CS520 Computer Architecture

Project 2 – Spring 2022

Due date: 04/11/2022

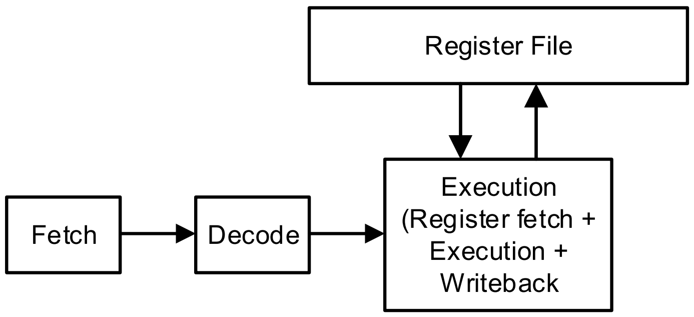
1. RULES

(1) All students must work alone. Cooperation is not allowed.
(2) Sharing of code between students is considered cheating and will receive appropriate action in accordance with University policy. The TAs will scan source code through various tools available to us for detecting cheating. Source code that is flagged by these tools will be dealt with severely.
(3) You must do all your work in the C/C++.
(4) Your code must be compiled on remote.cs.binghamton.edu or the machines in the EB-G7 and EB-Q22. This is the platform where the TAs will compile and test your simulator. They all have the same software environment.

2. Project Description

In this project, you will construct a simple pipeline with an instruction decoder.

3. Simple Pipeline



Model simple pipeline with the following three stages.

- 1 stage for fetch (IF)
- 1 stage for decode (ID)
- 1 stage for execution (EX)

The pipeline supports 4B fixed-length instructions, which have 1B for opcode, 1B for destination, and 2B for two operands. The destination and the left operand are always registers. The right operand can be either register or an immediate value.

Opcode (1B)	Destination (1B)	Left Operand (1B)	Right Operand (1B)

The supported instructions have nine different types as follows. The pipeline only supports integer arithmetic operations with 16 integer registers (R0 - R15), where each has 4B. All numbers between 0 and 1 will be discarded (floor).

Mnemonic	Opcode	Description		
		Destination	Left Operand	Right Operand
set	0x00	set Rx, #Imm (Set an immediate value to register Rx)		
		Register Rx	Immediate value	
add	0x10	add Rx, Ry, Rz (Compute Rx = Ry + Rz)		
		Register Rx	Register Ry	Register Rz
add	0x11	add Rx, Ry, #Imm (Compute Rx = Ry + an immediate valve)		
		Register Rx	Register Ry	Immediate value
sub	0x20	sub Rx, Ry, Rz (Compute Rx = Ry – Rz)		
		Register Rx	Register Ry	Register Rz
sub	0x21	sub Rx, Ry, #Imm (Compute Rx = Ry - an immediate valve)		
		Register Rx	Register Ry	Immediate value
mul	0x30	mul Rx, Ry, Rz (Compute Rx = Ry * Rz)		
		Register Rx	Register Ry	Register Rz
mul	0x31	mul Rx, Ry, #Imm (Compute Rx = Ry * an immediate valve)		
		Register Rx	Register Ry	Immediate value
div	0x40	div Rx, Ry, Rz (Compute Rx = Ry / Rz)		
		Register Rx	Register Ry	Register Rz
div	0x41	div Rx, Ry, #Imm (Compute Rx = Ry / an immediate valve)		
		Register Rx	Register Ry	Immediate value

An instruction is fetched at the fetch stage. The instruction is decoded at the decode stage, and it is executed at the execution stage. The fetch stage fetches one instruction every cycle unless the stage is occupied by a stalled instruction. The fetch and decode stage take 1 cycle. However, the execution stage requires varied cycles based on instruction types. The set (0x00), add (0x10, 0x11), and sub (0x20, 0x21) instructions take **1 cycle**, mul (0x30, 0x31) instructions take **2 cycles**, and div (0x40, 0x41) instructions take **4 cycles**.

Note, this pipeline has no hazards. The registers are read, executed, and updated by one instruction at the same stage (no data hazards). Therefore, the pipeline does not need to analyze data dependencies between instructions. Also, assume that your register file has two read ports and one write port (no structural hazards). Lastly, there are no branch instructions (no control hazards).

4. Validation and Other Requirements

4.1. Validation requirements

Sample simulation outputs will be provided on the website. You must run your simulator and debug it until it matches the simulation outputs. Your simulator must print the final contents in the register and performance results correctly as follows (the format is already coded).

```
Registers:
----------------------------------------
 R0:        |  R1:
----------------------------------------
 R2:       |  R3:
----------------------------------------
 R4:       |  R5:
----------------------------------------
 R6:       |  R7:
----------------------------------------
 R8:       |  R9:
----------------------------------------
 R10:       |  R11:
----------------------------------------
 R12:      |  R13:
----------------------------------------
 R14:        |  R15:
----------------------------------------

Total execution cycles:
IPC:
```

Your output must match both numerically and in terms of formatting, because the TAs will "diff" your output with the correct output. You must confirm correctness of your simulator by following these two steps for each program:

1) Redirect the console output of your simulator to a temporary file. This can be achieved by placing "> your_output_file" after the simulator command.

2) Test whether or not your outputs match properly, by running this unix command:
"diff –iw <your_output_file> <posted_output_file>"

The –iw flags tell "diff" to treat upper-case and lower-case as equivalent and to ignore the amount of whitespace between words. Therefore, you do not need to worry about the exact number of spaces or tabs as long as there is some whitespace where the sample outputs have whitespace. Both your outputs must be the same as the solution.

3) Your simulator must run correctly not only with the given programs. Note that TA will validate your simulator with hidden programs.

4.2. Compiling and running simulator

You will hand in source code and the TA will compile and run your simulator. As such, you must be able to compile and run your simulator on machines in EB-G7 and EB-Q22. This is required so that the TAs can compile and run your simulator. You also can access the machine with the same environment remotely at remote.cs.binghamton.edu via SSH.

The pipeline receives a program name.

e.g. simple_pipe trace1

5. What to submit

You must hand in project2.c. Also, you must submit a cover page with the project title, the Honor Pledge, and your full name as electronic signature of the Honor Pledge. A cover page is posted on the project website.

6. Penalties

Various deductions (out of 100 points):
-5 points for each date late during the first 6 days.
-10 points for each date late after the first 6 days.

Up to -10 points for not complying with specific procedures. Follow all procedures very carefully to avoid penalties.

Cheating: Source code that is flagged by tools available to us will be dealt with according to University Policy. This includes a 0 for the project and other disciplinary actions.