Single cycle CPU report

Chang Zihao
20206018
Cui Yuxuan
20206019

Contents

1 Introduction

Lab3 is required to design a Single-cycle CPU. A single-cycle CPU processes an instruction per every single clock cycle.

1.1 Data Path

1. Fetch the instruction at the address in PC.
2. Decode the instruction.
3. Execute the instruction.
4. Update the PC to hold the address of the next instruction.

1.2 Control-Single

1. Operation to be performed by ALU.
2. Whether register file needs to be written.
3. Signals for multiple intermediate multiplexors.
4. Whether data memory needs to be written.

2 Design

As the Description of TB file mentioned, We do not need to implement memory and register ourselves. So we design the path to connect them together. Then we have to design some necessary modules.

2.1 Module design

2.1.1 ALU module

ALU has been designed in lab1. But this time we need to modify the ALU slightly. At the beginning, I took out the Cout alone to deal specifically with branch instructions It works, but not necessary. So I put the Cout back and cancel it.

2.1.2 PC module

The PC is a state element that holds the address of the current instruction. It is updated at the end of every clock cycle.

2.1.3 add module

The adder is responsible for incrementing the PC to hold the address of the next instruction. It takes two input values, adds them together, and outputs the result

2.1.4 HALT module

HALT is used to halt the program when a specific condition is met. The terminate condition is ($RF\_RD1 = 0x0000000c$) when the received instruction is 0x00008067. HALT needs to set 1 when the terminal condition is met.

2.1.5 Sign extend module

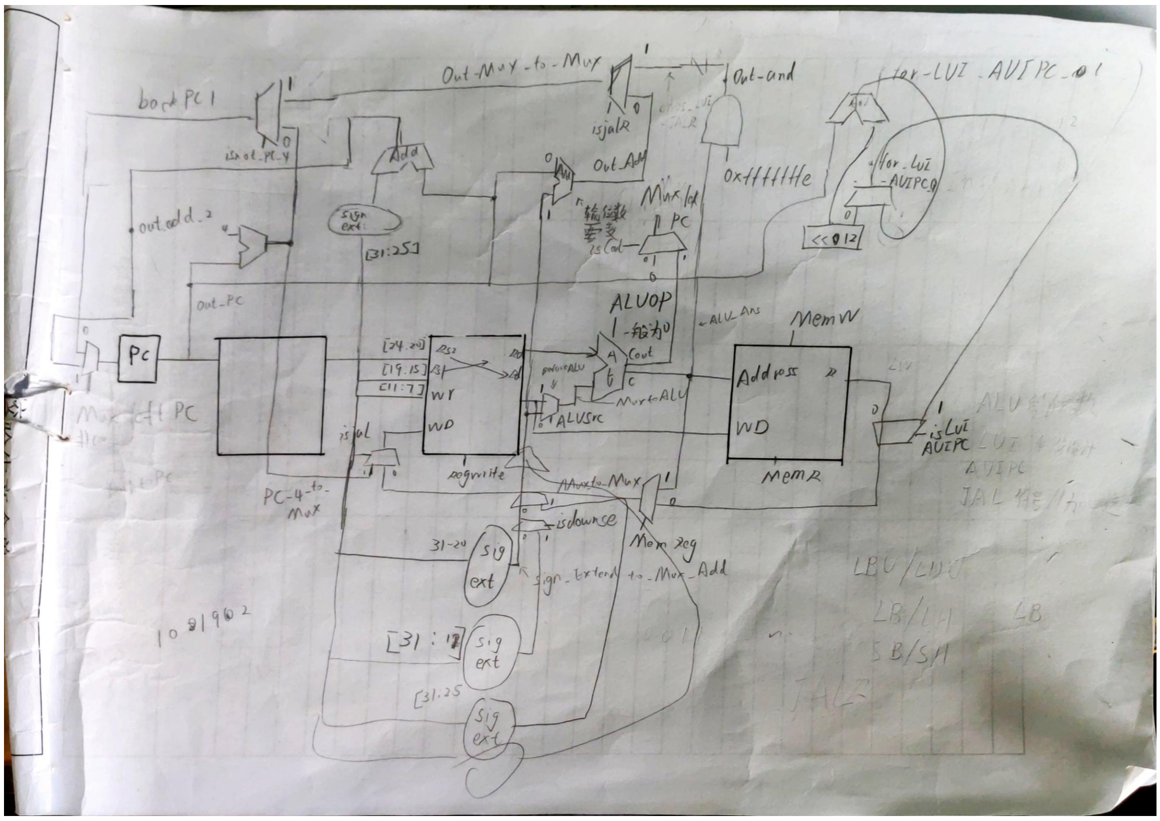
Single extend module is used to increase the number of bits of a binary number while preserving the number's sign and value.

2.1.6 MUX module

Mux, a data selector, which was used to select which the single will be output.

3 Implementation

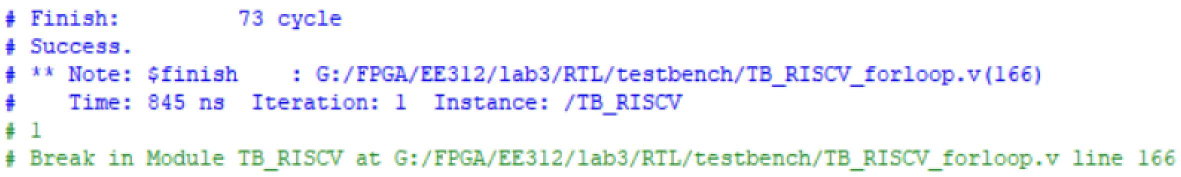
The whole circuit was shown below.



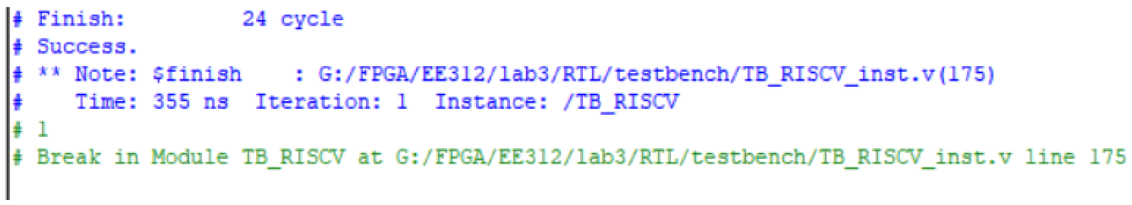
There might be something different with the circuit. Such as, I cancel the cout output.

4 Evaluation

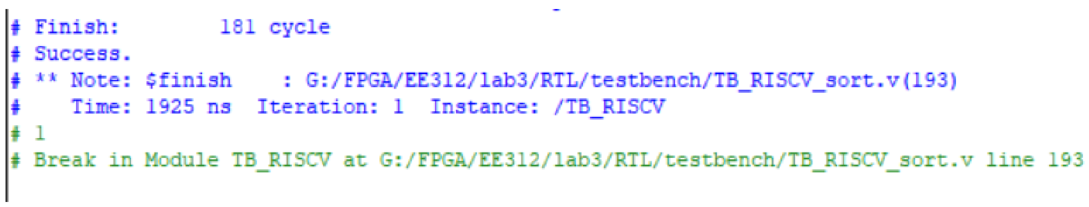
I pass all the test in the testbench folder.



```
# Finish:         73 cycle
# Success.
# ** Note: $finish    : G:/FPGA/EE312/lab3/RTL/testbench/TB_RISCV_forloop.v(166)
#    Time: 845 ns  Iteration: 1  Instance: /TB_RISCV
# 1
# Break in Module TB_RISCV at G:/FPGA/EE312/lab3/RTL/testbench/TB_RISCV_forloop.v line 166
```



```
# Finish:         24 cycle
# Success.
# ** Note: $finish    : G:/FPGA/EE312/lab3/RTL/testbench/TB_RISCV_inst.v(175)
#    Time: 355 ns  Iteration: 1  Instance: /TB_RISCV
# 1
# Break in Module TB_RISCV at G:/FPGA/EE312/lab3/RTL/testbench/TB_RISCV_inst.v line 175
```



```
# Finish:        181 cycle
# Success.
# ** Note: $finish    : G:/FPGA/EE312/lab3/RTL/testbench/TB_RISCV_sort.v(193)
#    Time: 1925 ns  Iteration: 1  Instance: /TB_RISCV
# 1
# Break in Module TB_RISCV at G:/FPGA/EE312/lab3/RTL/testbench/TB_RISCV_sort.v line 193
```

5 Conclusion

The single-cycle CPU is easy to design but have some disadvantages. Such as the clock cycle will be determined by the longest possible path, which is not the most common instruction.