CS203 Fall ’18 - Lab 1

Due: Wednesday, October 24 midnight (11:59:59 pm)

# Objective

The goal of project 1 is to extend a simple pipeline simulator to support forwarding. For this lab, we will build off Pipesim, a timing simulator for 5-stage pipeline processor.  
If you have any questions, please post to Piazza as others may be having the same issue.

# Pipesim

Pipesim is available at the following git repository: <https://bitbucket.org/danwong/cs203-labs-f16>. Note that Pipesim is still under development and have limited feature sets. Pipesim was developed with the sole purpose of teaching computer architecture. Future extensions of Pipesim functionality are left to you. ☺

## Pipesim source code files

Pipesim consists of 3 main files: pipeline.cpp, pipeline.h, and main.cpp.

pipeline.h contains the class definitions for various objects.

pipeline.cpp implements the main functionality of the pipeline timing simulator.

Functions of interest are, but not limited to, the following:   
**bool** Pipeline**::**hasDependency(**void**) - Checks for hazards between the instruction in   
 the decode stage and execute/memory/writeback stage.   
**void** Pipeline**::**cycle(**void**) - Simulates the flow of instruction through the pipeline.   
 Currently stalls the pipeline if RAW hazards are detected.

main.cpp initializes the pipeline and loads the instruction trace to run.

## Pipesim inputs

Pipesim takes in two command line parameters:

$ ./pipesim

usage is

-i fileName: to run input file fileName

-f : for enabling forwarding

The forwarding option sets a flag to enable or disable forwarding. Currently, forwarding is not implemented. You will be implementing forwarding for this lab.

The input file is a sequence of instruction that will be simulated through the pipeline. We provide three sample inputs: instruction.txt, instruction2.txt, and instruction3.txt.

The input instruction format uses a simplified ISA format that includes all the information we need to model the timing of the pipeline. (All we really need are registers to detect RAW hazards.) Currently, the ISA does not support immediate values, offset addressing, branches, etc.

For example, LW R1, 100(R2), is simplified to LW R1 R2.

## Running Pipesim

The following shows an example run of Pipesim. Pipesim prints out the flow of the instructions through a 5-stage pipeline. Pipesim currently supports stalling to resolve RAW hazards.

$ ./pipesim -i instruction.txt

Loading application...instruction.txt

Read file completed!!

Printing Application:

ADD r1 r2 r3

SUB r2 r3 r4

MULT r3 r1 r5

DIV r4 r3 r6

LW r5 r4

SW r5 r7

BNEZ r7 r8

Initializing pipeline...

Cycle IF ID EXEC MEM WB

0 \* \* \* \* \*

1 ADD r1 r2 r3 \* \* \* \*

2 SUB r2 r3 r4 ADD r1 r2 r3 \* \* \*

3 MULT r3 r1 r5 SUB r2 r3 r4 ADD r1 r2 r3 \* \*

4 DIV r4 r3 r6 MULT r3 r1 r5 SUB r2 r3 r4 ADD r1 r2 r3 \*

5 DIV r4 r3 r6 MULT r3 r1 r5 \* SUB r2 r3 r4 ADD r1 r2 r3

6 LW r5 r4 DIV r4 r3 r6 MULT r3 r1 r5 \* SUB r2 r3 r4

7 LW r5 r4 DIV r4 r3 r6 \* MULT r3 r1 r5 \*

8 LW r5 r4 DIV r4 r3 r6 \* \* MULT r3 r1 r5

9 SW r5 r7 LW r5 r4 DIV r4 r3 r6 \* \*

10 SW r5 r7 LW r5 r4 \* DIV r4 r3 r6 \*

11 SW r5 r7 LW r5 r4 \* \* DIV r4 r3 r6

12 BNEZ r7 r8 SW r5 r7 LW r5 r4 \* \*

13 BNEZ r7 r8 SW r5 r7 \* LW r5 r4 \*

14 BNEZ r7 r8 SW r5 r7 \* \* LW r5 r4

15 \* BNEZ r7 r8 SW r5 r7 \* \*

16 \* \* BNEZ r7 r8 SW r5 r7 \*

17 \* \* \* BNEZ r7 r8 SW r5 r7

18 \* \* \* \* BNEZ r7 r8

19 \* \* \* \* \*

Completed in 18 cycles

# Assignement

Lab 1 consists of two parts:

1. **Forwarding Conditions**. List all necessary and feasible forwarding in the MIPS 5-stage pipeline (IF-ID-EX-MEM-WB, you can refer to it as F-D-X-M-W). Recall that forwarding is done between pipeline latches. Give an example for each necessary and feasible forwarding.
2. **Extend Pipesim** to support forwarding. In your extended version, the user of Pipesim should be able to enable forwarding for different *window sizes*. For doing so, extend Pipesim to accept an input argument with -f flag that accepts three possible values as follows:
3. Forwarding window size is zero, i.e., forwarding is disabled (-f 0 or the default value if –f flag not provided)
4. Forwarding window size is one, i.e., forwarding for EXEC/MEM pipeline register to EXEC stage is enabled (-f 1)
5. Forwarding window size is two, i.e., forwarding for EXEC/MEM pipeline register to EXEC stage and MEM/WB to EXEC stage is enabled (-f 2)

# Notes:

Please make sure that you consider below notes before submission:

1) Change the forwarding input argument of Pipesim in order to get an integer from 0 to 2. Please follow the below format:

$ ./pipesim

usage is

-i fileName : to run input file fileName

-f forwardingWindowWidth: for forwarding window width [0-2]

2) Provide a script for compiling and running Pipesim for executing instruction.txt , instruction2.txt, and instruction3.txt while:

- Forwarding is disabled  
- Forwarding is enabled for EXEC/MEM pipeline register to EXEC stage  
- Forwarding is enabled for EXEC/MEM pipeline register to EXEC stage and EM/WB to EXEC stage

3) For part 1, complete and submit the provided Lab1\_answer\_template.docx as a pdf file to GradeScope.

4)­­­­­-Include a printout of your simulation run when executing instruction.txt, instruction2.txt, and instruction3.txt. Submit your source code and your printout to iLearn in a single zip file.