

Homework #4

TA in charge: Yue Chen
E-mail: yuechen@kaist.ac.kr

I. Goal of this assignment

- ✓ Understand what a computer architecture simulator is
- ✓ Understand what a benchmark is and how benchmarks are different from each other
- ✓ Execute a simulator, ‘SimpleScalar’ with a benchmark suite, ‘SPEC2000’
- ✓ Learn branch prediction design through the analysis of simulation results

II. Submission and grading

- ✓ You must submit all of the following (**Total of 100 points**):

<hw4_StudentID.zip>, <hw4_StudentID.pdf>

A. Simulation result files (**20 points**):

A **zip file** containing the simulation result folder <hw4_result>: The folder must include several folders: the folders each corresponding to the experimenting branch predictors parameters. It may also include the <swim_result.txt> generated by manually typing the command when you run a simulation. Compress this <hw4_result> folder into a zip file.

You must submit all these result files without any modifications.

The compressed zip file name must be <hw4_StudentID.zip>.

e.g., hw4_20241234.zip

B. Analysis report about simulation results (**80 points**):

Analyze the characteristics of the five benchmarks based on the simulation result. *There is no specific format for this report.* You can use any information if it is in the result files. You may also refer to any other literature about SimpleScalar and SPEC2000. You could include tables or figures to complement your analysis.

* Be cautious with the exact meaning of each value presented in result files.

e.g. ‘sim_elapsed_time’ represents the time elapsed on the system where you are running the simulation; not the time elapsed in the *simulated system*. Hence, it is irrelevant to the performance of the *simulated system*.

* You don’t have to understand all the values presented in the simulation result since many of them are not even covered in this course. But try to analyze the simulation results *in your own words* based on the knowledge covered in this course.

The report file name **must be** <hw4_ StudentID.pdf>.

e.g. hw4_20241234.pdf

Do not submit a non-pdf file.

You can only use English in the report.

- ✓ *Compress all these files into a .zip file and upload it on KLMS.*
- ✓ *Homework #4 is an individual assignment. You can talk and discuss with each other, but you cannot simulate benchmarks or write a report with others.*

III. Due date

- ✓ **Dec. 12th (Thu.) 23:59:00**
- ✓ **Late submission due date: Dec. 13th (Fri.) 23:59:00**
- ✓ If you miss the due date, there will be a **50% deduction from your total score.**
- ✓ **You cannot submit after the late submission due date.**

IV. Cheating

- ✓ If there are any cheatings in your submission, you will get **0 points.**
- ✓ *The following will be regarded as cheating:*
 - A. Copying other students' simulation results or reports
 - B. Modifying other students' results and using them as if they were your own
 - C. Using other sources without any references excluding your own simulation results
 - D. All other sorts of inappropriate behaviors

V. Prerequisite

- ✓ Follow the guide <**Environment_setup_guide.pdf**> to setup the simulation environment for this homework.

VI. Simulator and benchmark

- ✓ Simulator: SimpleScalar
 - 'SimpleScalar' is one of the classical simulators used in computer architecture research. With this simulator, you can simulate a CPU-based computer system and evaluate/estimate its performance for various program applications.
 - If you want to know about 'SimpleScalar' simulator, please check out <http://www.simplescalar.com/>.

✓ Benchmark suite: SPEC CPU2000

Basically, benchmarks are a kind of testing program to quantify the relative performance of various computing systems. ‘SPEC CPU’ is one of the most representative benchmark suites, including integer-intensive workloads and floating-point-intensive workloads. In this homework, you will use 5 benchmarks selected from SPEC CPU2000.

- A. gcc
- B. mcf
- C. bzip2
- D. swim
- E. art

If you want to know more about ‘SPEC CPU2000’, please read <SPEC2000.pdf> uploaded on KLMS.

VII. Execution example

- ✓ After setting up simulation environment with <Environment_setup_guide.pdf>, you should have an access to ‘simplesim-3.0’ directory with a terminal.
- ✓ This example assumes you have set up the simulation environment using the *provided VM image*. But the flow should mostly be the same for any other environments.
- ✓ This example also assumes that you are not familiar with Unix/Linux shell.
- ✓ As following this execution example, you can simulate benchmarks and get 5 result files.

- A. Open a new terminal and change the current directory to ‘simplesim-3.0’. You can do this by *typing the command below, next to the dollar sign (\$)*.

```
cs311@cs311:~$ cd simplesim-3.0
```

‘cd’ is a command to change the current directory.

‘simplesim-3.0’ is a directory name, which contains SimpleScalar source code files, tool scripts for the simulator, SPEC CPU2000 benchmark suite directory ‘SPEC2000’, and a script for this homework ‘hw4.sh’.

After typing the command, press ‘Enter’. Then you will see a new sentence on the terminal as follows.

```
cs311@cs311:~/simplesim-3.0$
```

*** If you didn’t use the provided VM image**, change the current directory to the ‘simplesim-3.0’ directory extracted from the archive file.

- B. Build SimpleScalar simulator by *typing the command below*.

```
cs311@cs311:~/simplesim-3.0$ make all
```

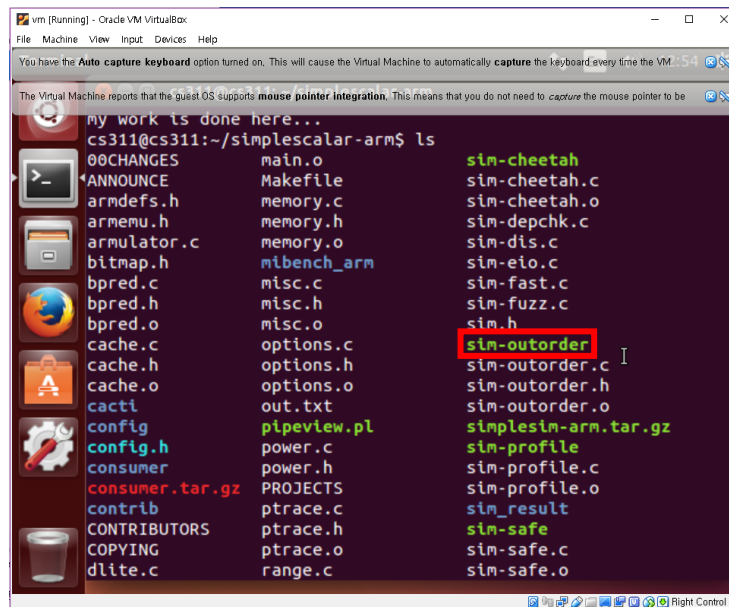
This should not fail if you are using the provided VM image and have correctly followed the guide. This will generate several *executable files*.

- C. List current directory by *typing the next command*.

```
cs311@cs311:~/simplesim-3.0$ ls
```

You will see several *executable files* within the listed files. Executable file names are colored in light green.

In HW4, you will use ‘sim-outorder’ executable file to run simulations.



- D. Before starting simulations for HW4, we recommend you a test run of ‘sim-outorder’.

Type the command below and press ‘Enter’ to run a simulation. The command is quite long, so please type carefully to not make typos. Be cautious if you copy and paste, since directly copying from a PDF file can deform the text.

```
cs311@cs311:~/simplesim-3.0 $ ./sim-outorder \
-redir:sim hw4_result/swim_result.txt \
-max:inst 10000000 SPEC2000/spec2000binaries/swim00.peak.ev6 \
< SPEC2000/spec2000args/swim/swim.in > /dev/null
```

Be careful about several dots (.), bars (-), and under-bars (_). There are 7 zeros. With the Korean keyboard, you can write back-slash (\) as the ‘won (₩)’ key. When you press ‘Enter’ after writing the back-slash, the terminal automatically makes an inequality sign (>). It just means that the command is not finished yet. Ignore it and write the next command line. Check twice you wrote every single

character including spaces correctly.

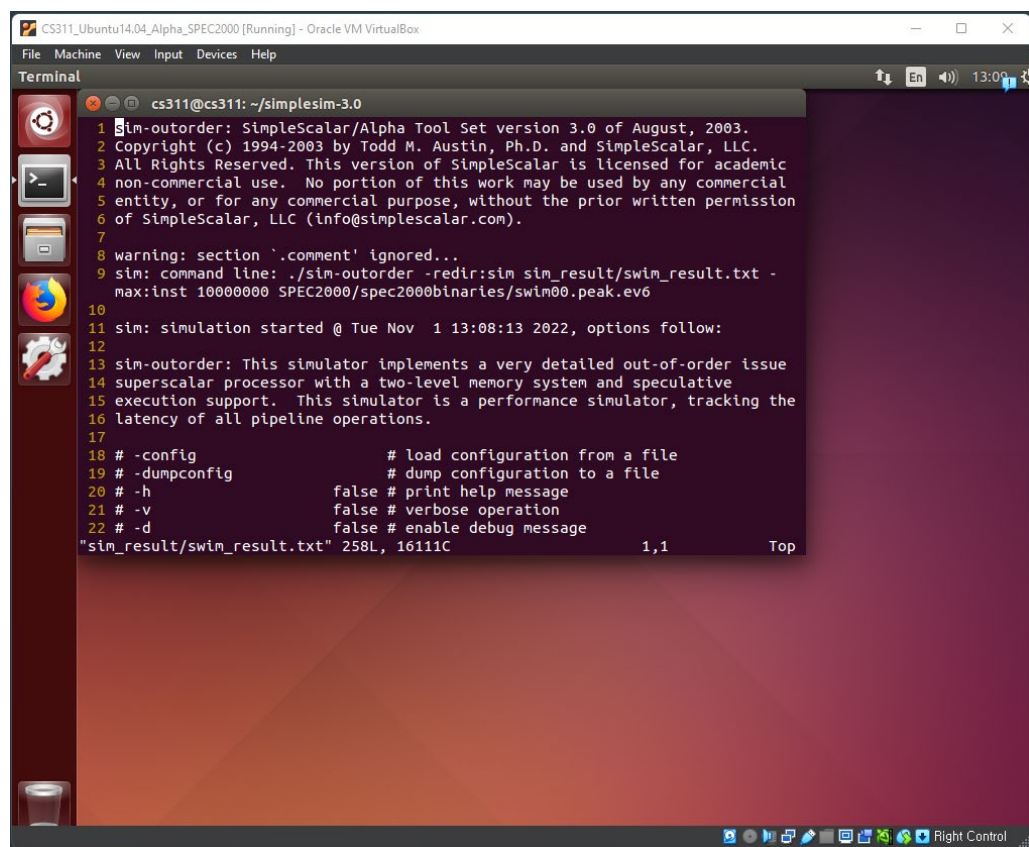
Wait until 'cs311@...' line is shown again.

- E. When the simulation completes, *type the command* below to check the simulation result.

```
cs311@cs311:~/simplesim-3.0$ vim hw4_result/swim_result.txt
```

'vim' is one of the text editors like 'notepad' in Windows. You can use any other text editors if you prefer.

You can see the contents of 'swim_result.txt' text file as follows.



The screenshot shows a terminal window titled 'CS311_Ubuntu14.04_Alpha_SPEC2000 [Running] - Oracle VM VirtualBox'. The terminal output is as follows:

```
cs311@cs311: ~/simplesim-3.0
1 sim-outorder: SimpleScalar/Alpha Tool Set version 3.0 of August, 2003.
2 Copyright (c) 1994-2003 by Todd M. Austin, Ph.D. and SimpleScalar, LLC.
3 All Rights Reserved. This version of SimpleScalar is licensed for academic
4 non-commercial use. No portion of this work may be used by any commercial
5 entity, or for any commercial purpose, without the prior written permission
6 of SimpleScalar, LLC (info@simplescalar.com).
7
8 warning: section '.comment' ignored...
9 sim: command line: ./sim-outorder -redir:sim sim_result/swim_result.txt -
  max:inst 10000000 SPEC2000/spec2000binaries/swim00.peak.ev6
10
11 sim: simulation started @ Tue Nov  1 13:08:13 2022, options follow:
12
13 sim-outorder: This simulator implements a very detailed out-of-order issue
14 superscalar processor with a two-level memory system and speculative
15 execution support. This simulator is a performance simulator, tracking the
16 latency of all pipeline operations.
17
18 # -config          # load configuration from a file
19 # -dumpconfig      # dump configuration to a file
20 # -h              false # print help message
21 # -v              false # verbose operation
22 # -d              false # enable debug message
"sim_result/swim_result.txt" 258L, 1611C          1,1          Top
```

The first several lines are explanation of simulator configuration. You can scroll down and find the sentence, “sim: ** starting performance simulation **”.

```

129
130
131
132 sim: ** starting performance simulation **
133 warning: partially supported sigprocmask() call...
134 warning: partially supported sigaction() call...
135 warning: unsupported setsysinfo() call...
136
137 sim: ** simulation statistics **
138 sim_num_insn      10000002 # total number of instructions committed
139 sim_num_refs      2452083 # total number of loads and stores
    committed
140 sim_num_loads      2201025 # total number of loads committed
141 sim_num_stores     251058.0000 # total number of stores committed
142 sim_num_branches   335426 # total number of branches committed
143 sim_elapsed_time   1 # total simulation time in seconds
144 sim_inst_rate      10000002.0000 # simulation speed (in insts/sec)
145 sim_total_insn     10016863 # total number of instructions executed
146 sim_total_refs     2455387 # total number of loads and stores
    executed
147 sim_total_loads    2203903 # total number of loads executed
148 sim_total_stores    251484.0000 # total number of stores executed
149 sim_total_branches 336397 # total number of branches executed
                                129,0-1 54%
  
```

After that line, it shows performance simulation results. *Each line is composed of three parts.*

Statistics value name	Measured value	# Meaning of value
-----------------------	----------------	--------------------

- F. You need to use these simulation results to write the analysis report. After confirming the simulation results, *you can exit the ‘vim’ editor with the command.*

```
:q
```

When you type the command, it shows the last line of the terminal. If you press ‘Enter’, the ‘vim’ editor will be closed and the terminal will be back to the state before starting the ‘vim’ editor.

If you are new to vim, try the command ‘**vimtutor**’ on the terminal. It will give you a simple, but practical tutorial.

- G. If there are no issues with the simulation result, it means there are no problems in the simulator. Now, you can simulate with the selected five benchmarks. *There are two ways to simulate these benchmarks.*

1. Simulate the other five benchmarks as described earlier

By using commands similar to the one used earlier, other benchmarks can be run similarly. However, it could be a little bit bothersome since each

benchmark requires different arguments. This means you have to examine which benchmark needs which arguments and how to pass them correctly. Furthermore, typing commands one by one may result in fat-finger errors. Thus, we highly recommend the way below:

2. Run a batch script to run a simulation for all five benchmarks at once

If you want to simulate the five benchmarks at once, you can use the shell script included in the ‘simplesim-3.0’ directory, ‘hw4.sh’.

A shell script is a file that has a set of shell commands. *You can execute commands in the script by calling the shell script as follows.*

```
cs311@cs311:~/simplesim-3.0$ ./hw4.sh
```

This way is far less error-prone, but you would need to wait a little longer until all the simulations are complete.

Even though it is simulated at once, simulation results are not affected. (In fact, this shell script just sequentially executes the six simulations.)

- H. After the end of the simulation, the result files will be saved into the ‘hw4_result’ directory. *You can go to the ‘hw4_result’ directory, and confirm results with the ‘vim’ editor.* There will be 6 directories in ‘hw4_result’ named **<nottaken taken bimod 2lev comb perfect>** for different branch predictors. In each directory, there should be five simulation results named:
<gcc_result.txt, mcf_result.txt, bzip2_result.txt, swim_result.txt, art_result.txt>.

```
cs311@cs311:~/simplesim-3.0$ cd hw4_result/<bp_name>
cs311@cs311:~/simplesim-3.0/sim_result$ vim ./<benchmark>_result.txt
```

VIII. Analyze Simulation Results

SimpleScalar provides six configurations for branch predictor type: *nottaken*, *taken*, *perfect*, *bimod*, *2lev*, and *comb*. *nottaken* and *taken* do static predictions: they always predict branches as not taken or taken. *perfect* literally does perfect prediction on branches (of course, this is not a practical branch predictor). *bimod* is a bimodal predictor, based on a branch target buffer (BTB) with 2-bit saturating counters. *2lev* is a 2-level predictor, which is also called a correlating predictor (see textbook 4.8). *comb* is a combined predictor, which adaptively chooses between bimodal and 2-level predictor (also see textbook 4.8; tournament predictor).

SimpleScalar allows to configure the design parameters of each predictor, such as table size. You can check these configurable parameters from the help message by running the simulator with a single argument “-h” (in fact, simply running without any arguments prints the same message). In the help message, you would be able to find options starting with “-bpred”, which are the options used to configure the design parameters.

Your submitted Lab report should cover the following topics:

A. Characterize the benchmark workloads

- i. Investigate what these selected workloads are (e.g., what problems are they trying to solve, what algorithms and data structure they are using, ...).
- ii. Discuss how could these workload characteristics affect performance based on the statistics obtained from the simulation result.

B. Static Predictor and Dynamic Predictor Performance Analysis

- i. Compare the system performance between when using a bimodal predictor and using a perfect predictor based on the default config. How much performance gain do you expect if we improve the branch predictor further than the bimodal predictor? Is it large or small? Why is it? Reason your answer with simulation results and analysis.
- ii. Compare the performance of the three dynamic branch predictors available in SimpleScalar (bimodal, 2-level, and combined). Briefly discuss the performance difference between these branch predictors based on their key idea (see textbook 4.8; Dynamic Branch Prediction) and simulation results.

C. The Key Insights in Branch Predictor Design

- i. You may notice that there is a configurable parameter '<xor>' in '-bpred:2lev <l1size> <l2size> <hist_size> <xor>' in file 'config/hw4_default'. Briefly explain and discuss *what* and *why* xor operation in 2-level branch prediction.
- ii. Investigate one of the modern branch predictors (**excluding the six branch predictors in SimpleScalar**), briefly describe its key idea, and discuss when and how it can improve prediction accuracy.

IX. Tips

- ✓ Please read this guideline carefully if you are not familiar with Linux. Most of you can do HW4 without difficulties by following the instructions in this guide.
- ✓ There are some **red-highlighted words** and *italic sentences* in this guideline. They are very important information for this homework. Do not ignore these highlighted contents.
- ✓ If you have any questions, please post them on *KLMS QnA board*.