R10922171 陳嘉政 HW4

- ## Development environment

1. 使用 vs code 進行 programming
2. 使用 windows powershell 進行 compiling:

   iverilog -o xxx.vvp xxx.v
   vvp xxx.vvp
3. 最後透過 gtkwave 軟體輸出波形檔(.vcd)

   gtkwave xxx.vcd

- ## Module implementation explanation

1. Adder :

有兩個 input 及一個 output。Input 1 為從 PC 計算出來的 32-bit instruction address。Input 2 為固定的 32-bit 常數 4。每輪都會將兩個輸入相加 assign 給 32-bit 的 output wire，當作下一輪的 PC 輸入。

2. Control :

一個 input 及三個 outputs。輸入為從 instruction memory 讀出來的 32-bit 指令中[6:0]bit 的 opcode，經由 control 判斷此指令為何，再發出三個 output 來控制各個元件，分別為 2-bit 的 ALUop 來控制 ALU control，1-bit 的 ALUSrc 來控制 MUX32，1-bit 的 RegWrite 來控制 Registerfile，因為範例指令都會寫回 register file 所以永遠設 1。

3. Sign-extend :

只有一個輸入及一個輸出，輸入為指令中的[31:20]bit 的常數，經過 sign extension 擴充成 32-bit 的常數再輸出。

4. **MUX32 :**

   是一個 2 對 1 的選擇器，輸入分別為從 register file 讀出的 rs2 data 及 sign-extend 輸出的常數，再藉由輸入的控制信號 select_i，0 選 rs2 data，1 則選常數通過。

5. **ALU Control :**

   輸入為將指令 funct7[31:25]及 funct3[14:12]接合的 funct，以及從 control 傳來的 2-bit ALUop，00 表示 I type 指令，01 表示 beq 指令，而 10 表示 R type 指令，經過這些 bit 來判斷出 3-bit ALUCtrl 來決定 ALU 要做甚麼 operation。

6. **ALU :**

   輸入為從 register file 傳來的 32-bit rs1 data 及從 MUX32 選出來的 32-bit data，以及 3-bit ALUCtrl，透過 ALUCtrl 的值來決定 ALU operation，000 做 and，001 做 or，010 做 add，110 做 sub，011 做 mul，100 做 xor，101 做 sll，111 做 srai。輸出有 32-bit data 及 1-bit zero，data 為前面計算產生的結果，將他傳回 register file 的 RDdata_i 中，而 zero 是用來比較兩個輸入的值是否相同，此處我們沒用到所以設為 0。