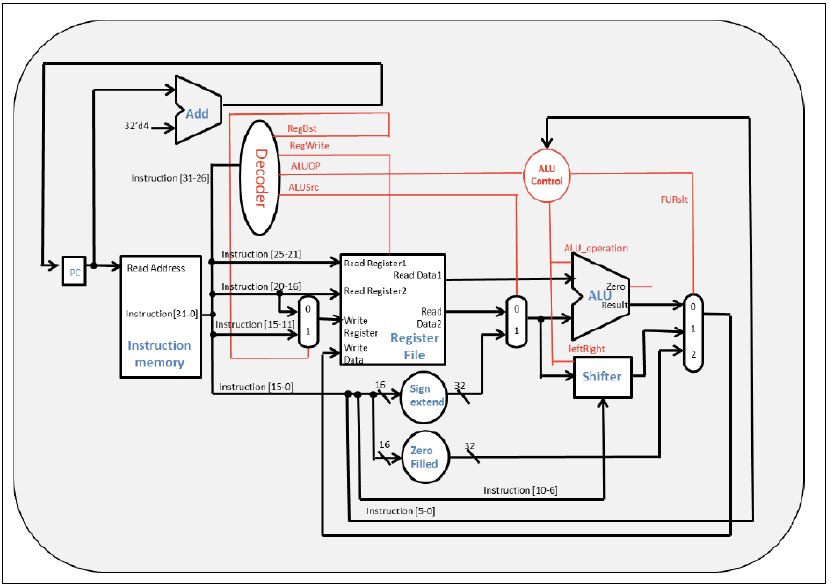
**Computer Organization**

**Architecture diagrams:**

因為完全照著spec中的架構圖來實作所有module與接線，所以這裡直接放上spec裡的Architecture Diagram

****

**Hardware module analysis:**

**Program Counter(助教提供)：**

* PC是一個32bits的register，會隨著clock變化而不斷上升，每個clock會+4，並且會用rst signal來reset

**Instruction memory(助教提供)：**

* Instruction memory是32x32-bits memory(32 words). 指令會依照PC給出的PC address 來讀取與輸出

Adder：

* Input1+input2=output，這裡主要是用在計算PC+4

Decoder：

* 用來decode每個指令，依照spec的表格輸出正確的control signals與ALUOp

Sign Extend：

* 把16 bits的singal用第16個bit延展(第17~32個bit都設成跟第16個bit一樣)成32 bits

Zero Filled：

* 這次lab似乎沒有會使用到這個module的測資，主要是用在lui instruction，會把input的16 bits signal放在upper的16 bits，lower的16 bits都用0填滿，最後輸出32 bits

3/2to1 Mux：

* 輸入數個data，利用輸入的select\_i來決定要輸出哪一個input

Register File(助教提供)：

* Outputs (RS and RT) is obtained by using the addresses input into the file.
* If RegWrite is set (RegWrite == 1), the input data is assigned to the register specify by RDaddr

ALU:

* 利用spec中最後一頁的範例照著寫出一個32 bits的ALU，用來處理R-type與address計算

ALUCtrl：

* 輸入decoder輸出的ALUOp與指令的function field，用來確定ALU需要做甚麼operation
* 輸出的FURslt是用在最後的3to1 Mux，決定最後要輸出哪個data
* 為了bonus的2個指令，多加了一個output signal (ShiferV\_o)

Shifter(助教提供)：

* 由於不確定上次lab的shifter有沒有寫對，所以使用助教提供的shifter，輸入往左或往右、shamt與需要shifter的32bits，輸出shifter後的result

Simple Single CPU：

* 把所有的module全部連接起來，組成一個single cycle的CPU

**Finished part:**

所有module都完成了，包括bonus的部分，完成在ALU\_Ctrl與Decoder裡面，並且在Simple\_Single\_CPU中額外增加了一個2to1的Mux。

**Problems you met and solutions:**

問題1：在simple single cpu的module中，把PC線接錯，導致PC address不斷出錯

Solutioin：想到可以利用GTKWave看波型觀察哪裡有線接錯，後來發現是自己PC+4的output沒有接回pc\_in\_i，更正後PC address就沒有出錯了

問題2：因為助教給的input、output大多都是wire，但wire不能在always中assign，不知道怎麼處理

Solution：原本想把wire改成reg，但不知道會不會出現問題，所以把需要用到if-else的地方全部改成？：的三元運算子來處理，就不需要用到always了

**Summary:**

我覺得這個lab讓我對於上課Ch4提到的single cycle cpu有更深入的了解，並且知道每個module與指令在實作時要如何處理，雖然過程中遇到一些問題，但是靠著同學與助教們的幫忙，還是成功寫出了這個lab。