RISC-V Single-cycle Processor Implementation Report

B06901190 陳昱仁 B06505022 袁肇謙 B06502029 陳心屏

- **CPU Architecture**

  按照課本上的硬體架構去寫，需要額外注意的是 memToReg 這邊會有很多訊號往 register file 回傳，因為 jal 和 jalr 是需要把 pc + 4 存回去的，以及 auipc 需要傳回現在的 pc，所以那邊我寫了一個 case 來做 decoder，和課本上架構用一個 mux 不同。以及需要額外加上 rs1_data 加上 immediate 的加法器，原因是因為需要支援 jalr 指令，才能使 CPU 正常運作。

- **Datapath**
  - jal

    The ImmGen extracts the immediate from the machine code of JAL. The Control unit sets ALUsrc to low, regWrite to high, memToReg to 2'b10, and ALUOp to 2'b00. The ALU control unit reads ALUOp which is equal to 0 that implies the ALU to do addition. The ALU reads rs1_data and rs2_data as its input. Finally, pc + 4 is written to the rd_data and then the pc will jump to the output of ALU.
  - jalr

    The ImmGen extracts the immediate from the machine code of JALR. The Control unit sets ALUsrc to high, regWrite to high, memToReg to 2'b10, and ALUOp to 2'b00. The ALU control unit reads ALUOp which is equal to 0 that implies the ALU to do addition. The ALU reads rs1_data and rs2_data as its input. Finally, pc + 4 is written to the rd_data and then the pc will jump to the output of ALU.
  - auipc

    The ImmGen extracts the immediate from the machine code of AUIPC. The Control unit sets ALUsrc to low, regWrite to high, memToReg to 2'b11, and ALUOp to 2'b00. The ALU control unit reads ALUOp which is equal to 0 that implies the ALU to do addition. The ALU reads rs1_data and immediate as its input. Finally, pc + immediate is written to the rd_data.

- **Multi-cycle Instructions**

  We assign the output signal "ready" and "valid" of multDiv hardware to the wire "lock", which enables the chip to complete the entire multiplication process before moving on to execute the next instruction.

- **Simulation**
  - Leaf

```
================================================================

Success!
The test result is .....PASS :)

================================================================

Simulation complete via $finish(1) at time 255 NS + 0
```

- Fact

```
================================================================

Success!
The test result is .....PASS :)

================================================================

Simulation complete via $finish(1) at time 4795 NS + 0
```

- HW1 for n = 10

```
================================================================

Success!
The test result is .....PASS :)

================================================================

Simulation complete via $finish(1) at time 2645 NS + 0
```

- **Observation**

  比較值得觀察的是 Mul 在執行時需讓 PC 不能更新，才可以一直執行 mul，並在結束後才更新 PC，並且還沒做完前不能更新 register，所以更新 PC 的 register 用了 wen 信號去控制，在 code 中為 lock 變數。

- **Register Table**

```
================================================================================
|    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  |
================================================================================
```

```
================================================================================
|    Register Name    |    Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
================================================================================
|       mem_reg       | Flip-flop |  995  |  Y  | N  | Y  | N  | N  | N  | N  |
|       mem_reg       | Flip-flop |  29   |  Y  | N  | N  | Y  | N  | N  | N  |
================================================================================
```

```
================================================================================
|    Register Name    |    Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
================================================================================
|      shreg_reg      | Flip-flop |  64   |  Y  | N  | N  | N  | N  | N  | N  |
|     alu_in_reg      | Flip-flop |  32   |  Y  | N  | N  | N  | N  | N  | N  |
|      state_reg      | Flip-flop |   2   |  Y  | N  | Y  | N  | N  | N  | N  |
|     counter_reg     | Flip-flop |   5   |  Y  | N  | Y  | N  | N  | N  | N  |
================================================================================
```

- **Work Distribution**

陳昱仁	袁肇謙	陳心屏
➢ ALU control ➢ ALU ➢ pc address ➢ memToReg	➢ ImmGen ➢ mulDiv ➢ report	➢ control unit ➢ hw1.s