Computer Architecture LAB 2 Report

Modules Explanation

Adder:

Same as LAB 1, input two variable a and b, and output the sum c.

ALU:

Same as LAB 1, input two variable a and b, with a control signal ALUCtrl. Base on ALUCtrl to select the operation to perform and output the result out.

MUX32:

Same as LAB 1, input two variable a and b, with a control signal sel, Base on sel to choose whether to output a or b as out.

Control:

Most of it same as LAB 1, but has some modification.

Input Opcode and a signal NoOp generated by Hazard Detection Unit, Base on Opcode and NoOp, determine RegWrite, MemtoReg, MemRead, MemWrite, ALUOp, ALUSrc, and Branch, and output them.

If NoOp is 1, set all of them to 0, and Branch == 1 only when the instruction is beq.

ALU_Control:

Same as LAB 1, input funct7, funct3, and ALUOp, and use them to determine the output ALUCtrl.

Immediate_Gen:

Input instruction, and base on the instruction to extend the immediate value (sign extended).

Forwarding_Unit:

Input MEM_RegWrite, MEM_Rd, Ex_Rs1, Ex_Rs2, WB_RegWrite, WB_Rd.

base on the rules provided in SPEC, we can determine the output control signal Forward_A and Forward_B.

Forward_Mux:

Idea similar to MUX32, but can take four input and a two bits control signal sel, and output the selected one.

Hazard_Detection:

Input RS1addr, RS2addr, MemRead and RDaddr, and do some evaluation to determine the output control signal PCWrite, Stall, and NoOp.

if MemRead is true and RDaddr is same as RS1addr or RS2addr, set PCWrite to 0, Stall to 1, NoOp to 1. Otherwise, set PCWrite to 1, Stall to 0, NoOp to 0.

Pipeline Register(IF/ID, ID/EX, EX/MEM, MEM/WB)

Pipeline Register have four types, but the idea is almost the same.

Used to seperate the five pipeline stages. With pipeline register, we can divide the precedure into five stages.

In the pipeline registers, the input would be the output of the previous stage, and the output would be the input of the next stage.

The special part is if flush == 1, IF/ID should produce a bubble(set pc and instruction to 0).

If flush == 0 and stall == 0, the pipeline should perform as usual.

CPU:

Connect all the modules mentioned above together to implement the pipeline processor.

Difficulties Encountered and Solutions in This Lab

The Scale of this lab's CPU is too large, typo is a common problem.

I should strictly obey the naming rules like camelCase to prevent typo.

When use switch case, I should always set defaults.

Development Environment

OS: MacOS

Compiler: iverilog