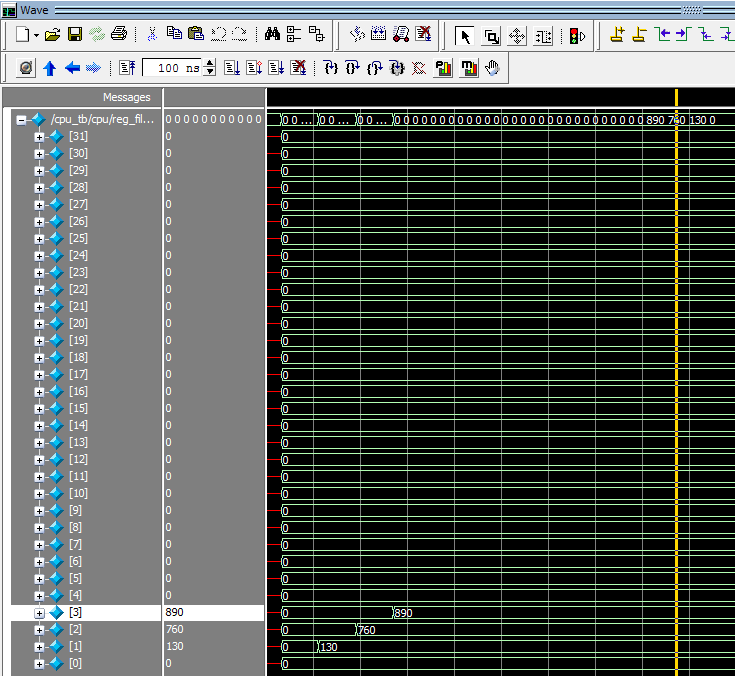
0\_100010\_00010\_00000000001011111000 //R2 = 760

0\_100000\_00011\_00001\_00010\_00000\_00000 //R1 + R2 = R3

ｄｉｓｐｌａｙ的值為：



暫存器的波形圖為：



以下為第一部分(ｂ)TB的ｍｉｎｓ＿ｉｉ.ｐｒｏｇ裡面code：

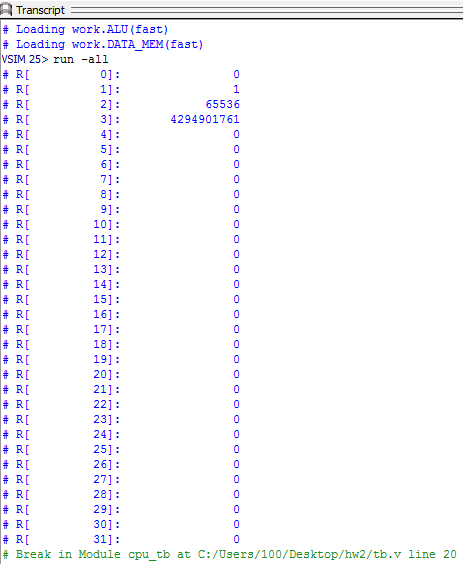
//1-65536

0\_100010\_00001\_0000\_0000\_0000\_0000\_0001 //R1 = 1

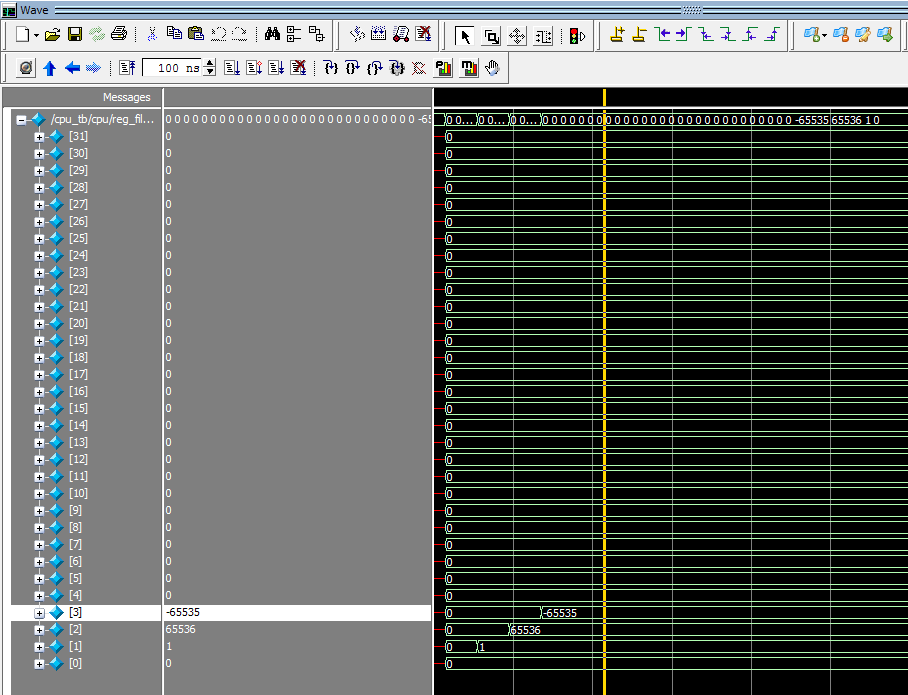
0\_100010\_00010\_0001\_0000\_0000\_0000\_0000 //R2 = 65536

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = -65535

ｄｉｓｐｌａｙ的值為：



暫存器的波形圖為：



以下為第一部分(ｂ)TB的ｍｉｎｓ＿ｉｉｉ.ｐｒｏｇ裡面code：

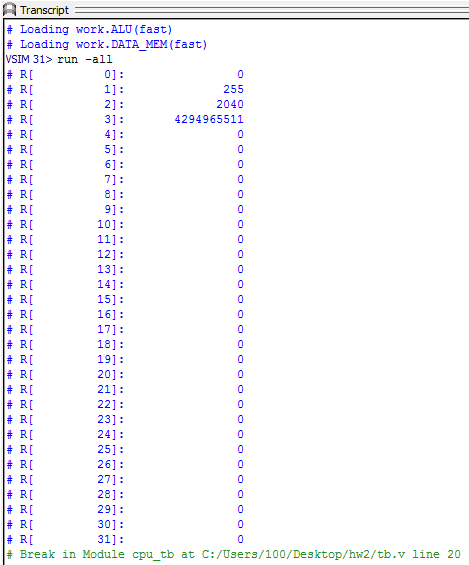
//255\*(-7)

0\_100010\_00001\_0000\_0000\_0000\_1111\_1111 //R1 = 255

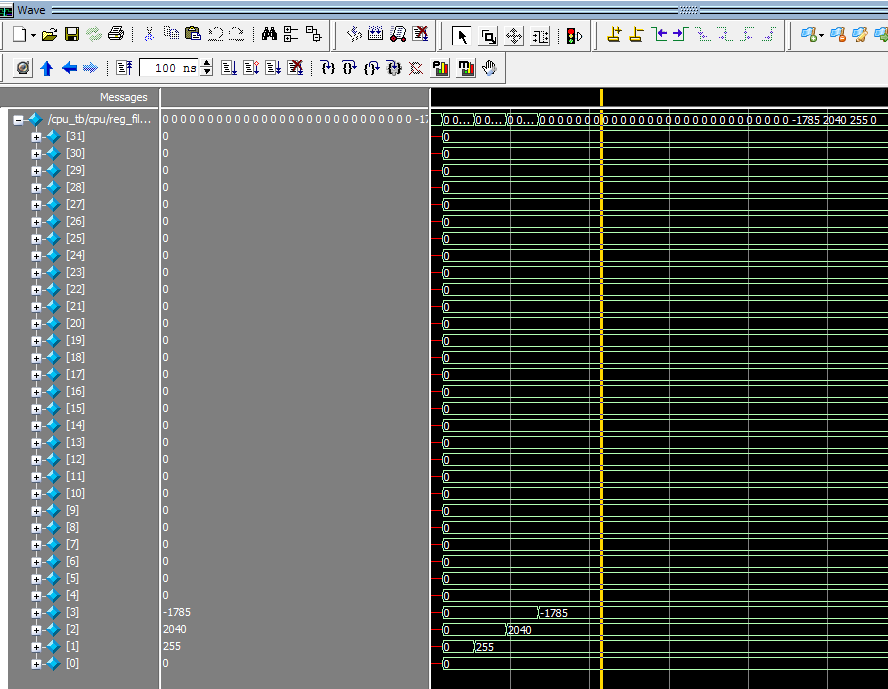
0\_100000\_00010\_00001\_00011\_00000\_01000 //R2 = 255<<8

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = R1 - R2

ｄｉｓｐｌａｙ的值為：



暫存器的波形圖為：



以下為第一部分(ｂ)TB的ｍｉｎｓ＿ｉｖ.ｐｒｏｇ裡面code：

//40\*2-160/4

0\_100010\_00001\_0000\_0000\_0000\_0010\_1000 //R1 = 40

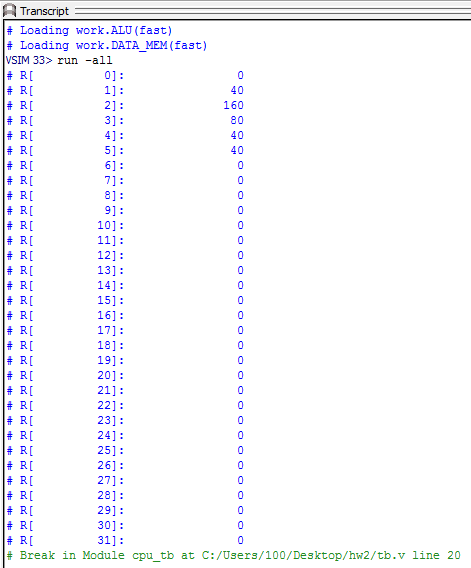
0\_100010\_00010\_0000\_0000\_0000\_1010\_0000 //R2 = 160

0\_100000\_00011\_00001\_00001\_00000\_01000 //R3 = R1<<1 = 40\*2

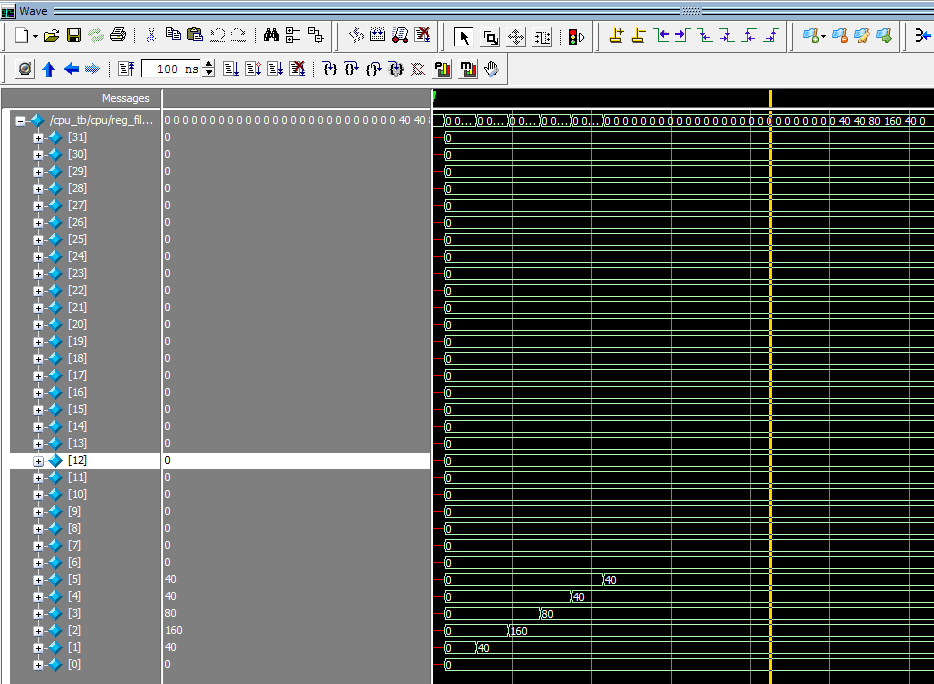
0\_100000\_00100\_00010\_00010\_00000\_01001 //R4 = R2>>2 = 160/4

0\_100000\_00101\_00011\_00100\_00000\_00001 //R5 = R3 - R4

ｄｉｓｐｌａｙ的值為：



暫存器的波形圖為：



以下為第一部分(ｂ)TB的ｍｉｎｓ＿ｖ＿ｄａｔａ１.ｐｒｏｇ裡面code：

ｄａｔａ１我是用最壞的情況也就是大到小的排序（正確為小到大排序）

也就是我Ｍ［２０：１］（４個為一個單位）存的值分別為：５．４．３．２．１。

0\_100010\_00000\_0000\_0000\_0000\_0000\_0001 //R0 = 1

0\_100010\_00001\_0000\_0000\_0000\_0000\_0101 //R1 = 5

0\_100010\_00010\_0000\_0000\_0000\_0000\_0100 //R2 = 4

0\_100010\_00011\_0000\_0000\_0000\_0000\_0011 //R3 = 3

0\_100010\_00100\_0000\_0000\_0000\_0000\_0010 //R4 = 2

0\_100010\_00101\_0000\_0000\_0000\_0000\_0001 //R5 = 1

0\_001010\_00001\_00000\_\_00000\_00000\_00000 //M4 = R1 = 5

0\_001010\_00010\_00000\_\_00000\_00000\_00001 //M8 = R2 = 4

0\_001010\_00011\_00000\_\_00000\_00000\_00010 //M12 = R3 = 3

0\_001010\_00100\_00000\_\_00000\_00000\_00011 //M16 = R4 = 2

0\_001010\_00101\_00000\_\_00000\_00000\_00100 //M20 = R5 = 1

0\_000010\_00001\_00000\_\_00000\_00000\_00000 //R1 = M4 = 5

0\_000010\_00010\_00000\_\_00000\_00000\_00001 //R2 = M8 = 4

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = R1 - R2

0\_100000\_00100\_00011\_11111\_00000\_01001 //R4 = R3>>31

0\_100110\_00100\_00000\_00000\_00000\_01000 //BEQ(R4==R0)

0\_001010\_00010\_00000\_\_00000\_00000\_00000 //M4 = R2

0\_001010\_00001\_00000\_\_00000\_00000\_00001 //M8 = R1

0\_100100\_0\_00000\_00000\_00000\_00000\_0110 //j

0\_001010\_00001\_00000\_\_00000\_00000\_00000 //M4 = R1

0\_001010\_00010\_00000\_\_00000\_00000\_00001 //M8 = R2

0\_000010\_00001\_00000\_\_00000\_00000\_00000 //R1 = M4 = 4

0\_000010\_00010\_00000\_\_00000\_00000\_00010 //R2 = M12 = 3

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = R1 - R2

0\_100000\_00100\_00011\_11111\_00000\_01001 //R4 = R3>>31

0\_100110\_00100\_00000\_00000\_00000\_01000 //BEQ(R4==R0)

0\_001010\_00010\_00000\_\_00000\_00000\_00000 //M4 = R2

0\_001010\_00001\_00000\_\_00000\_00000\_00010 //M12 = R1

0\_100100\_0\_00000\_00000\_00000\_00000\_0110 //j

0\_001010\_00001\_00000\_\_00000\_00000\_00000 //M4 = R1

0\_001010\_00010\_00000\_\_00000\_00000\_00010 //M12 = R2

0\_000010\_00001\_00000\_\_00000\_00000\_00000 //R1 = M4 = 3

0\_000010\_00010\_00000\_\_00000\_00000\_00011 //R2 = M16 = 2

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = R1 - R2

0\_100000\_00100\_00011\_11111\_00000\_01001 //R4 = R3>>31

0\_100110\_00100\_00000\_00000\_00000\_01000 //BEQ(R4==R0)

0\_001010\_00010\_00000\_\_00000\_00000\_00000 //M4 = R2

0\_001010\_00001\_00000\_\_00000\_00000\_00011 //M16 = R1

0\_100100\_0\_00000\_00000\_00000\_00000\_0110 //j

0\_001010\_00001\_00000\_\_00000\_00000\_00000 //M4 = R1

0\_001010\_00010\_00000\_\_00000\_00000\_00011 //M16 = R2

0\_000010\_00001\_00000\_\_00000\_00000\_00000 //R1 = M4

0\_000010\_00010\_00000\_\_00000\_00000\_00100 //R2 = M20

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = R1 - R2

0\_100000\_00100\_00011\_11111\_00000\_01001 //R4 = R3>>31

0\_100110\_00100\_00000\_00000\_00000\_01000 //BEQ(R4==R0)

0\_001010\_00010\_00000\_\_00000\_00000\_00000 //M4 = R2

0\_001010\_00001\_00000\_\_00000\_00000\_00100 //M20 = R1

0\_100100\_0\_00000\_00000\_00000\_00000\_0110 //j

0\_001010\_00001\_00000\_\_00000\_00000\_00000 //M4 = R1

0\_001010\_00010\_00000\_\_00000\_00000\_00100 //M20 = R2

0\_000010\_00001\_00000\_\_00000\_00000\_00001 //R1 = M8

0\_000010\_00010\_00000\_\_00000\_00000\_00010 //R2 = M12

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = R1 - R2

0\_100000\_00100\_00011\_11111\_00000\_01001 //R4 = R3>>31

0\_100110\_00100\_00000\_00000\_00000\_01000 //BEQ(R4==R0)

0\_001010\_00010\_00000\_\_00000\_00000\_00001 //M8 = R2

0\_001010\_00001\_00000\_\_00000\_00000\_00010 //M12 = R1

0\_100100\_0\_00000\_00000\_00000\_00000\_0110 //j

0\_001010\_00001\_00000\_\_00000\_00000\_00001 //M8 = R1

0\_001010\_00010\_00000\_\_00000\_00000\_00010 //M12 = R2

0\_000010\_00001\_00000\_\_00000\_00000\_00001 //R1 = M8

0\_000010\_00010\_00000\_\_00000\_00000\_00011 //R2 = M16

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = R1 - R2

0\_100000\_00100\_00011\_11111\_00000\_01001 //R4 = R3>>31

0\_100110\_00100\_00000\_00000\_00000\_01000 //BEQ(R4==R0)

0\_001010\_00010\_00000\_\_00000\_00000\_00001 //M8 = R2

0\_001010\_00001\_00000\_\_00000\_00000\_00011 //M16 = R1

0\_100100\_0\_00000\_00000\_00000\_00000\_0110 //j

0\_001010\_00001\_00000\_\_00000\_00000\_00001 //M8 = R1

0\_001010\_00010\_00000\_\_00000\_00000\_00011 //M16 = R2

0\_000010\_00001\_00000\_\_00000\_00000\_00001 //R1 = M8

0\_000010\_00010\_00000\_\_00000\_00000\_00100 //R2 = M20

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = R1 - R2

0\_100000\_00100\_00011\_11111\_00000\_01001 //R4 = R3>>31

0\_100110\_00100\_00000\_00000\_00000\_01000 //BEQ(R4==R0)

0\_001010\_00010\_00000\_\_00000\_00000\_00001 //M8 = R2

0\_001010\_00001\_00000\_\_00000\_00000\_00100 //M20 = R1

0\_100100\_0\_00000\_00000\_00000\_00000\_0110 //j

0\_001010\_00001\_00000\_\_00000\_00000\_00001 //M8 = R1

0\_001010\_00010\_00000\_\_00000\_00000\_00100 //M20 = R2

0\_000010\_00001\_00000\_\_00000\_00000\_00010 //R1 = M12

0\_000010\_00010\_00000\_\_00000\_00000\_00011 //R2 = M16

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = R1 - R2

0\_100000\_00100\_00011\_11111\_00000\_01001 //R4 = R3>>31

0\_100110\_00100\_00000\_00000\_00000\_01000 //BEQ(R4==R0)

0\_001010\_00010\_00000\_\_00000\_00000\_00010 //M12 = R2

0\_001010\_00001\_00000\_\_00000\_00000\_00011 //M16 = R1

0\_100100\_0\_00000\_00000\_00000\_00000\_0110 //j

0\_001010\_00001\_00000\_\_00000\_00000\_00010 //M12 = R1

0\_001010\_00010\_00000\_\_00000\_00000\_00011 //M16 = R2

0\_000010\_00001\_00000\_\_00000\_00000\_00010 //R1 = M12

0\_000010\_00010\_00000\_\_00000\_00000\_00100 //R2 = M20

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = R1 - R2

0\_100000\_00100\_00011\_11111\_00000\_01001 //R4 = R3>>31

0\_100110\_00100\_00000\_00000\_00000\_01000 //BEQ(R4==R0)

0\_001010\_00010\_00000\_\_00000\_00000\_00010 //M12 = R2

0\_001010\_00001\_00000\_\_00000\_00000\_00100 //M20 = R1

0\_100100\_0\_00000\_00000\_00000\_00000\_0110 //j

0\_001010\_00001\_00000\_\_00000\_00000\_00010 //M12 = R1

0\_001010\_00010\_00000\_\_00000\_00000\_00100 //M20 = R2

0\_000010\_00001\_00000\_\_00000\_00000\_00011 //R1 = M16

0\_000010\_00010\_00000\_\_00000\_00000\_00100 //R2 = M20

0\_100000\_00011\_00001\_00010\_00000\_00001 //R3 = R1 - R2

0\_100000\_00100\_00011\_11111\_00000\_01001 //R4 = R3>>31

0\_100110\_00100\_00000\_00000\_00000\_01000 //BEQ(R4==R0)

0\_001010\_00010\_00000\_\_00000\_00000\_00011 //M16 = R2

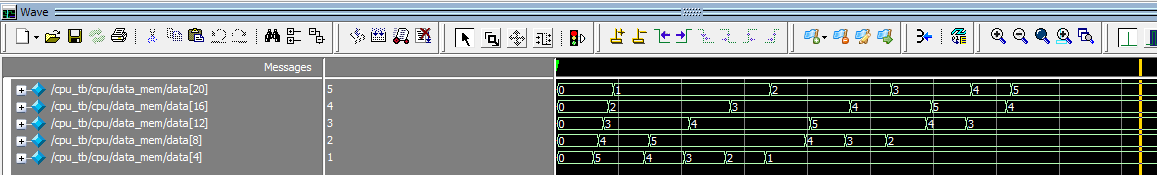
0\_001010\_00001\_00000\_\_00000\_00000\_00100 //M20 = R1

0\_100100\_0\_00000\_00000\_00000\_00000\_0110 //j

0\_001010\_00001\_00000\_\_00000\_00000\_00011 //M16 = R1

0\_001010\_00010\_00000\_\_00000\_00000\_00100 //M20 = R2

波形圖為：



以下為第一部分(ｂ)TB的ｍｉｎｓ＿ｖ＿ｄａｔａ２.ｐｒｏｇ裡面code：

ｄａｔａ２是用最好的情況也就是小到大的排序（正確為小到大排序）

也就是我Ｍ［２０：１］（４個為一個單位）存的值分別為：１．２．３．４．５。因為下半部跟ｄａｔａ１一樣所以我只貼上半部ｍｏｖｉ值的部分。

0\_100010\_00000\_0000\_0000\_0000\_0000\_0001 //R0 = 1

0\_100010\_00001\_0000\_0000\_0000\_0000\_0001 //R1 = 1

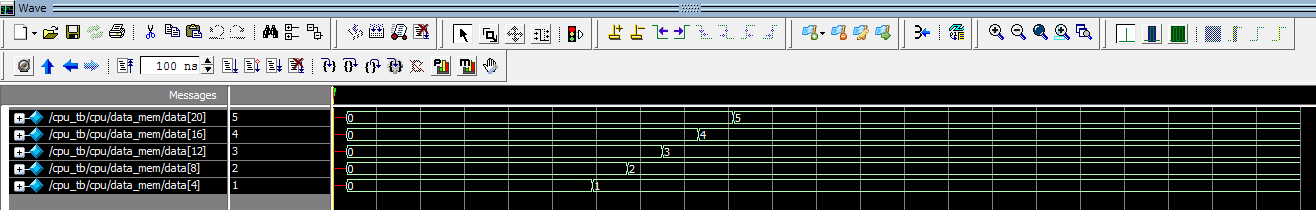
0\_100010\_00010\_0000\_0000\_0000\_0000\_0010 //R2 = 2

0\_100010\_00011\_0000\_0000\_0000\_0000\_0011 //R3 = 3

0\_100010\_00100\_0000\_0000\_0000\_0000\_0100 //R4 = 4

0\_100010\_00101\_0000\_0000\_0000\_0000\_0101 //R5 = 5

波形圖為：



以下為第一部分(ｂ)TB的ｍｉｎｓ＿ｖ＿ｄａｔａ３.ｐｒｏｇ裡面code：

ｄａｔａ３是用正負數交叉情況（正確為小到大排序）

Ｍ［２０：１］（４個為一個單位）存的值分別為：－３．７．－８．９．－３４。因為下半部跟ｄａｔａ１一樣所以我只貼上半部ｍｏｖｉ值的部分。

0\_100010\_00000\_0000\_0000\_0000\_0000\_0001 //R0 = 1

0\_100010\_00001\_1111\_1111\_1111\_1111\_1101 //R1 = -3

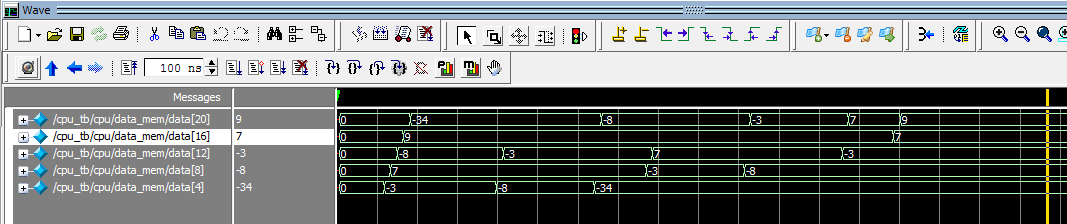
0\_100010\_00010\_0000\_0000\_0000\_0000\_0111 //R2 = 7

0\_100010\_00011\_1111\_1111\_1111\_1111\_1000 //R3 = -8

0\_100010\_00100\_0000\_0000\_0000\_0000\_1001 //R4 = 9

0\_100010\_00101\_1111\_1111\_1111\_1101\_1110 //R5 = -34

波形圖為：



**D. Learned lesson and Conclusion**

ANS：這次主要是加了ｂｒａｎｃｈ跟ｊｕｍｐ的指令，所以主要的困難點是在ｐｃ的選擇，這次因為有上次的硬體做基礎，其實只是在寫回ｐｃ的地方多了是要寫入ｐｃ＋４或者是ａｌｕ＿ｒｅｓｕｌｔ，所以基本上只要接線不要接錯，或是控制指令不要給錯，大致上應該很快就完成了。可是事情往往沒有這麼容易，因為我的設計是給ｐｃ＋１一次讀３２ｂｉｔ的資料，所以在跳的位子上也會有所調整，也就是說我是維持ｐｃ＋１但是我取ｐｃ的值是取［３１：２］這樣就變成ｐｃ＋４。再加上我一開始所設計的硬體是依照ＭＩＰＳ的ＣＰＵ所下去設計，所以就會產生我的ｐｃ永遠指在我現在這個指令的下一個指令，所以就會跟ｐｃ是指在當前指令的設計有差別，這樣會使得我在ｂｒａｎｃｈ會多跳一個指令，所以我在ｂｒａｎｃｈ與ｊｕｍｐ指令尚在位移完畢後會再執行減一的動作，這樣就不會多跳一個指令了。

**2. (30 points) Used AndeSight to run the following C/C++ code and extract the machine code from the tool. Then execute those machine codes in your CPU system and verify the result of your CPU system with AndeSight. If yours is wrong, please modify your CPU system until your result meets the result of AndeSight.**

**a. A summary in the beginning to state what has been done (such as SMILE CPU, synthesis, post-synthesis simulation, additional branch instruction with verification)**

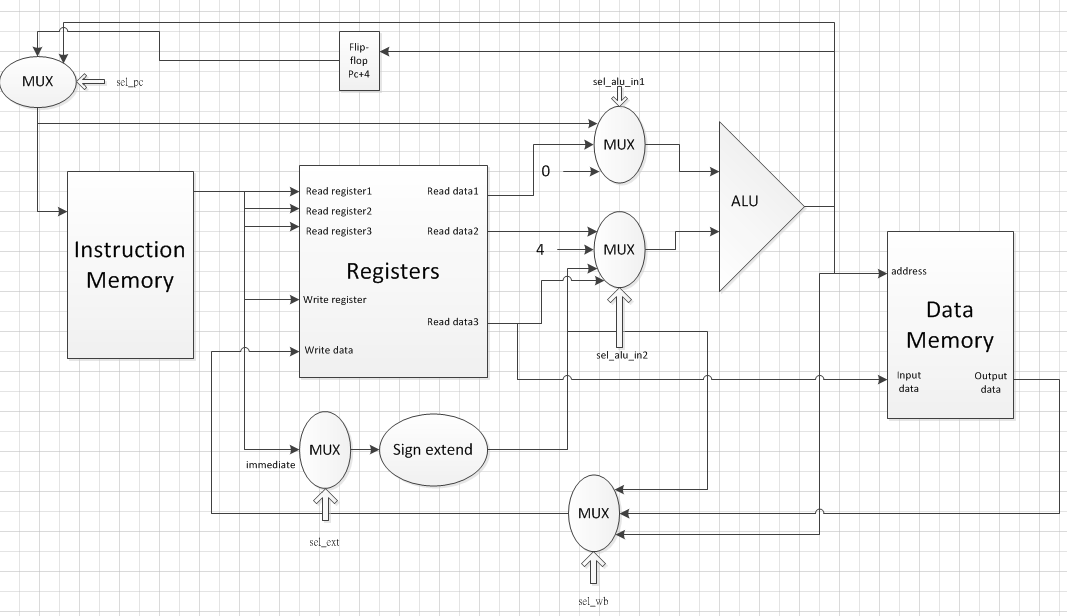
ANS：

這二題跟第一題一樣，除了跟之前一樣做了ｐｒｅｓｉｍ的部分，再加上Ｂｒａｎｃｈ以及Ｊｕｍｐ的指令。也就是說，第二題我是用第一題設計的硬體下去跑。

**b. A block diagram for your completed SMILE CPU indicating all necessary components and I/O pins. Note please use MS Visio that is available in computer center in the university.**

**ANS：**

**這邊得圖形用跟第一題是一樣的架構。**



**C. Simulated waveforms with proper explanation**

**ＴＢ在第二題也是分為兩個部分（ａ）跟（ｂ）。**

**以下是（ａ）的ｃｏｄｅ測試：**

**轉成ｍａｃｈｉｎｅ　ｃｏｄｅ為：**

**//@50045c //ADDI R31=R31-15'd32//R31=-32**

**0101\_0001\_1111\_1111\_1111\_1111\_1110\_0000**

**//@500460 //SWI M[R31+1<<2]=R28 //M[-28]=0**

**0001\_0101\_1100\_1111\_1000\_0000\_0000\_0001**

**//@500464 //ADDI R28=R31+15'd0//R28=-32**

**0101\_0001\_1100\_1111\_1000\_0000\_0000\_0000**

**//@500468 //MOVI R0=20'd170 //R0=170**

**0100\_0100\_0000\_0000\_0000\_0000\_1010\_1010**

**//@50046c //SWI M[R28+2<<2]=R0 //M[-24]=170**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_0010**

**//@500470 //MOVI R0=20'd240 //R0=240**

**0100\_0100\_0000\_0000\_0000\_0000\_1111\_0000**

**//@500474 //SWI M[R28+3<<2]=R0 //M[-20]=240**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_0011**

**//@500478 //MOVI R0=20'd15 //R0=15**

**0100\_0100\_0000\_0000\_0000\_0000\_0000\_1111**

**//@50047c //SWI M[R28+4<<2]=R0 //M[-16]=15**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_0100**

**//@500480 //MOVI R0=20'd85 //R0=85**

**0100\_0100\_0000\_0000\_0000\_0000\_0101\_0101**

**//@500484 //SWI M[R28+5<<2]=R0 //M[-12]=85**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_0101**

**//@500488 //LWI R1=M[R28+2<<2] //R1=M[-24]=170**

**0000\_0100\_0001\_1110\_0000\_0000\_0000\_0010**

**//@50048c //LWI R0=M[R28+3<<2] //R0=M[-20]=240**

**0000\_0100\_0000\_1110\_0000\_0000\_0000\_0011**

**//@500490 //AND R0=R1&R0 //R0=160**

**0100\_0000\_0000\_0000\_1000\_0000\_0000\_0010**

**//@500494 //SWI M[R28+6<<2]=R0 //M[-8]=160**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_0110**

**//@500498 //LWI R1=M[R28+6<<2] //R1=M[-8]=160**

**0000\_0100\_0001\_1110\_0000\_0000\_0000\_0110**

**//@50049c //LWI R0=M[R28+4<<2] //R0=M[-16]=15**

**0000\_0100\_0000\_1110\_0000\_0000\_0000\_0100**

**//@5004a0 //OR R0=R1|R0 //R0=175**

**0100\_0000\_0000\_0000\_1000\_0000\_0000\_0100**

**//@5004a4 //SWI M[R28+6<<2]=R0 //M[-8]=175**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_0110**

**//@5004a8 //LWI R1=M[R28+6<<2] R1=M[-8]=175**

**0000\_0100\_0001\_1110\_0000\_0000\_0000\_0110**

**//@5004ac //LWI R0=M[R28+5<<2] //R0=M[-12]=85**

**0000\_0100\_0000\_1110\_0000\_0000\_0000\_0101**

**//@5004b0 //XOR R0=R1^R0 //R0=250**

**0100\_0000\_0000\_0000\_1000\_0000\_0000\_0011**

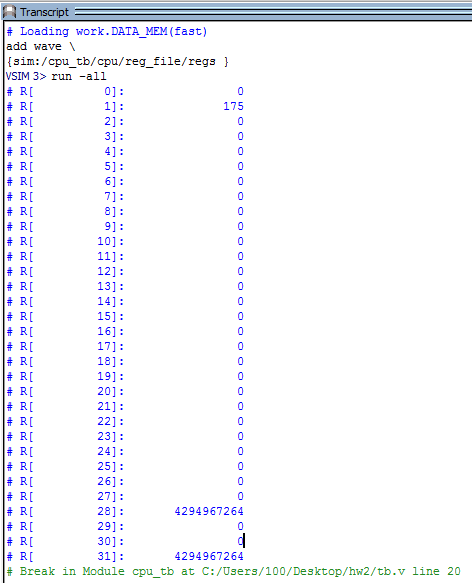
**//@5004b4 //SWI M[R28+6<<2]=R0 M[-8]=250**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_0110**

**//@5004b8 //MOVI R0=0 R0=0**

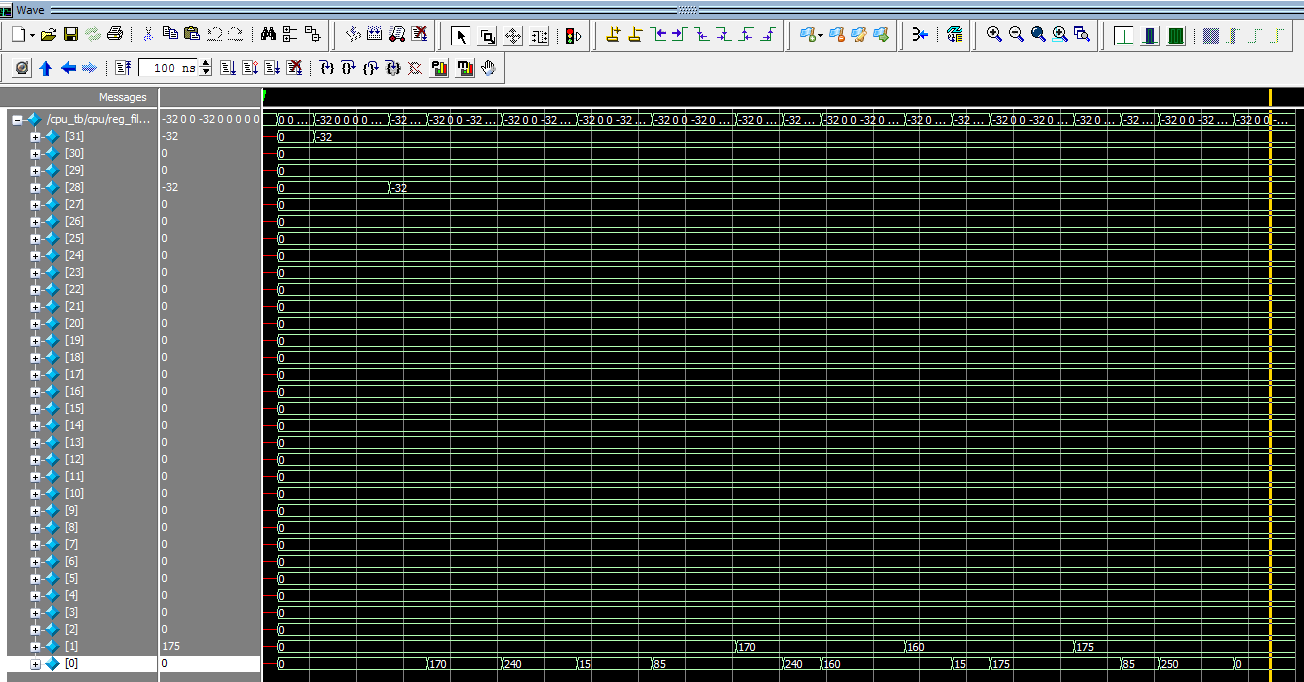
**0100\_0100\_0000\_0000\_0000\_0000\_0000\_0000**

**結果如下：**

****

**其中，我的Ｒ２８跟Ｒ３１的值都為－３２，將－３２換成無號數後，就變為４２９４９６７２６４，因為我還不知道ｄｉｓｐｌａｙ可不可以顯示有號數，所以在這邊ｄｉｓｐｌａｙ出來的值為無號數。**

**波形圖為：**



**由波形圖可以很清楚的看到每個暫存器裡面的值如何變化。**

**以下是（ｂ）的ｃｏｄｅ測試：**

**轉成ｍａｃｈｉｎｅ　ｃｏｄｅ為：**

**//@50045c**

**0\_101000\_11111\_11111\_11111\_11110\_11000 //R31 = R31 + (-40) = -40**

**//@500460**

**0\_001010\_11100\_11111\_00000\_00000\_00001 //M[R31+4] = R28 => M[-36] = 0**

**//@500464**

**0\_101000\_11100\_11111\_00000\_00000\_00000 //R28 = R31 + 0 = -40**

**//@500468**

**0\_100010\_00000\_00000\_00000\_00110\_01000 //R0 = 200**

**//@50046c**

**0\_001010\_00000\_11100\_00000\_00000\_00010 //M[R28+8] = R0 =>M[-32] = 200**

**//@500470**

**0\_100010\_00000\_00000\_00000\_00101\_11000 //R0 = 184**

**//@500474**

**0\_001010\_00000\_11100\_00000\_00000\_00011 //M[R28+12] = R0 =>M[-28] = 184**

**//@500478**

**0\_100010\_00000\_00000\_00000\_01001\_10111 //R0 = 311**

**//@50047c**

**0\_001010\_00000\_11100\_00000\_00000\_00100 //M[R28+16] = R0 =>M[-24] =311**

**//@500480**

**0\_000010\_00000\_11100\_00000\_00000\_00010 //R0 = M[R28+8] = M[-32] = 200**

**//@500484**

**0\_100000\_00000\_00000\_00101\_00000\_01001 //R0 = R0>>5 = 6**

**//@500488**

**0\_001010\_00000\_11100\_00000\_00000\_00101 //M[R28+20] = R0 =>M[-20] = 6**

**//@50048c**

**0\_000010\_00000\_11100\_00000\_00000\_00011 //R0 = M[R28+12] = M[-28] =184**

**//@500490**

**0\_100000\_00000\_00000\_00011\_00000\_01000 //R0 = R0<<3 = 1472**

**//@500494**

**0\_001010\_00000\_11100\_00000\_00000\_00110 //M[R28+24] =R0 =>M[-16] = 1472**

**//@500498**

**0\_100010\_00000\_00000\_00000\_00000\_00000 //R0 = 0**

**//@50049c**

**0\_001010\_00000\_11100\_00000\_00000\_00111 //M[R28+28] = R0 =>M[-12] = 0**

**//@5004a0**

**0\_000010\_00001\_11100\_00000\_00000\_00101 //R1 = M[R28+20] = M[-20] = 6**

**//@5004a4**

**0\_100010\_00000\_00000\_00000\_00000\_00101 //R0 = 5**

**//@5004a8**

**0\_100110\_00001\_00000\_00000\_00000\_11000 //if()**

**//@5004ac**

**0000\_0100\_0000\_1110\_0000\_0000\_0000\_0111**

**//@5004b0**

**0000\_0100\_0001\_1110\_0000\_0000\_0000\_0110**

**//@5004b4**

**0100\_0000\_0000\_0000\_0000\_0100\_0000\_0001**

**//@5004b8**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_0110**

**//@5004bc**

**0100\_1000\_0000\_0000\_0000\_0000\_0000\_1110**

**//@5004c0**

**0000\_0100\_0000\_1110\_0000\_0000\_0000\_0111**

**//@5004c4**

**0101\_0000\_0000\_0000\_0111\_1111\_1111\_1111**

**//@5004c8**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_0111**

**//@5004cc**

**0100\_0100\_0000\_0000\_0000\_0000\_0000\_0000**

**//@5004d0**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_1000**

**//@5004d4**

**0100\_1000\_0000\_0000\_0000\_0000\_0001\_0100**

**//@5004d8**

**0000\_0100\_0001\_1110\_0000\_0000\_0000\_0101**

**//@5004dc**

**0000\_0100\_0000\_1110\_0000\_0000\_0000\_0110**

**//@5004e0**

**0100\_0000\_0000\_0000\_1000\_0000\_0000\_0011**

**//@5004e4**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_0111**

**//@5004e8**

**0000\_0100\_0001\_1110\_0000\_0000\_0000\_0111**

**//@5004ec**

**0000\_0100\_0000\_1110\_0000\_0000\_0000\_0100**

**//@5004f0**

**0100\_0000\_0000\_0000\_1000\_0000\_0000\_0100**

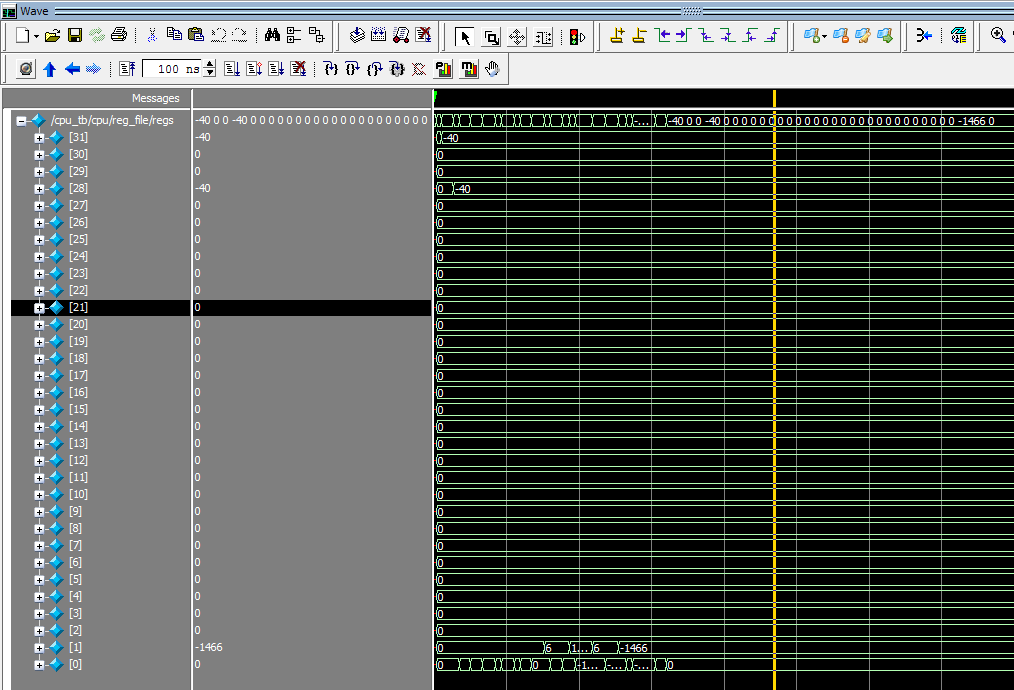
**//@5004f4**

**0001\_0100\_0000\_1110\_0000\_0000\_0000\_0111**

**//@5004f8**

**0100\_1000\_1111\_1111\_1111\_1111\_1110\_0100**

**波形圖為：**



**D.Learned lesson and Conclusion：**

作業二的部分比較簡單一點，就是試著把ｃｏｄｅ轉成ｍａｃｈｉｎｅ　ｃｏｄｅ讓ＣＰＵ下去跑，其實感覺做到這裡已經可以很接近ｐｉｐｅｌｉｎｅ的ｃｐｕ了，雖然歷經了很多問題，以及每天都要花好幾個小時甚至熬夜來寫ｃｏｄｅ或是ｄｅｂｕｇ，隨著成果一一實現，會發現其實學到的地方真的很多。