Lab Report

Experiment:	**Single-cycle CPU Lab**
Department：	**Electrical Engineering**
Name:	**Xinyue Ma 20170765**
	Pengfei Gao 20206017

I INTRODUCTION

The central processing unit (CPU), as the computing and control core of a computer system, is the final execution unit for information processing and program operation. In this lab, we design a single-cycle CPU based on Verilog.

RISC-V is an open instruction set architecture (ISA) based on Reduced Instruction Set Computing (RISC). We aim to implement over 30 instructions of RISC-V in this lab.

II DESIGN

A CPU is composed of control units and data units. Data path perform data processing operations, registers and buses. Control path control the data path of the processor, it determines how to respond to the instruction that is fetched to the processor. In this lab, memory model, register file and clock signal is given, thus we just should implement other parts of CPU.

2.1 Data Path

2.1.1 ALU

ALU part should implement the different calculation function of two inputs according to the given instruction. There are total 16 kinds of instructions ALU should deal with.

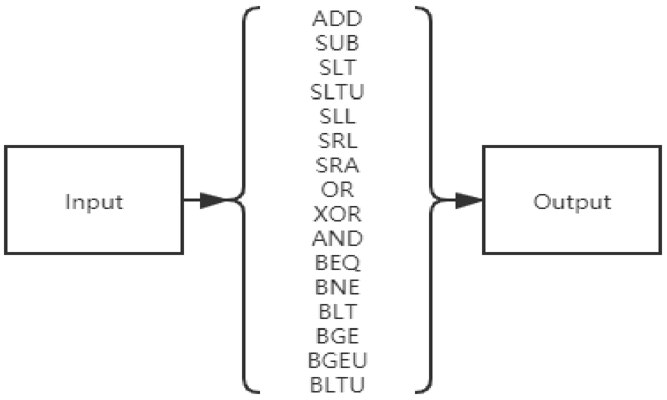


Figure 2.1 flow chart of ALU

2.1.2 Reconstruction of the immediate fields (module imm_processing)

Because the number of bits for the immediate is different for different instruciton type and the location of bits of the immediate is also scattered in the 32-bit instruction, we reconstruct the immediate field according to their types for ALU operation.

2.1.3 Memory output

As the data memory is byte addressable, the output of the D_MEM (D_MEM_IN) is processed before feeding to the registers according to the "byte enable" field, determined by the instruction (LB, LH,LW, LBU,LHU)

2.2 Control Path

In control path, there are total nine instruction types we should deal with. We use the first seven bits of the instruction (opcode) to determine which instruction type is it is and func3 and func7 function code to determine different behaviors of the ALU and muxes. The control path also checks whether the program is to be halted, signaled by intruction 0x00008067 and GPR(rs1) = 0x00c00093.

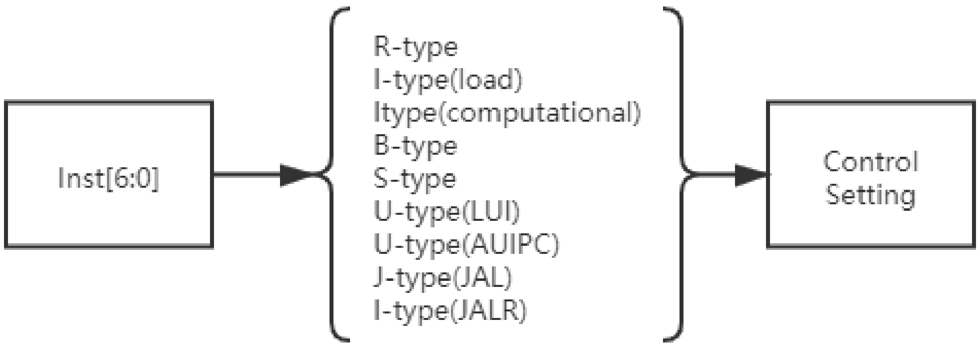


Figure 2.2 flow chart of control path

III IMPLEMENTATION

At every single clock cycle, CPU will perform Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory operation (MEM), and Write Back (WB) stages of a single instruction. Memory is divided into instruction part(4KB) and data part(16KB).

The Program Counter (PC) will be set to 0x000 and the stack pointer will be set to 0xF00 initially. When PC send the data to instruction memory, instruction memory send the instruction to register file to decode the instruction. Then ALU will do the operation of calculation and send it to data memory, and finally the output data will write back.

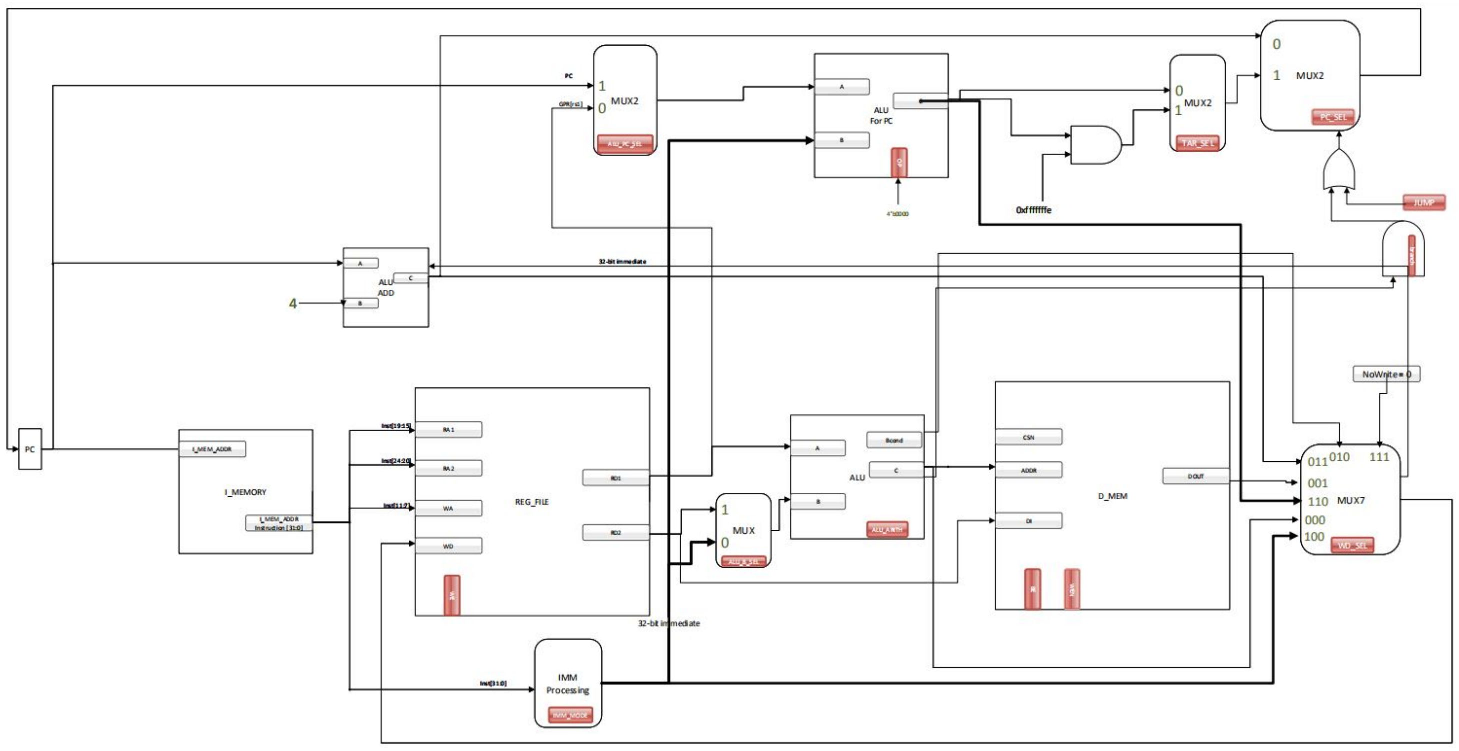
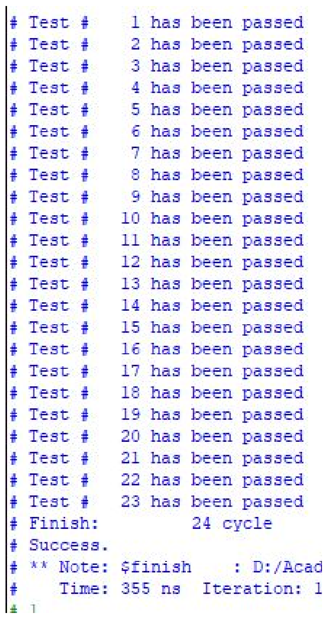


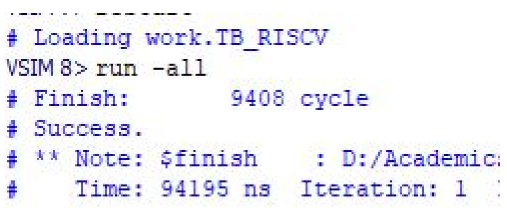
Figure 3.1 Overall design figure, outputs of Control Unit is colored red

IV EVALUATION

After finishing the code, we use the given Testbench files to evaluate the effectiveness. We used ModelSim to simulate our project, and according to the wave figure, all tests are pass.

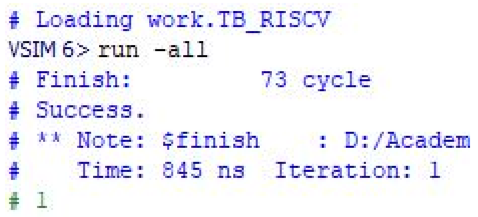


```
# Test #    1 has been passed
# Test #    2 has been passed
# Test #    3 has been passed
# Test #    4 has been passed
# Test #    5 has been passed
# Test #    6 has been passed
# Test #    7 has been passed
# Test #    8 has been passed
# Test #    9 has been passed
# Test #   10 has been passed
# 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
# Test #   20 has been passed
# Test #   21 has been passed
# Test #   22 has been passed
# Test #   23 has been passed
# Finish:         24 cycle
# Success.
# ** Note: $finish    : D:/Acad
#    Time: 355 ns  Iteration: 1
# 1
```



```
# Loading work.TB_RISCV
VSIM 8> run -all
# Finish:        9408 cycle
# Success.
# ** Note: $finish    : D:/Academic
#    Time: 94195 ns  Iteration: 1
```

For sort.asm



```
# Loading work.TB_RISCV
VSIM 6> run -all
# Finish:          73 cycle
# Success.
# ** Note: $finish    : D:/Academ
#    Time: 845 ns  Iteration: 1
# 1
```

For forloop.asm

For inst.hex

Figure 5.2 Results of the Test

V DISCUSSION

The hardest part building the CPU was debugging. There are not as many tools provided to debug verilog than provided to debug other popular programming language. We depended on inspecting the wave form and dataflow to debug our CPU. We think it is a good idea to include a short session to give tips on debugging verilog in the tutorial given at the beginning of the semester.

VI CONCLUSION

In this lab, we make a VISC-V Single-Cycle CPU based on Verilog. After simulation, every given test was passed. Thus, the lab achieves the expected result successfully.