Electrical Engineering
& Computer Science
University of Missouri

ECE 4270: Computer Organization, Spring 2024

LAB 5: Pipelined RISC-V Simulator

Handling Control Hazards

Scope

In this lab assignment, you need to extend the pipelined RISC-V simulator you have developed in Lab 4 (that detects data hazards and supports forwarding). Extended version should handle control hazards, and it should support branch and jump instructions.

1. Control Hazard Detection and Handling:

A control hazard occurs if there is an instruction that changes the control flow (e.g., conditional branches, and unconditional jumps). The program counter (PC) of the next instruction to be fetched cannot be known before the control instruction computes the outcome of branch (whether it should be taken, or not). For jump instructions, there is no condition to compute; however, the next instruction cannot be fetched until the new PC is computed.

An instruction is fetched in IF stage, then decoded in ID stage. Until the end of the ID stage, we cannot know if the given instruction is control instruction (i.e., branch or jump) or not. When the instruction is in ID stage, subsequent instruction will be fetched. So, assuming the instruction in ID stage is control instruction, there would be another instruction already fetched, although the outcome of the control instruction (in case of branch), thus the desired PC is not known, yet. Thus, the fetched instruction might be wrong. To avoid fetching new instructions until the new PC is evaluated (based on the outcome of branch), you need to introduce a pipeline stall. For the given RISC-V pipeline, we assume that the branch outcome is not known until the end of EX stage (the new PC based on the branch outcome will be available at the end of EX stage). Similarly, the new PC will be known at the end of EX stage for unconditional jump instructions.

There are two cases that you need to handle for branch instructions based on the outcome of the branch condition.

Case 1: Branch should not be taken

In this case, the new PC would be PC+4, so the instruction following the branch instruction in the instruction sequence should be executed. Notice that this instruction has been fetched already (at the time branch instruction is in ID stage, the following instruction is in IF stage). So, once the branch outcome is known (in this case, branch should not be taken) the following instruction can resume its progress in the

pipeline. Next cycle, following EX stage of branch instruction, the instruction can move into ID stage. Effectively, the fetched instruction would be stalled by one cycle.

The pipeline should look like the following (assuming ins+1 is not branch or jump):

ins (branch):	IF	ID	EX	MEM	WB		
ins+1:		IF	stall	ID	EX	MEM	WB
ins+2:			stall	IF	ID	EX	MEM	WB

Case 2: Branch should be taken

In this case, new PC would be different from PC+4 and we assume that it will be available at the same cycle that we know the branch outcome (i.e., in EX stage). In the following cycle, new instruction should be fetched by using the new PC. The instruction that was fetched at the time the branch instruction was in ID stage must be flushed out from the pipeline.

The pipeline should look like the following:

ins (branch):	IF	ID	EX	MEM	WB		
ins+1:		IF	stall	flush			
ins+j:			stall	IF	ID	EX	MEM	WB

Jump instructions are unconditional, so they always change the control flow. The implication is that the instruction following the jump instruction should be flushed out, and new instruction should be fetched after new PC is calculated. So, jump instructions can be regarded in the same way the branch instructions that are taken.

2. Branch and Jump Instructions that should be implemented

You need to implement the control flow instructions given in Lab 1. These were: BEQ, BNE, BLEZ, BLTZ, BGEZ, BGTZ, J, JR, JAL, JALR

3. Testing

You are given a test input file in Canvas that contains a set of instructions. You should use it to test your implementation.

Grading Rubric

Code: Pipelined RISC-V simulator that handles control hazards (75)
Report: 25 points

Pipelined RISC-V simulator that handles control hazards (75 points):

To get full credit for the code, your pipelined RISC-V simulator should handle the control hazards correctly.

Lab report (25 points):

Your report should give details about the work distribution within the group (who did what), milestones in your work, and your implementation decisions (why did you choose the way you did it, and/or how did you do that). Explain any difficulty you observed, and any optimizations you have made.

Submission

Please pay attention to the submission guidelines. You should submit the lab report along with the code you developed (provide makefile, as well). Please, generate a pdf file for your report and name it lab5_report_groupX.pdf, then place it into the folder called lab5_groupX (where X is your group number). The folder lab5_groupX should also contain the src/ folder that contains the simulator code (i.e., mu-riscv.h, mu-riscv.c and Makefile). Then, please compress the lab5_groupX folder as lab5_groupX.tar.gz and submit it through the Canvas.

Due Date

Your lab is due on: 04/18/2024