**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

1. 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。

1. **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。

1. **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。

1. **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。

1. **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。

1. **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。