**Computer Organization**

**Architecture diagrams:**

****

**Finished part:**

**完成本次作業的基本要求。**

**Hardware module analysis:**

**最左邊的PC input從0開始，經過PC輸出再往上進到Add加4輸出下一個PC address，往右進去Instruction memory取得指令輸出。接著分別將Instruction[31:26]輸入進上方的控制單元Decoder，而控制單元根據指令的形式輸出不同的訊號給其它單元。將指令輸出的Instruction[25:21]輸入進暫存器Register File的Read Register 1、Instruction[20:16]輸入進暫存器Register File的Read Register 2。而Instruction[20:16]和Instruction[15:11]輸入多工器，由控制單元的RegDst輸出決定哪一個輸入暫存器Register File的Write Reg 1。 Instruction[15:0]輸入給sign extend轉換成32bits輸出。最後是將Instruction[5:0]輸入給ALU control，再根據控制單元的ALUOp訊號和輸入的function code輸出不同的ALUCtrl。**

**中間的Register根據輸入的Read Register 1和Read Register 2去讀取相對應的值，分別輸出Read Data 1和Read Data 2，其中Read Data 2會和Sign Extend後的結果進去多工器，由控制單元輸出的ALUSrc訊號決定誰作為ALU\_i和Read Data 1進去ALU進行運算。ALU依據ALUCtrl判斷該用哪些輸入執行哪些運算，運算結果輸出給暫存器，暫存器再依據控制單元的RegWrite訊號決定要不要寫入。另外，ALU還有根據Read Data 1和ALU\_i是否相同輸出訊號，正常情況相同輸出1，不同輸出0，如果是Branch on not equal指令時則相反。**

**Problems you met and solutions:**

**在完成程式的過程中，首先遇上的是vivado無法模擬編輯的問題，在嘗試好幾個小時之後只能放棄解決這問題，轉而使用modelsim來模擬這次的線路圖。不得不說modelsim真的是一項很難用的程式，有時候會有程式閃退或者是模擬到一辦當掉的情況。強烈建議助教之後推薦給我們使用其他種軟體來進行模擬。**

**接下來線路的部分，每個元件的線路之間只需照著指示將每個module串起來就好，當output是wire的時候再串接一個register給他就好。**

**這次在alu的部分一開始slt以及sll、srl的部分老是出錯，程式執行到這裡時就會顯示錯誤，根據線路追蹤回去發現是alusrc1和alusrc2的部分沒有改成signed值，因此比較負值時就會出錯，答案會誤判為正。**

**再來是最後一個多工器，在選擇時一開始沒注意，將flrsult值預測錯誤，所以在lui指令輸出的結果不小心從alu的輸出端輸出。再發現後我就在alu control的部分進行判斷，當op field是110000後只要把flrsult設為1就好。這樣就可以將正確的答案寫進rf裡。另外，hardware module analysis的部分不清楚要打多少，因此以上先將執行的過程大概敘述過。**

**Summary:**

**本次實作在目前為止，在程式碼的部分總共花了快15小時，其中有一半的時間都在應付modelsim這個惱人的軟體以及懷疑人生，真的在演算法判斷這部分基本上沒有很多難度，要擔心的就是每個指令在輸入和輸出端要確實接對以及signed和unsigned的判斷。**

**不過這次實作練習在cpu的簡單流程以及線路圖的部分，也因為本次的練習有深刻的理解。**