Hw4 Report

B05504066 李旻翰

在各module裡面有條件控制的部分，我都以always @(\*)裡面寫if，else處理，代表只要有變數(block變數跟PC)變化就要進行block裡面的事情。

1. Control：此部分主要在控制register和ALU的行為。我們先輸入instruction[6:0]為Op\_i，然後在always block裡面進行decode。因為只有R-type跟I-type，我直接使用if else區隔開來：
2. 如果是R-type (Op\_i == 51)，ALUop設為10，ALUSrc設為0。
3. 如果是I-type (Op\_i == 19)，ALUop設為00，ALUSrc設為1。

RegWrite\_o 都是1。

Decode完會送到Register，ALU Control跟MUX給那些module做判別。

1. ALU：這部分主要運作數學運算，input兩個運算元，並且藉由ALUCtrl\_i決定進行何種operation。分為以下幾種：
2. and: 做”&” operation。
3. xor: 做”^” operation。
4. sll: 做”<<”operation。
5. add，addi: 做”+”operation。
6. sub: 做”-”operation。
7. mul: 做”\*”operation。
8. sra: 先換成signed之後，右移data2\_i個。

Zero\_0為0。

算完之後就寫回destination register，若是零就output 0。

1. ALU\_Control：這部份決定進行哪種操作，會先讀ALUOp\_i決定他的type，再來看funct\_i，來看他的實際運是甚麼。因為I-type的funct\_i只有[2:0]有效，所以在比較時只看這三碼，其他instruction都看全部。
2. And = 1
3. Xor = 2
4. Sll = 3
5. Add = 4
6. Sub =5
7. Mul = 6
8. Addi = 4，這跟add一樣，差別在data2是甚麼，由MUX32控制。
9. Srai = 7

Decode完送到ALU去執行相對應的operation。

1. Adder：負責給PC加4，因為這份作業沒有branch condition，所以就一直加就好。加完之後送回PC。
2. Sign\_Extend：只有在I-type有用。將原本12碼的immediate加長成32碼，做法就是先判斷最高位的bit，如果是1就放20個1在前面；反之放20個0。寫完之後送進MUX然後進ALU進行操作。
3. MUX32：從Controller拿到select碼之後，若為1(R-type)就讓reg2讀來的data通過；如果是0(I-type)就把sign\_extend的通過。
4. CPU：由於腳位，reg都在各module寫好，我們只要把各個module的input，output接好即可，也不需要額外的register跟wire。寫的方式為讓module的input承接其他module的output，而output不需要放變數，基本上就照圖片所指示的接線。