COSC 530 – Fall 2020
Programming Assignment 2
Speculative Dynamically Scheduled Pipeline Simulator
DUE: 11:59pm on 11/10

Your assignment is to write a program that reads in a trace of RISC-V instructions to be executed and determines for each instruction the exact cycles when it proceeds through each appropriate stage of the dynamically scheduled pipeline. The report you generate should contain the conguration of the pipeline, the cycles when each instruction proceeds through the various stages of the pipeline, and the number of cycles associated with each type of delay.

You should first read in a configuration of the dynamically scheduled pipeline. This conguration will be read from the file `config.txt`, which is assumed to be in the current directory. An example conguration file is included in the starter package and is shown on the following page. You can make the following assumptions about the pipeline configuration:

1. The maximum total number of reservation stations that can be specified is 10.
2. The maximum number of entries in the reorder buffer is 10.
3. You should assume that latencies for an effective address calculation, a branch operation, and other integer operations are 1.
4. You should also assume that all branches are predicted correctly.

The instructions will be read in from standard input. You should only accept the following types of RISC-V instructions as input:

class	instructions
data transfers	lw, flw, sw, fsw
arithmetic	add, sub
control	beq, bne
floating-point	fadd.s, fsub.s, fmul.s, fdiv.s

Since we are only simulating the pipeline and not the actual effects of the instructions, you will need to include the address accessed by loads and stores, which will be given to the right of the instruction. This will allow you to dynamically disambiguate memory references to check for dependences between a store and a load. An example file of instructions is available in `trace.dat` in your starter package and is shown on the following page.

The configuration, results of the pipeline simulation, and the information about delays should be written to standard output. You should makethe following assumptions for the simulation:

1. At most one instruction can be issued per cycle.
2. At most one instruction can write its result on the CDB per cycle.
3. At most one data memory access (memory read or memory write) can occur per cycle.
4. At most one instruction can be committed per cycle.

The output for the specified configuration and instructions is shown on the last page of the assignment. You should make your output match mine exactly. My executable for the simulation is available in the starter package and is named `dynamsched`. To understand more about the details of the simulation, you can execute my simulator using the -v option on the command line.

```
> dynamsched -v < trace.dat
```

This will give a detailed state of the reservation stations, reorder buffer, and register status after each cycle. Note that you are not required to give such information in your assignment.

When you have completed your assignment, you should upload a gzipped tar file (created with `tar cvzf ...`) with your source files and instructions for building your simulator to the Canvas course website by the submission date and time at the top of this assignment. Your simulator should build and run on the `hydra` machines. Partial credits will be given for incomplete efforts. However, a program that does not compile or run will get 0 points. Point breakdown is below:

- Configuration file parsed and all options set correctly (10)
- Instructions executed / committed in the correct order (20)
- Pipeline stages / latencies simulated correctly (20)
- Functional units / buffers simulated correctly (20)
- Dependences and hazards simulated correctly (20)
- Output is in the appropriate format (10)

The text below shows an example configuration file, input trace data file, and their corresponding output with our solution simulator.

config.txt

```
buffers

eff addr: 2
fp adds: 3
fp muls: 3
ints: 2
reorder: 5

latencies

fp_add: 2
fp_sub: 2
fp_mul: 5
fp_div: 10
```

trace.dat

```
flw    f6,32(x2):0
flw    f2,48(x3):4
fmul.s f0,f2,f4
fsub.s f8,f6,f2
fdiv.s f10,f0,f6
fadd.s f6,f8,f2
```

output

```
Configuration
-------------
buffers:
   eff addr: 2
    fp adds: 3
    fp muls: 3
       ints: 2
    reorder: 5

latencies:
   fp add: 2
   fp sub: 2
   fp mul: 5
   fp div: 10


                   Pipeline Simulation
------------------------------------------------------------
                                      Memory Writes
     Instruction      Issues Executes  Read  Result Commits
-------------------- ------ -------- ------ ------ -------
flw    f6,32(x2):0         1   2 -  2      3      4       5
flw    f2,48(x3):4         2   3 -  3      4      5       6
fmul.s f0,f2,f4            3   6 - 10            11      12
fsub.s f8,f6,f2            4   6 -  7             8      13
fdiv.s f10,f0,f6           5  12 - 21            22      23
fadd.s f6,f8,f2            6   9 - 10            12      24


Delays
------
reorder buffer delays: 0
reservation station delays: 0
data memory conflict delays: 0
true dependence delays: 11
```