**VLSISystemDesign**

**Homework-2**

**Part1**

**學號：N26991784**

**姓名：林宏達**

**1. (20 points) Add in a 32x32bit instruction memory into the CPU system that designed in HWI(part 2) exercise 4. Please load the following machine code into the instruction memory and modified the controller so that the CPU system could read the sequence of the machine codes from the instruction memory and implement the corresponding operations. The machine code is as below:**

**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)**

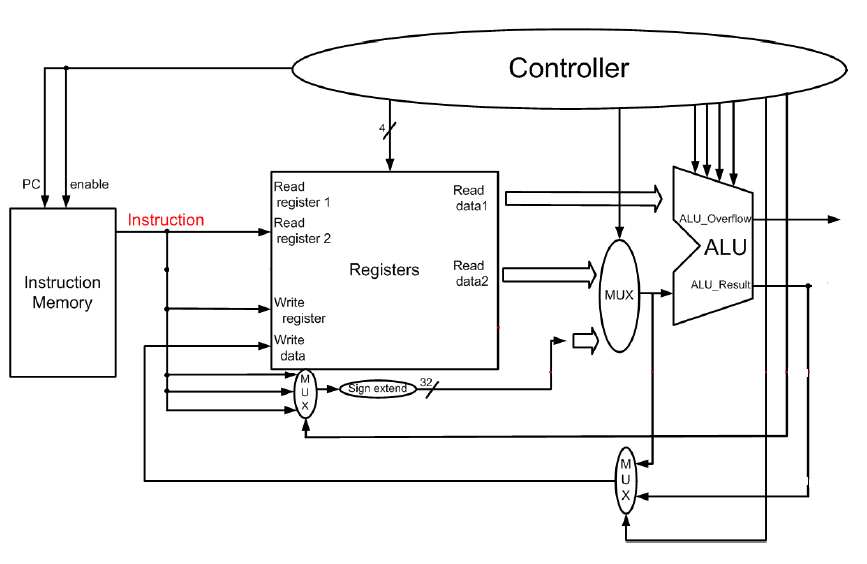
ANS：

這次的作業跟之前一樣，我做了ｐｒｅｓｉｍ的部分。

**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：

這次因為觀念上跟上次做得有很大的不同，我把MUX抽掉，改在CONTROL裡面去做MUX的功能，只是在CODE的寫法上有很大的改變，但是在圖上並沒有很大的改變。因為MS的VISIO是試用版的，所以也到期了，所以我重灌電腦後是WIN-64BIT，就不能安裝MS的VISIO32BIT試用版，所以這次的圖片就先用小畫家配合助教附的圖修改代替，等下次找到可以用的VISIO在使用VISIO。因為作業一跟二的圖只差DATA\_MEMORY以及REGISTER幾個接腳，加上設計方式是一樣的，所以這邊我就附上作業二的圖片修改當作說明。



這次也把ALU重新設計過一次，ALU只做很單純的運算，所以也把ADD跟ADDI之類的運算都做了統整，等於就是說我所有的MODULE只做單純的存資料、送資料以及運算，所有的控制選擇以及判斷都是由CONTROL去決定，所以CODE在CONTROL上會寫的比較複雜，簡單的說就只是把MUX跟OPCODE的判斷都丟到CONTROL去寫。

**C. Simulated waveforms with proper explanation**

ANS：

我的ＴＢ的數值前半部是用說明附贈上的，後半部是用自己多加的來做ＬＷ跟ＳＷ，所以會在Ｒ６－Ｒ８暫存器的波形會有錯，這是沒有關係的，因為作業一本來就沒有ＤＡＴＡ＿ＭＥＭＯＲＹ，所以只需要看Ｒ０－Ｒ２暫存器的值對不對即可。

0\_100010\_00000\_00000000000000000100 //MOVI R0=20'd4

0\_101000\_00000\_00000\_000000000001101 //ADDI R0=R0+4'b1101

0\_101100\_00001\_00000\_000000000000010 //ORI R1=R1|4'b0010

0\_101011\_00001\_00001\_000000000000111 //XORI R1=R1^4'b0111

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

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

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

0\_100000\_00010\_00000\_00001\_00000\_00011 //XOR R2=R0^R1

0\_100000\_00010\_00000\_00011\_00000\_01001 //SRLI R2=R0 SRL(3)

0\_100000\_00010\_00010\_00011\_00000\_01000 //SLLI R2=R2 SLL(3)

0\_100000\_00010\_00000\_11101\_00000\_01011 //ROTRI R2=R0 ROTR(29)

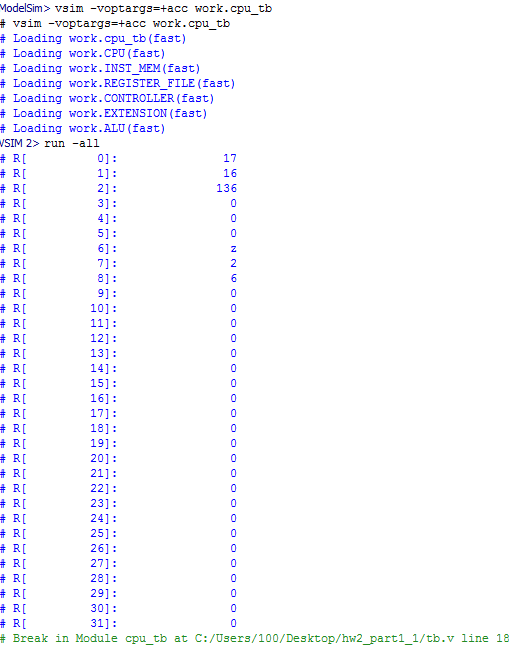
0\_100010\_00111\_00000000000000000010 //MOVI R7=20'd2

0\_100010\_01000\_00000000000000000110 //MOVI R8=20'd6

0\_011100\_00000\_01000\_00111\_00\_00001010 //M8=R0

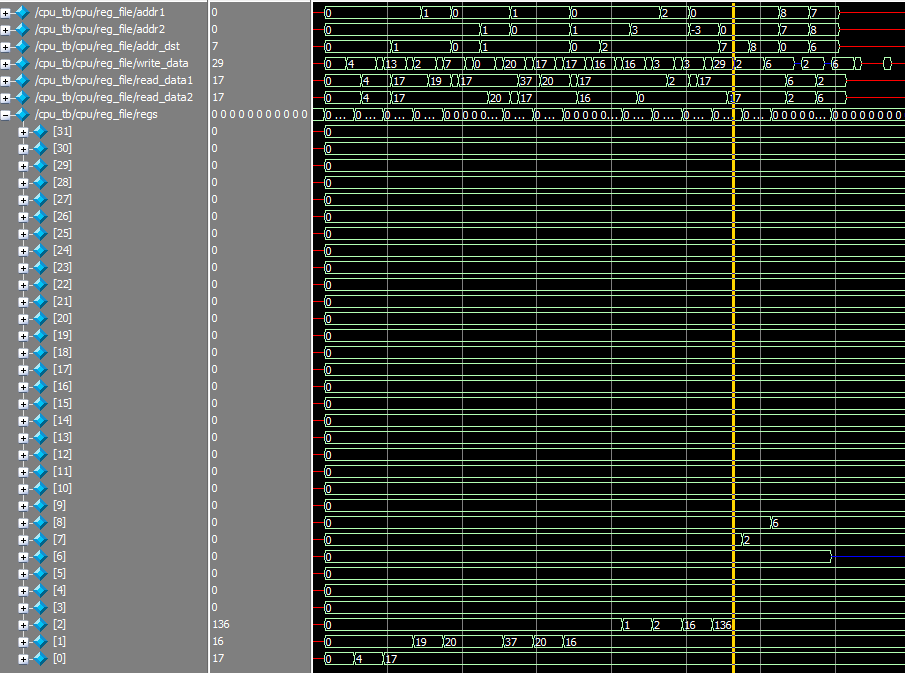
0\_011100\_00110\_00111\_01000\_00\_00000010 //R6=M8

因為我有寫ｍｏｎｉｔｅｒ所以有把每個暫存器的值印出來如下圖：



Ｒ０＝０、Ｒ１＝１６、Ｒ２＝１３６。

而波形圖如下：



**D. Learned lesson and Conclusion**

ANS：這個地方因為我是從第２題先做，之後再做回第１題，所以學到甚麼以及心得我在的２題做說明。

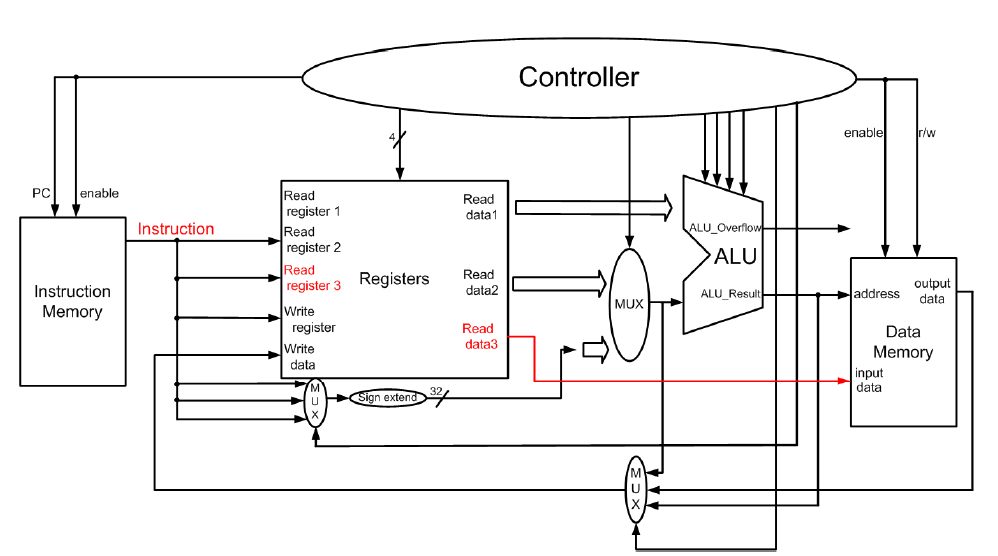
**2. (20 points) Add in a 128x8bit byte addressing data memory into the CPU system. And modify your register file become three read ports as show in Fig.3. Please load the mins.prog(you have to convert it to machine code first by yourself) in the next page into the instruction memory and modified the CPU system so that it could execute the 17 instructions as show in the table at the beginning.**

**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)**

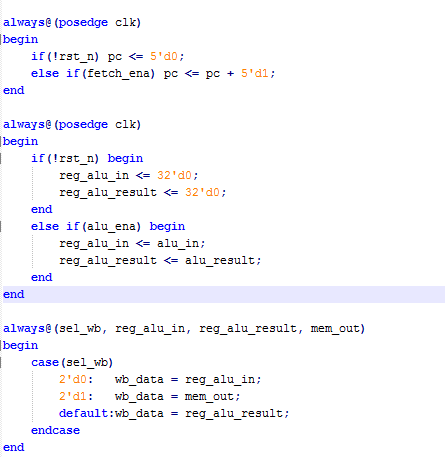
ANS：

這一題也是只有做了ｐｒｅｓｉｍ的部分。

**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：**

大致上設計還是依照這張圖下去寫ｖｅｒｉｌｏｇ　ｃｏｄｅ，只是在寫法上有些許的不同，例如上次作業的部分，因為在ｗｒｉｔｅｂａｃｋ時我的ｅｎａｂｌｅｅｘｅｃｕｔｅ被拉為０，所以導致我的ａｌｕｒｅｓｕｌｔ被設定為０，所以才會在ｗｒｉｔｅｂａｃｋ前面的線，多加了一顆ｂｕｆｆｅｒ。這次則是用了ｆｌｉｐ－ｆｌｏｐ的寫法把值鎖住，例如：



我用ａｌｕ＿ｅｎａ去驅動ｆｌｉｐ－ｆｌｏｐ讓值傳到一個暫存器之後再由ｓｅｌ＿ｗｂ等等去選擇我要寫回去的值是哪個。

**C. Simulated waveforms with proper explanation**

**ＴＢ為：**

**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**

**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)**

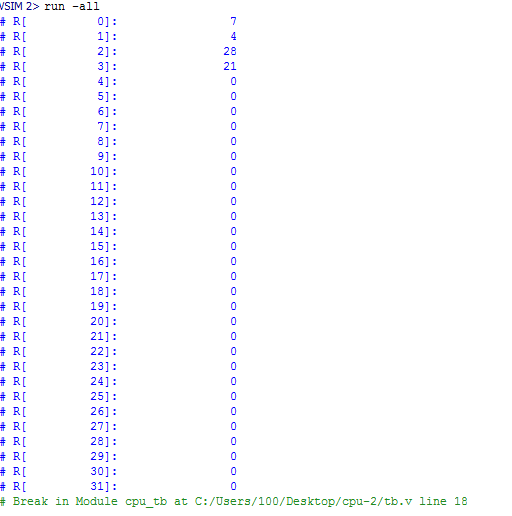
**0\_011100\_00010\_\_00011\_00010\_01\_00001010 //SW M56=R2**

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

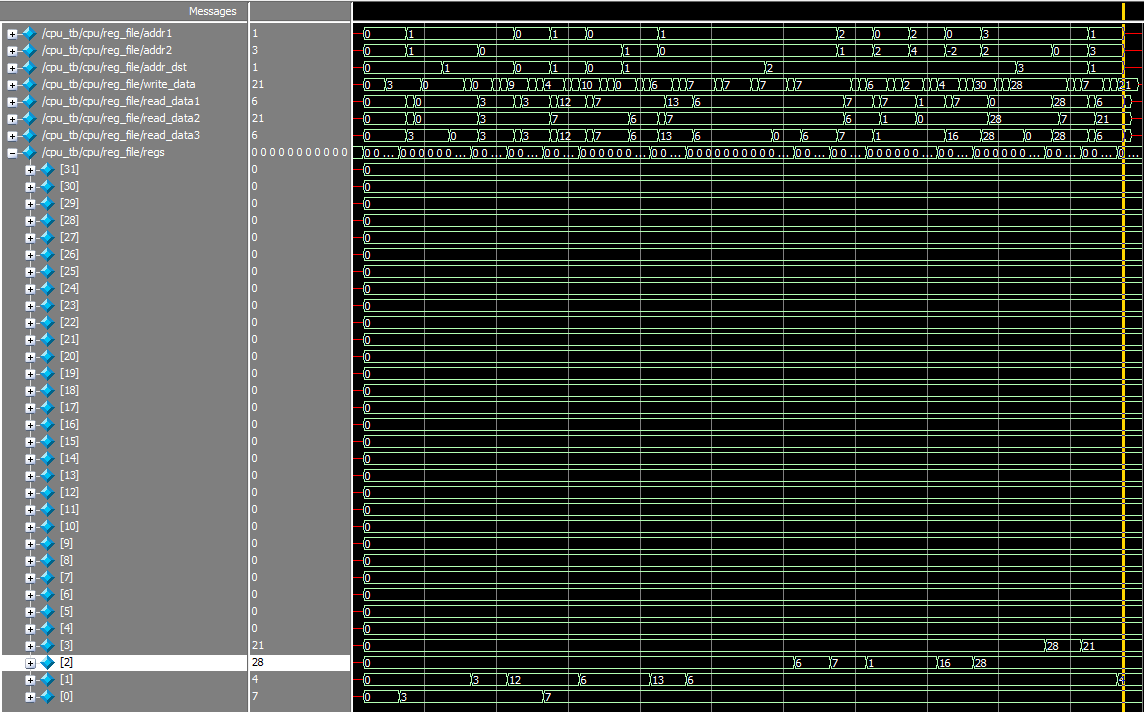
**0\_100000\_00011\_00011\_00000\_\_00000\_\_00001 //SUB R3=R3-R0**

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

**結果如下：**

****

**波形圖為：**

****

**D.Learned lesson and Conclusion：**

這次最大的困難點應該就是把CONTROL跟ＭＵＸ合再一起了，前前後後遇到的問題有很多，接腳接錯地方還是忘了接等等。這次作業是只要讀取檔案裡面的ＩＮＳＴＲＵＴＩＯＮ，讓我更加了解電路整個再對指令做處裡的詳細流程，這次的作業也讓我嘗試到不一樣的寫法，線變少了，但是控制訊號變多了，不過看起來更簡潔了。其實像這次的作業，建議可以指做作業2就好，因為我是先做完作業2才回頭去做作業1，感覺是一樣的東西，還要特地把作業2退化成作業1，只為了繳交作業1。