ECE/CS 570 – High Performance Computer Architecture
Instructor: Prof. Lizhong Chen
Final Project
Due: Tuesday, June 11 at 5:00pm, Online submission

The purpose of this project is to gain understanding of branch prediction and explore its design space. Computer architects in industry use software-based microarchitectural simulation tools to analyze bottlenecks, rapid prototyping of new ideas, and to compare the relative merits of different ideas.

In this project we will use an existing out-of-order superscalar processor simulator called SimpleScalar as the simulation tool. This simulator, like many other processor simulators, reads a machine configuration file and generates a processor model that matches the requirements specified in the configuration file. The configuration file specifies parameters such as the size and associativity of caches, the size of RoB, the number of functional units etc.

Part I. Instructions

1. **Basic info**
    a. Due: Tuesday, 6/11 at 5pm
    b. Up to 3 students per team are allowed (1 or 2 per team is perfectly fine)
    c. Using university servers might not be a good option as “sudo” is not supported on those servers
    d. Late submission is not allowed
    e. Technical questions should be directed to the TA Yongbin Gu
    f. Extra points may be given if your team wins the competition
2. **General Setup Instructions**
    a. System Requirement: The following setup is tested under Ubuntu 12.04. You are encouraged to use Virtual Machines to install Ubuntu 12.04. Other versions might work, but you will be on your own.
    b. After you got Ubuntu 12.04, please install git: “apt-get install git”
    c. Enter “git clone https://github.com/gyb1325/lab_setup”
    d. Enter lab_setup folder, and run “./Install-SimpleScalar.sh”
    e. After executing the script above, the simplescalar and its dependence will automatically be installed.
    f. Change configuration to Alpha by typing the following three commands in a sequence (under the simplesim-3.0 path):
        i. make clean

ii. make config-alpha
iii. make

3. **Try to run test benchmarks with simplescalar**
    a. cd <path to simplesim-3.0>
    b. ./sim-outorder -config config/default.cfg tests/bin.little/test-math
4. **Benchmark Download**
    a. This project will use the following benchmarks
    b. canvas/file/benchmark

Part II. Tasks and Requirements

This project will let you become familiar with branch prediction schemes and their design spaces. You will use "**sim-outorder**" to run different benchmarks. The guidance of the SimpleScalar can be found at link "http://www.simplescalar.com/docs/users_guide_v2.pdf". It will resolve the most of problems during testing. "./sim-outorder -h" can also give you some help information.

Note: Make sure that the simulator is configured as **Alpha** Simulator. Use "**sim-outorder**" to run benchmarks. You will need to modify config/default.cfg file to simulate different configurations.

Reading Materials:
Chapter 3.4.3 "Dynamic branch prediction" in the book "Parallel Computer Organization and Design", as well as other materials that you can/need to find and read.

Benchmarks Requirement:

- The following tasks all need to run 3 benchmarks included in the downloaded benchmark folder: GCC, ANAGRAM (input data: anagram.in), GO (input config: 50 9 2stone9.in)
- The instructions of running benchmarks are included in the benchmark folder
- Use **Alpha** Benchmark

1. **Execute Benchmarks and fill Table 1 with address-prediction rate (see users guide for more details)**

Benchmark	Taken	Not taken	Bimod	2 level	Combined
Anagram					
go					
gcc					

Plot a histogram for all the three benchmarks and draw conclusion. Does your conclusion agree with your intuition?

2. **Execute Benchmarks and fill Table 2 with IPC**

Benchmark	Taken	Not taken	Bimod	2 level	Combined
Anagram					
go					
gcc					

Plot a histogram for all the three benchmarks.

3. **Use bimodal branch predictor, change the bimodal predictor table entry numbers, execute Benchmarks and fill Table 3 with address-prediction rate**

Benchmark	256	512	1024	2048	4096
Anagram					
go					
gcc					

Plot a histogram for all the three benchmarks.

4. **In your report, please describe how combined branch predictor works?**

5. **Find the best two-level branch predictor**
   In SimpleScalar, it can simulate five 2-level branch predictor configurations: GAg, GAp, PAg, PAp, gshare. Your task is to find the best 2-level branch predictor in terms of performance by changing parameters without exceeding storage constraints. You can change only four parameters of each 2-level branch predictor: l1_size, l2_size, hist_size, and XOR. The performance is defined as follow:
   Performance evaluation metric: Average IPC = (IPC_{GCC} + $IPC_{ANAGRAM}$ + IPC_{GO}) / 3
   Please make sure the storage consumption of the predictor (storage consumption =l1_size * hist_szie + l2_size * 2, unit: bits) does not exceed **2 KB**. More details can be found in the user's guide. Designs that violate this constraint will receive 0 point on this part.
   Bonus: The team with the highest performance gets extra 10 points, added to the final total score. The second highest receives 3 extra points.

Part III. Final Report & Submission

Submission:
- Submit electronically by 5pm on 6/11
- Email your final submission (one per team) to both chenliz@oregonstate.edu and guyo@oregonstate.edu with the title “[570-S19] Final Report”. Make sure to cc all the team members in the email.
- The submission includes a pdf report (which needs to include the best 2-level branch predictor that you have found, along with the parameter values) and your final configuration file for task 5 (default.cfg). Your performance/storage results will be verified by the TA by replacing the original configuration file with yours.

Format:
- Use the standard IEEE template (double column):
http://www.ieee.org/conferences_events/conferences/publishing/templates.html
- Report file in pdf (can be from either MS Word or Latex)
For MS Word, use “Page Layout => Hyphenation” for better layout
- Page limitation: mini 2 pages, max 30 pages