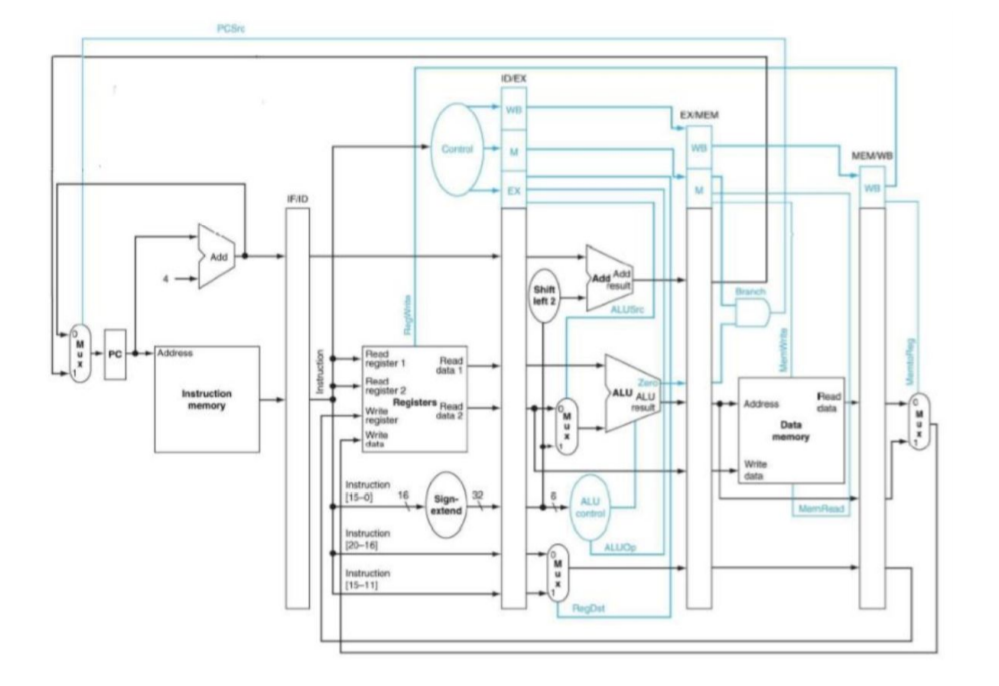
**Computer Organization Lab4**

**Architecture diagrams: 直接用附圖**

****

**Hardware module analysis:** ModelSim

**Adder.v** : 實作加法就好，將兩個數值相加

**ALU.v** : 實作ALU，可以直接參考PDF的Appendix，基本的運算。

**ALU\_Ctrl.v** : 決定在ALU.v 的operation，用switch…case…實作

**Decorder.v** : 將instruction轉變成實作用的code，以及對應的ALU\_Ctrl，用switch…case…實作

**MUX\_2to1.v** : 兩個可能的多功器，套if…else or .. ? .. : .. 即可

**Shift\_Left\_Two\_32.v** : 將所有位數左移2即可，利用”<< ”

**Sign\_Extend.v** : 重複16次MSB 再concatenate input value

**Pipe \_CPU\_1.v** : 將之前的CPU改成Pipe版，只是要加上IF,ID等區域，還是一樣將小程式組裝起來即可。

**Instruction Memory. v** : 紀錄所有 Instruction 資訊的記憶體，只在 Positive Clock Edge 時才可輸出值的改寫。

**Pipe\_Reg. v** : 將Reg變成Pipe版

**Reg File. v** : 將值存進暫存器

**Program Counter(PC). v** : 計數器，只在 Positive Clock Edge 時才

可輸出值的改寫，決定要取得的 Instruction Memory 的 Address。

**Data Memory. v** : 模擬外部記憶體。需要允許寫入的訊號才可以在

Positive Clock Edge時將想寫入的數值寫到指定的記憶體位址。一樣

也需要允許讀取的訊號才能讀取。只要允許讀取，便會輸出記憶體位

置所儲存的值。

**TestBench. v** :設計實驗進行模擬。

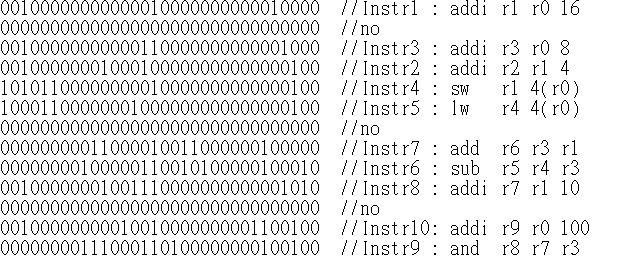
**Decorder:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Op** | **ALUOp** | **ALUSrc** | **RegWrite** | **RegDst** | **Branch** | **MemRead** | **MemWrite** | **MemtoReg** |
| **R** | **000** | **0** | **1** | **01** | **0** | **0** | **0** | **0** |
| **ADDI** | **001** | **1** | **1** | **00** | **0** | **0** | **0** | **0** |
| **SLTi** | **010** | **1** | **1** | **00** | **0** | **0** | **0** | **0** |
| **BEQ** | **011** | **0** | **0** | **00** | **1** | **0** | **0** | **0** |
| **LW** | **100** | **1** | **1** | **00** | **0** | **1** | **0** | **1** |
| **SW** | **100** | **1** | **0** | **00** | **0** | **0** | **1** | **0** |
| **jump** | **100** | **0** | **0** | **00** | **0** | **0** | **0** | **0** |

**ALU\_Ctrl:**

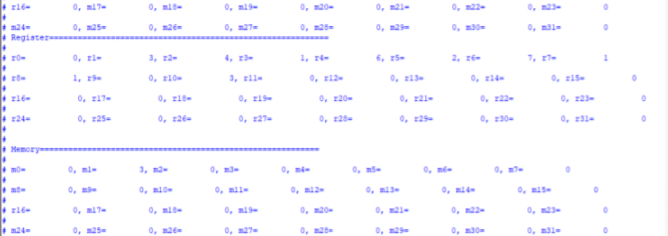
|  |  |  |  |
| --- | --- | --- | --- |
| **Op** | **ALUOp** | **function** | **ALUCtrl(‘b)** |
| **ADD** | **0** | **32** | **0010** |
| **SUB** | **0** | **34** | **0110** |
| **AND** | **0** | **36** | **0000** |
| **OR** | **0** | **37** | **0001** |
| **SLT** | **0** | **42** | **0111** |
| **JR** | **0** | **8** | **0000** |
| **MULT** | **0** | **24** | **1111** |
| **ADDi** | **1** | **X** | **0010** |
| **SLTi** | **2** | **X** | **0111** |
| **BEQ** | **3** | **X** | **0110** |
| **LW, SW** | **4** | **X** | **0010** |
| **ANDi** | **5** | **X** | **0000** |

**Bonus**

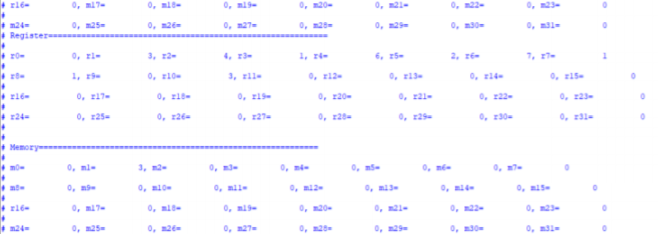


Result :

CO\_P4\_test\_1\_result : 與 result1 reg、memory相符

****

CO\_P4\_test\_2\_result : 與 result2 reg、memory相符

****

**Problems you met and solutions:**

Decorder.v的部分從lab2去更改，新增了一些參數，也順便更動ALU.v跟ALU\_C.v，而且畢竟它是CPU的核心部分，解碼錯會導致它全盤錯掉，所以要好好賦值。一個值的影響很大，只好不斷檢驗。

同Lab2，最困難的依然是把所有的小程式併在一起，也就是CPU.v，而且這次新增了新區塊，還要弄成Pipe版本，相當燒腦費神，還是一樣一個沒弄好，輸出都有問題，依然要花很多時間思考研究和整理每條電路的連接，然後宣告跟填上相對應的參數，常常想錯參數就會擺錯位置，就導致輸出有問題，完全弄懂CPU後參數才放對，也才能看到輸出，命名也很重要，因為這樣才比較不會亂掉。這次更重要的是要弄對reg的大小，真的很容易數錯。還有result要去了解Bonus，不然會一直覺得不合。

**Summary:**

這次依然要模擬外部記憶體。一樣是只要把每個小部分完成，project就大致做完了，不過不知道為何這次很容易當機，對每個細節必須更用心。先弄懂CPU的實作後，比較困難的只有接線填參數(Pipe\_CPU\_1.v)和解碼(Decorder.v)，還有要記得讀寫檔的路徑問題。

在Lab中，因為有Pipe的加入，讓程式變得更複雜，多了一些區塊和必須為此作考量，但也更貼近完善的CPU，我更加理解CPU，也更了解CPU的實作方法，越來越像是一台電腦的核心，能用軟體模擬硬體，真的提供不小的便利性。