Computer Architecture

Homework 2 Report

B11901110 電機三 陳璿吉

April 27, 2025

Question 1

Implement simulation on `hw1_2.s` with different processors on Ripes and make discussion.

1. (5%) What is the cycle count when using 5-stage processor and 5-stage processor without forwarding unit?

   Answer:

Processor	Cycle Count
5-stage processor	812
5-stage processor without forwarding	1299

2. (10%) How does forwarding improve efficiency? Provide a small example from your code.

   Answer: As the previous table shows, forwarding improves the performance by 30%. If there's no forwarding, the processor has to stall or insert `nop` to prevent hazards. This is illustrated by the following two figures. They are running the same part of the program.

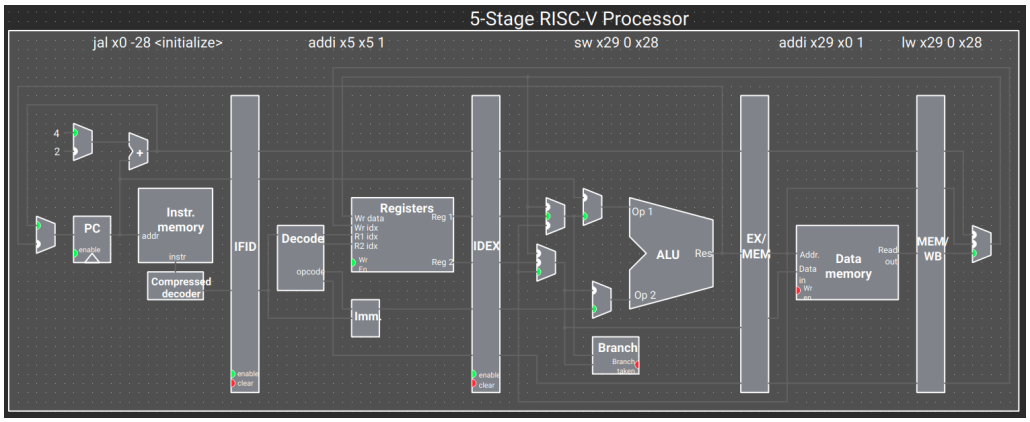

Figure 1: 5-stage processor

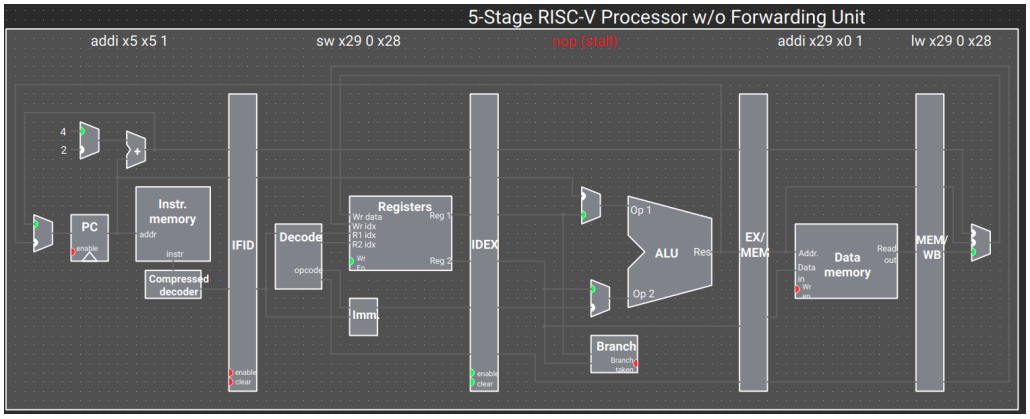

Figure 2: 5-stage processor without forwarding

Hence, with the forwarding design, we don't need to stall or insert `nop`, leading to improvement of performance.

3. (25%) Did your program run correctly under 5-stage processor w/o hazard detection? What kind of hazard might occur in this setting? Try to fix the program and compare the cycle count with 5-stage processor.

   Answer: The program didn't run correctly, but stopped halfway. Since there is no hazard detection, the program terminates just before "`ecall`". There might be hazards while executing "`li a0`". To solve this, I added 3 `nop` before `ecall`. However, the number of cycles was only 213, and the final answer `a0` was still incorrect. There must be **data hazards** while executing the loops.

   More specifically, we frequently load data from memory, and use it immediately in `hw1_2.s`. Due to the delay of each instruction, we may not be able to use the latest data when a new instruction needs it. We need to inserts `NOP` of stall the program to make sure it works properly.

Question 2

1. (30%) Run your Homework 1 assembly code (`hw1_2.s`) on gem5. Optimize the simulation time and list the changes in simulation time (ticks) before and after the modifications. Describe the methods you used to optimize the simulation time.

Version	`simTicks`
Original (`hw1_2.s`)	
Final (`hw2_2.s`)	

Question 3

Implement the same function using both `C++` and assembly.

1. (10%) Record the instruction count and execution time of two version.

Version	`simInsts`	`simTicks`
`hw2_assembly`		
`hw2_ccode`		

2. (20%) Discuss your observations. What are the advantages and disadvantages of these two implementations?

   Answer: Assembly code runs with more instruction, but with less time than the `C++` code. On the other hand, the `C++` code executes less instruction, but takes more time than the assembly code. Each instruction in assembly code is fundamental (only one step at a time), while the instructions in `C++` may be more complex (multiple steps in one instruction). However, `C++` is easier to write and read, which is overwhelmingly favored over assembly code.