Computer Architecture Final Project

經濟四 李品樺 B07303024

1. **Briefly describe your CPU architecture**

   我的 CPU 架構中首先會將從 mem_I 中讀取出的指令解碼(decode)，並且根據 opcode 和 funct3 & funct7 來決定這條指令要對 CPU 中的 reg_file 存取什麼地址(address)的資料。從 reg_file 存取出的 rs1_data 和 rs2_data 會根據指令進行相關的運算，像是 ADD, XOR 等。若要運算的指令是和存取記憶體有關(LW, SW)，要讀取或寫入的 memory address 由 rs1_data/rs2_data 和 imm 運算得出的結果決定。有關 JAL, JALR, AUIPC 則是將 PC 的值同樣透過計算後跳至結果，並且將原本 PC+4 存到相關的 rd 中。

2. **Describe how you design the data path of instructions not referred in the lecture slides (jal, jalr, auipc, ...)**

   我參考下面連結中的說明來設計 JAL, JALR, AUIPC, MUL 指令：

   RV32I, RV64I Instructions — riscv-isa-pages documentation (msyksphinz-self.github.io)

   其中 JAL 會把 PC 的值和從指令編碼出的 signed 21-bit imm 相加，加完後的答案是下一個 cycle 的 PC。另外也會把原本的 PC+4 存入 rd 中，使得之後 CPU 還可以跳回相關的位置。JALR 和 JAL 的處理方式相同，差別在於新的 PC 值是 rs1_data 和 signed 12-bit imm 相加的結果。AUIPC 的處理方式則是將現在 PC 和 unsigned 20-bit imm 的值相加存到 rd 中。MUL 的話我使用作業二設計的硬體來計算兩個 32-bit 的資料相乘的 64-bit 結果，並且將其中的第 0-31 bits 存到 rd 中。

3. **Describe how you handle multi-cycle instructions (mul)**

   我使用作業二的 mulDiv 設計，在 CPU 解碼出乘法的指令後其中的 FSM 會跳到特別設計的 S_EXEC_MUL 中。在此狀態下 CPU 會等待 mulDiv 回傳 ready 的訊號(代表乘法計算結束)再將乘法的結果存到 rd 中，並且在下個 cycle 繼續執行接下來的指令。

4. **Record total simulation time (CYCLE = 10 ns)**

   - **Leaf: a = 3, b = 9, c = 5, d = 17**

     275 ns

     ```
     Simulation complete via $finish(1) at time 275 NS + 0
     ./Final_tb.v:182              $finish;
     ```

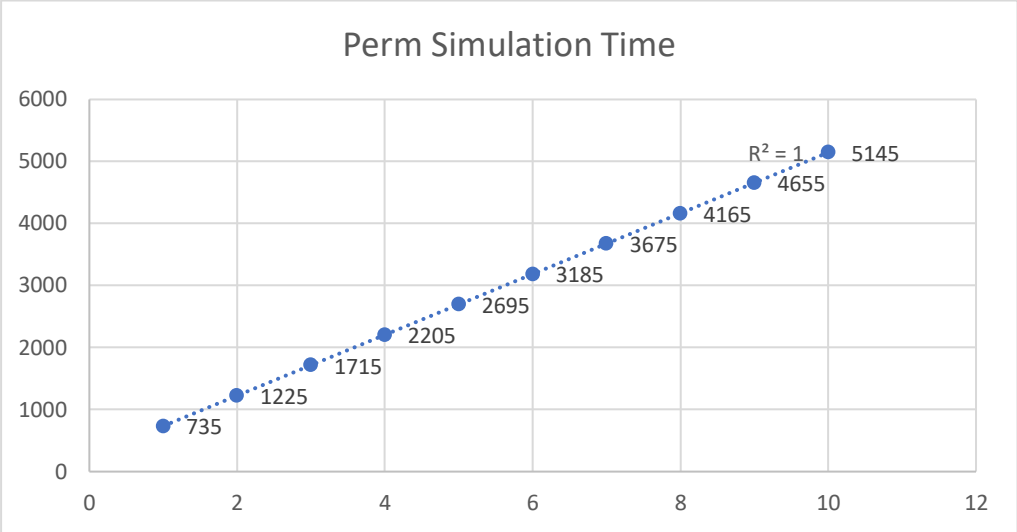
   - **Perm: n = 8, r = 5**

2695 ns

```
Simulation complete via $finish(1) at time 2695 NS + 0
./Final_tb.v:182              $finish;
```

5. Describe your observation

圖中的橫軸為 Perm 的 r 值，縱軸為模擬時間(單位為 ns)。可以觀察到模擬時間和 r 成線性的關係，並且在固定 n 的情況之下，當 r 的值增加 1，模擬時間增加 490ns。



6. Snapshot the “Register table” in Design Compiler

```
Inferred memory devices in process
        in routine CHIP line 243 in file
                '/home/raid7_2/userb07/b7303024/final/Verilog/CHIP.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|       PC_reg        | Flip-flop |  31   |  Y  | N  | Y  | N  | N  | N  | N  |
|       PC_reg        | Flip-flop |   1   |  N  | N  | N  | Y  | N  | N  | N  |
|     state_r_reg     | Flip-flop |   2   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================
```

```
Inferred memory devices in process
        in routine reg_file line 280 in file
                '/home/raid7_2/userb07/b7303024/final/Verilog/CHIP.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|       mem_reg       | Flip-flop |  994  |  Y  | N  | Y  | N  | N  | N  | N  |
|       mem_reg       | Flip-flop |  30   |  Y  | N  | N  | Y  | N  | N  | N  |
===============================================================================
```

```
Inferred memory devices in process
        in routine mulDiv line 473 in file
                '/home/raid7_2/userb07/b7303024/final/Verilog/CHIP.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|     alu_in_reg      | Flip-flop |  32   |  Y  | N  | Y  | N  | N  | N  | N  |
|       rdy_reg       | Flip-flop |   1   |  N  | N  | Y  | N  | N  | N  | N  |
|      state_reg      | Flip-flop |   3   |  Y  | N  | Y  | N  | N  | N  | N  |
|     counter_reg     | Flip-flop |   5   |  Y  | N  | Y  | N  | N  | N  | N  |
|      shreg_reg      | Flip-flop |  64   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================
```

7. List a work distribution table

N/A