Computer Architecture LAB 1 Report

1. Modules Explanation

Adder

Adder module reads the PC of current cycle and a value as input, and outputs the PC of next cycle. This module adds current PC with the value, and the result will be the PC of next cycle.

Control

Control module reads Opcode(instruction[6:0]) as input, and outputs ALUOp, ALUSrc, and RegWrite three control signals. In this lab, we always write back to register, so RegWrite should be set to 1. ALUOp and ALUSrc depend on Opcode[5], if Opcode[5] == 0, the instruction contains immediate segment, so ALUSrc should be set to 1(which means the ALU should read immediate as input), and ALUOp should be set to 01. Otherwise, ALUSrc should be set to 0 and ALUOp should be set to 00.

ALU_Control

ALU_Control module reads funct7(instruction[31:25]), funct3(instruction[14:12]), and ALUOp(output by Control) as input, and outputs ALUCtrl signal to determine function type. The differences in funct7, funct3, and ALUOp enable us to separate eight types of function, and thus map them to individual signal ALUCtrl.

The map of ALUCtrl signal to function:
000: add 001: and 010: xor 011: sll
100: sub 101: mul 110: srai

Sign_Extend

Sign_Extend reads to_extend(instruction[31:20]) and outputs immediate(the sign extended result). This module extends to_extend base on the MSB of to_extend. If it is 0, which means it is a positive number, we pad 0 for it. Otherwise, pad 1. Finally, store the result back to immediate.

MUX32

MUX32 module reads three input, rs2_data(data of register 2), immediate(the sign-extended value of imme field), sel(signal for whether to choose rs2_data or immediate, which is actually ALUSrc outputs by Control), and outputs ALU_i_2. This module selects output base on sel signal, if sel == 0, ALU_i_2 = rs2_data, else, ALU_i_2 = immediate. This module aims to decide whether to use the value of immediate.

ALU

ALU module reads ALU_i_1(data of register 1), ALU_i_2(output of MUX32), sel(which is ALUCtrl, output by ALU_Control), and outputs ALU_result(which should be write back to register rd). This module will decide a calculation type base on sel signal and implement that on ALU_i_1 and ALU_i_2, and store the result back to ALU_result.

CPU

CPU module consists of Adder, Control, ALU_Control, Sign_Extend, MUX32, ALU, PC, Instruction_Memory, and Register, 9 modules in total.
The workflow of CPU is

1. Get the current PC from module PC.
2. Use module Instruction_Memory to fetch instruction base on current PC and use module Adder to get the PC of next cycle.
3. Forward instruction[6:0] to module Control, and it will generate ALUOp, ALUSrc, and RegWrite, three signals.
4. Forward instruction [19:15], [24:20], [11:7], ALU_Result and RegWrite to module Registers, it will generate rs1_data(ALU_i_1) and rs2_data.
5. Forward Instruction[31:20] to module Sign_Extend, and it will output the extend result immediate.
6. Forward rs2_data, immediate, and ALUSrc to module MUX32, and it will output the selected value ALU_i_2 base on ALUSrc.
7. Forward instruction[31:25], [14:12] and ALUOp to module ALU_Control, and it will generate ALUCtrl to determine what action should ALU take.
8. Module ALU reads ALU_i_1, ALU_i_2, and ALUCtrl. And take the corresponding action to generate output ALU_Result, and pass it back to module Resisters.

2. Development Environment

MacOS with M1 Apple Chip