HW3 Report

資工四 胡材溢 b08208032

- Development environment

這次的作業我使用的作業系統為 macOS，在 Visual Studio Code 上寫這次作業的程式，並下載了 Verilog-HDL/SystemVerilog/Bluespec SystemVerilog 延伸套件作為輔助。編譯的部分使用 iverilog，並藉由 iverilog -o CPU.out *.v 指令來編譯，再使用 vvp CPU.out 執行程式，應該沒什麼大問題。

- Module implementation explanation

這次的作業我實作了以下模組，都沒有使用 register 型態的變數，大多都是 wire，用 assign 來傳遞值。

1. Control：

讀 instruction 中的 opcodes 並輸出 control signals 來控制 ALU 以及 registers。因為讀入的 opcodes 只有兩種形式，故以讀入 7’b0010011 設定 ALUOp 為 2b’00 反之設定為 2b’01，spec 中的設計圖顯示 ALUOp 應該放兩個 bit，但實際上只用了一個。

opcodes 為 7’b0110011 的 instruction 都需要 sr2，而 opcodes 為 7’b0010011 的 instruction，因此 ALUSrc 設定方式同上。

因為這次所有 instructions 都要將值寫回 register，因此 RegWrite 恆為 1。

2. ALU control：

先根據 ALUOp，若 ALUOp 為 2’b01，代表為 R-type 的 instructions。再由 funct7 的 bits，判斷現在要執行哪個 instruction。若 ALUOp 為 2’b00，代表剩下兩個 instructions，同樣藉由 funct7 的 bits，判斷要執行哪一個 instruction。將 SLL、ADD、SUB、MUL、ADDI、SRAI 分別編號為 0~7，作為輸出。

3. Sign extend：

用 concatenate 的方式，將 data 的第一個 bit 延長，再接上原本的 data 作為輸出。

4. ALU：

根據 ALUCtrl 的數值，以及 2.提到的編號方式，執行相應的 operation。特別的是 srai，在取 imm 的值時，只需取 4~0 的 bits 即可。另外 zero 沒有用到。

5. Adder：

實作一加法器，將兩個 32-bits 的 inputs 相加並輸出到 output。

6. MUX32：

根據 ALUSrc 的數值（0 或 1），決定 output 為 input 的 data1 或是 data2。

7. CPU：

定義更多的 wires，根據 modules 的輸入與輸出對應到的 wire，將整個電路連接起來。另外我在此處定義了一 parameter four = 4，作為常數來更新 pc。