NTUEE CA 2021

Group 25, b06502152 許書銓, b07204024 李昱呈, b03208039 鐘友棣

1. Briefly describe your CPU structure.

Basically, we design our CPU structure from concept of the lecture. We first, decode the instruction to realize what instruction it is, and set the correct control signals. Secondly, by the right control signals, we can compute the desired a ALU output from ALU operations. The next step is handling the memory writeing and register writing part from the control signals. Last, compute the correct PC.

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

- JAL

  set regWrite = 1, jump to PC + imm, and write rd as PC+4.
- JALR

  set regWrite = 1, jump to rs1 + imm, and write rd as PC+4.
- AUIPC

  set regWrite = 1, write rd as PC + imm.
- SLLI

  set regWrite = 1, and control alu to shift rs1 left imm bits.
- SRLI

  set regWrite = 1, and control alu to shift rs1 right imm bits.
- SLTI

  set regWrite = 1,ALU_src = 1, and set ALU as a SUB operation. If ALU_output < 0, set rd as 1, else set rd as 0.

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

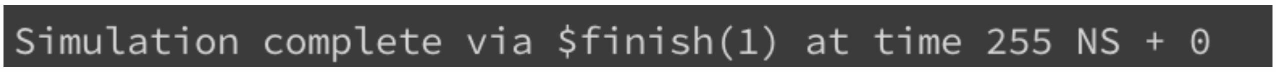
We instantiate `mulDiv` module in `ALU` module. Since `mulDiv` module requires `clk` and `rst_n` inputs, to let `mulDiv` module share the same `clk` and `rst_n` that were used by `CHIP` module, we added `clk` and `rst_n` inputs to `ALU` module, connect these inputs to `clk` and `rst_n` of `CHIP` module, and let `clk` and `rst_n` of `mulDiv` module connect to `clk` and `rst_n` of `ALU` module. Furthermore, `mulDiv` need 32 cycles to complete calculation, so we have to make sure the sequential part of `CHIP` module wait for the calculation to be done. That is, neither change PC nor write registers while `ALU` module is still computing. Hence, output `done`, which was assigned to the output `ready` of `mulDiv` module, was added to `ALU` module. `done == 1` indicates that the calculation was finished. Finally, we can add conditions to the sequential part:

```verilog
if (ALU_ctrl==3'b100) begin    // If ALU module encounters a MUL instruction
    if (alu_done==1) begin    // Modify PC and registers when computation was done
        PC <= PC_nxt;
        register[0] <= 32'd0;
        for(i = 1; i < 32; i = i+1) begin
            register[i] <= next_register[i];
        end
    end
    else begin    // Do not modify PC and registers before computation was done
        PC <= PC;
        register[0] <= register[0];
        for(i = 1; i < 32; i = i+1) begin
            register[i] <= register[i];
        end
    end
end
```

It can be seen easily that PC and registers change only when `MUL` was done.

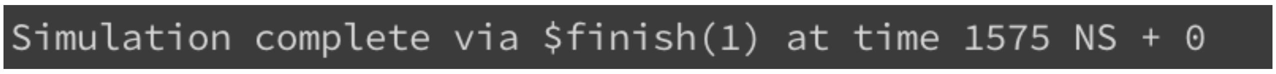
4. Record total simulation time (CYCLE = 10 ns)

- leaf program



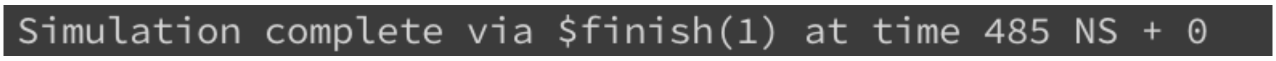
```
Simulation complete via $finish(1) at time 255 NS + 0
```

- fact program



```
Simulation complete via $finish(1) at time 1575 NS + 0
```

- hw1 program



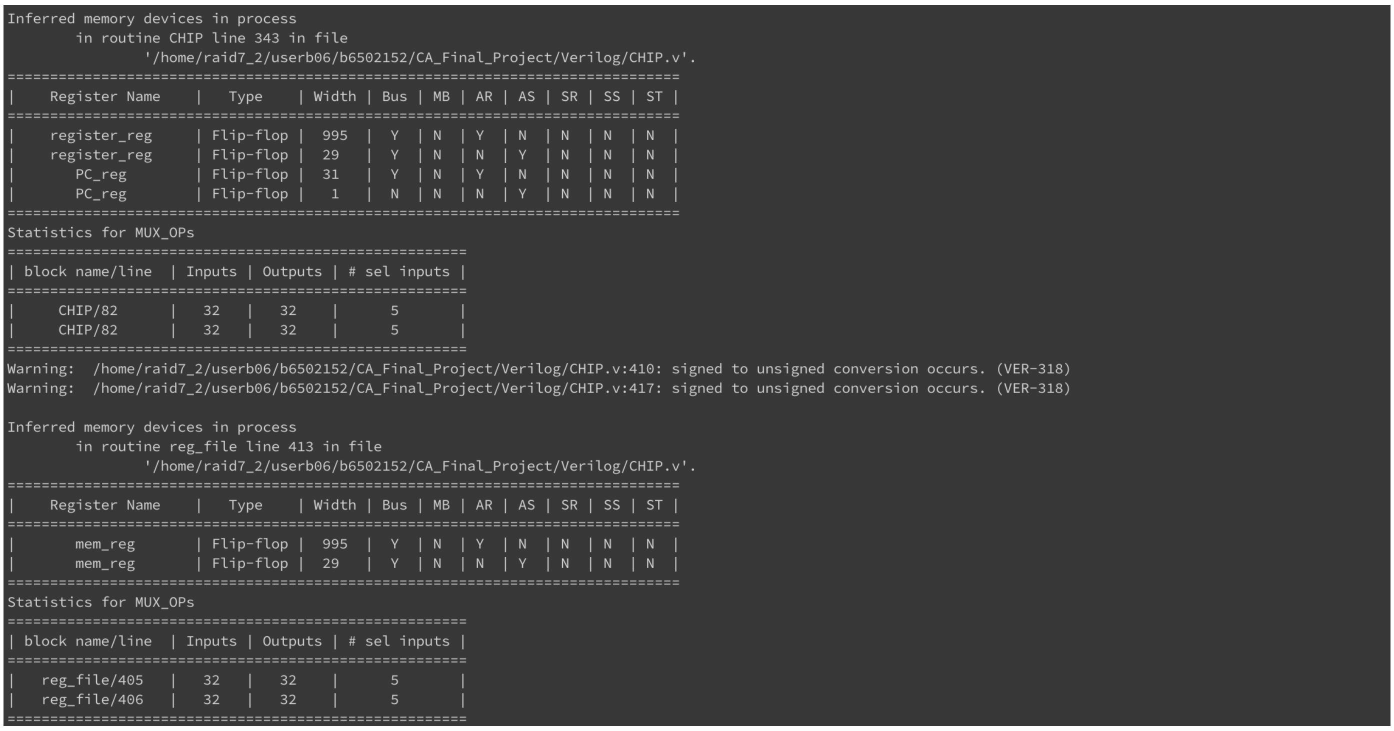
```
Simulation complete via $finish(1) at time 485 NS + 0
```

5. Describe your observation

Basically, we follow the architecture of CPU based on what is taught throigh the lectures. It is quite neat and well-behaved. Furthermore, we find that the risc v instruction are well-organized that we can used the same block of instruction as rd the same block of instruction as op code, which is quite convenient for us to handle.

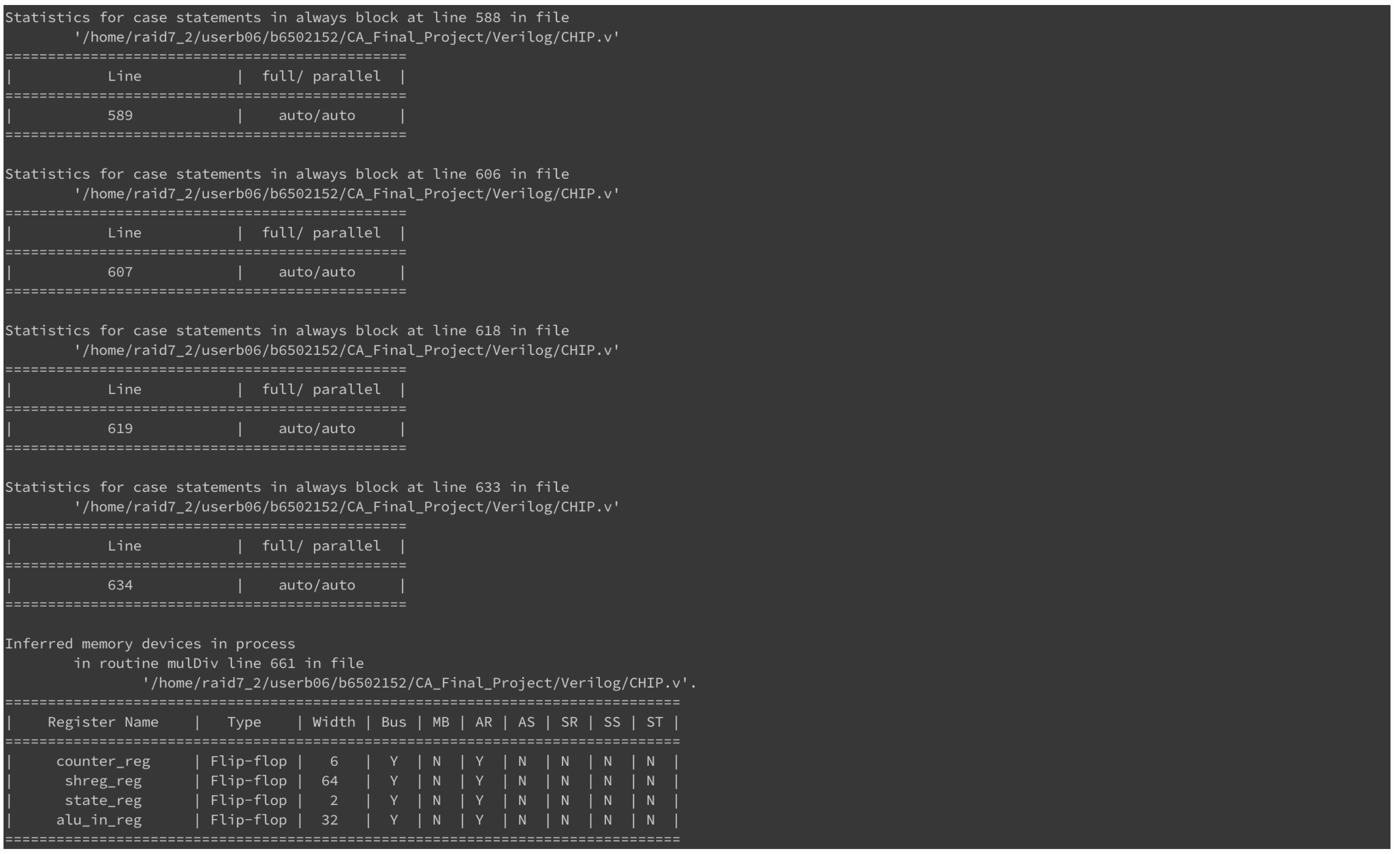
In bonus problem, we initially used MUL and DIVU instruction to implement multiplication and division. But when we used it, we found MUL and DIVU cost to many CPU cycles so as to easily reach the limit of total simulation time. In order to solve this problem, we changed the upper two instructions into SLLI and SRLI which only cost one CPU cycle and finally made our CPU work functionally.

6. Snapshot the "Register table" in Design Compiler



```
Inferred memory devices in process
        in routine CHIP line 343 in file
                '/home/raid7_2/userb06/b6502152/CA_Final_Project/Verilog/CHIP.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|    register_reg     | Flip-flop |  995  |  Y  | N  | Y  | N  | N  | N  | N  |
|    register_reg     | Flip-flop |  29   |  Y  | N  | N  | Y  | N  | N  | N  |
|       PC_reg        | Flip-flop |  31   |  Y  | N  | Y  | N  | N  | N  | N  |
|       PC_reg        | Flip-flop |   1   |  N  | N  | N  | Y  | N  | N  | N  |
===============================================================================
Statistics for MUX_OPs
======================================================
| block name/line  | Inputs | Outputs | # sel inputs |
======================================================
|     CHIP/82      |   32   |   32    |      5       |
|     CHIP/82      |   32   |   32    |      5       |
======================================================
Warning:  /home/raid7_2/userb06/b6502152/CA_Final_Project/Verilog/CHIP.v:410: signed to unsigned conversion occurs. (VER-318)
Warning:  /home/raid7_2/userb06/b6502152/CA_Final_Project/Verilog/CHIP.v:417: signed to unsigned conversion occurs. (VER-318)

Inferred memory devices in process
        in routine reg_file line 413 in file
                '/home/raid7_2/userb06/b6502152/CA_Final_Project/Verilog/CHIP.v'.
===============================================================================
|    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  |
===============================================================================
Statistics for MUX_OPs
======================================================
| block name/line  | Inputs | Outputs | # sel inputs |
======================================================
|   reg_file/405   |   32   |   32    |      5       |
|   reg_file/406   |   32   |   32    |      5       |
======================================================
```



```
Statistics for case statements in always block at line 588 in file
        '/home/raid7_2/userb06/b6502152/CA_Final_Project/Verilog/CHIP.v'
===============================================
|           Line           |  full/ parallel  |
===============================================
|           589            |    auto/auto     |
===============================================

Statistics for case statements in always block at line 606 in file
        '/home/raid7_2/userb06/b6502152/CA_Final_Project/Verilog/CHIP.v'
===============================================
|           Line           |  full/ parallel  |
===============================================
|           607            |    auto/auto     |
===============================================

Statistics for case statements in always block at line 618 in file
        '/home/raid7_2/userb06/b6502152/CA_Final_Project/Verilog/CHIP.v'
===============================================
|           Line           |  full/ parallel  |
===============================================
|           619            |    auto/auto     |
===============================================

Statistics for case statements in always block at line 633 in file
        '/home/raid7_2/userb06/b6502152/CA_Final_Project/Verilog/CHIP.v'
===============================================
|           Line           |  full/ parallel  |
===============================================
|           634            |    auto/auto     |
===============================================

Inferred memory devices in process
        in routine mulDiv line 661 in file
                '/home/raid7_2/userb06/b6502152/CA_Final_Project/Verilog/CHIP.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|     counter_reg     | Flip-flop |   6   |  Y  | N  | Y  | N  | N  | N  | N  |
|      shreg_reg      | Flip-flop |  64   |  Y  | N  | Y  | N  | N  | N  | N  |
|      state_reg      | Flip-flop |   2   |  Y  | N  | Y  | N  | N  | N  | N  |
|     alu_in_reg      | Flip-flop |  32   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================
```

7. work distribution table

student id	Workload
B06502152 許書銓	Task 1 - leaf, Writing report
B07204024 李昱呈	Bonus - hw1, Writing report
B03208039 鐘友隸	Task 2 - fact, Writing report