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：
   (a) 如果是 R-type (Op_i == 51)，ALUop 設為 10，ALUSrc 設為 0。
   (b) 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。
2. 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：
   (a) and: 做”&” operation。
   (b) xor: 做”^” operation。
   (c) sll: 做”<<”operation。
   (d) add，addi: 做”+”operation。
   (e) sub: 做”-”operation。
   (f) mul: 做”*”operation。
   (g) 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。
3. 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。
   (a) And = 1
   (b) Xor = 2
   (c) Sll = 3
   (d) Add = 4
   (e) Sub =5
   (f) Mul = 6
   (g) Addi = 4，這跟 add 一樣，差別在 data2 是甚麼，由 MUX32 控制。
   (h) Srai = 7
   Decode 完送到 ALU 去執行相對應的 operation。
4. 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。

5. 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。
6. 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。
7. 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。