pipelined CPU report

Chang Zihao
20206018
Cui Yuxuan
20206019

Contents

1 Introduction

Lab5 is required to design a pipelined CPU. A pipelined CPU processes five stage. And use pipelined to save time. Then deal with the data hazard and control hazard.

1.1 Data Path

1. Update the PC to hold the address of the next instruction and fetch the instruction at the address in PC.
2. Decode the instruction.
3. Execute the instruction.
4. Save data into the memory.
5. Use bypass unit to solve the data hazard.

1.2 Control-pipelined

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.
5. Control signal send on time at the first stage, then use register to send control signal to the module at the next stage.

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 Sign extend module

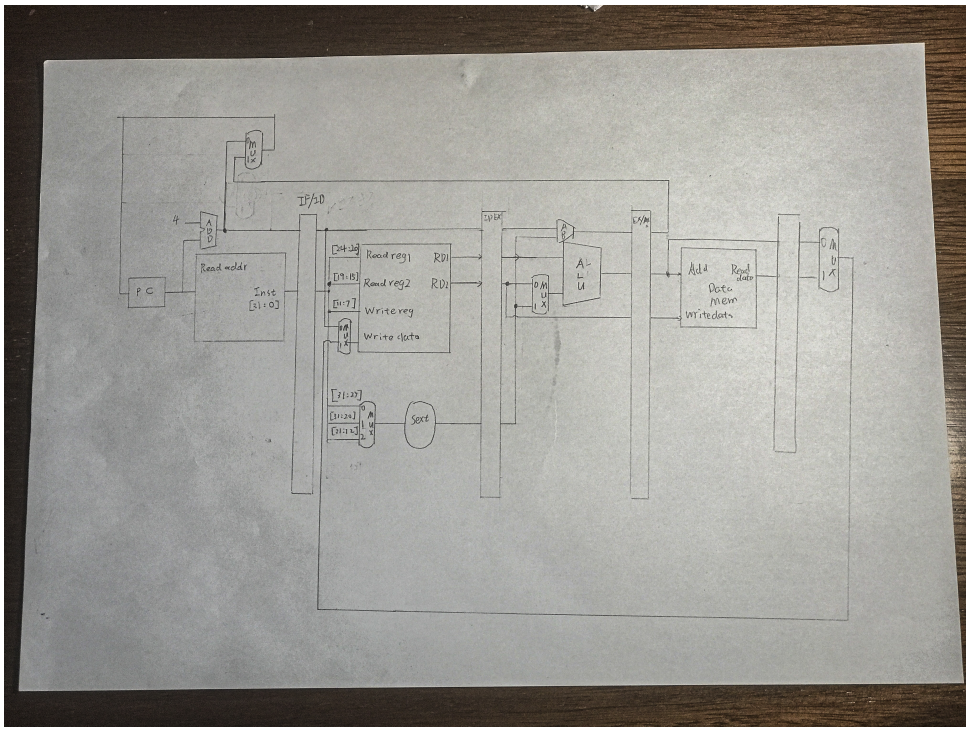
Single extend module is used to increase the number of bits of a binary number while preserving the number's sign and value.

2.1.5 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.



4 Evaluation

I pass all the test in the testbench folder.

```
# Test #    9 has been passed
# Test #   10 has been passed
# Test #   11 has been passed
# Test #   12 has been passed
# Test #   13 has been passed
# Test #   14 has been passed
# Test #   15 has been passed
# Test #   16 has been passed
# Test #   17 has been passed
# Test #   18 has been passed
# Test #   19 has been passed
# Break in Module HALT at G:/FPGA/EE312/lab5/RTL/modules/HALT.v line 9
VSIM 448> run -all
# Test #   20 has been passed
# Finish:         23 cycle
# Success.
# ** Note: $finish    : G:/FPGA/EE312/lab5/RTL/testbench/TB_RISCV_inst.v(173)
#    Time: 340 ns  Iteration: 1  Instance: /TB_RISCV
# 1
# Break in Module TB_RISCV at G:/FPGA/EE312/lab5/RTL/testbench/TB_RISCV_inst.v line 173

VSIM 448>
```

```
# Test #    8 has been successed!
# Test #    9 has been successed!
# Test #   10 has been successed!
# Test #   11 has been successed!
# Test #   12 has been successed!
# Test #   13 has been successed!
# Test #   14 has been successed!
# Test #   15 has been successed!
# Test #   16 has been successed!
# Test #   17 has been successed!
# Break in Module HALT at G:/FPGA/EE312/lab5/RTL/modules/HALT.v line 9
VSIM 450> run -all
# Finish:         97 cycle
# Success.
# ** Note: $finish    : G:/FPGA/EE312/lab5/RTL/testbench/TB_RISCV_forloop.v(167)
#    Time: 1085 ns  Iteration: 1  Instance: /TB_RISCV
# 1
# Break in Module TB_RISCV at G:/FPGA/EE312/lab5/RTL/testbench/TB_RISCV_forloop.v line 167
```

```
# Test #    33 has been successed!
# Test #    34 has been successed!
# Test #    35 has been successed!
# Test #    36 has been successed!
# Test #    37 has been successed!
# Test #    38 has been successed!
# Test #    39 has been successed!
# Test #    40 has been successed!
# Break in Module HALT at G:/FPGA/EE312/lab5/RTL/modules/HALT.v line 9
VSIM 444> run -all
# Finish:       12354 cycle
# Success.
# ** Note: $finish    : G:/FPGA/EE312/lab5/RTL/testbench/TB_RISCV_sort.v(193)
#    Time: 123650 ns  Iteration: 1  Instance: /TB_RISCV
# 1
# Break in Module TB_RISCV at G:/FPGA/EE312/lab5/RTL/testbench/TB_RISCV_sort.v line 19
```

5 Conclusion

The pipelined CPU is quicker than single-cycle CPU and multi-cycle CPU, but it is more difficult than single-cycle CPU and multi-cycle CPU.