[CPEN-411 2023] Assignment 4:
Branch Predictor Implementations

This assignment aims to implement branch prediction policies. The assignment uses an industry-grade simulator, called **ChampSim**, to implement and test these policies. The assignment is divided into two parts. The first part discusses the preparatory steps. The second part discusses 3 different tasks. You will need a unix box, linux box, a virtual machine that runs linux, a linux within windows, etc. **Please read this document very carefully!**

Preparatory Steps for ECE Server

1. **[0 Marks]** Copy **Assignment-4.tar.gz** into the ECE server and untar it.
   $ tar -xvzf **Assignment-4.tar.gz**

```
ssh-linux3:~/CPEN411-2022> ls
Assignment4.tar.gz  ChampSim-Intel
ssh-linux3:~/CPEN411-2022>
```

2. **[0 Marks]** Change directory into **ChampSim-Intel** folder.

```
ssh-linux3:~/CPEN411-2022> ls
Assignment4.tar.gz  ChampSim-Intel
ssh-linux3:~/CPEN411-2022> cd ChampSim-Intel/
ssh-linux3:~/CPEN411-2022/ChampSim-Intel> ls
bin     build_2bitcorr.sh   build_hashed_gselect.sh  create_submitarchive.sh  LICENSE   obj         replacement  simscript  src
branch  build_baselines.sh  compile_champsim.sh      inc                      Makefile  prefetcher  results      spec2006
ssh-linux3:~/CPEN411-2022/ChampSim-Intel>
```

3. **[0 Marks]** Create **five softlinks** of the the compressed benchmarks into the **spec2006 folder**.
   $ ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/400.perlbench.gz spec2006/400.perlbench.gz
   $ ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/401.bzip2.gz spec2006/401.bzip2.gz
   $ ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/403.gcc.gz spec2006/403.gcc.gz
   $ ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/464.h264ref.gz spec2006/464.h264ref.gz
   $ ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/462.libquantum.gz spec2006/462.libquantum.gz
   $ ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/471.omnetpp.gz spec2006/471.omnetpp.gz

   Note that: You can also download these benchmarks from the link below
   https://courses.ece.ubc.ca/cpen411/2023/

4. **[0 Marks]** Change directory into **ChampSim-Intel** folder.
5. **[0 Marks]** Change directory into **branch** folder.
   a. The hashed_perceptron.bpred file implements a “hashed” version of the perceptron branch predictor.
   b. The perceptron.bpred file implements the perceptron branch predictor.

6. **[0 Marks]** Please investigate the .bpred files. You will notice that, while the code is written in C++, these files do not have a .cc or .cpp extension. This is on-purpose and you do not need to worry about it. However, .bpred files must be edited like they are C++ files.
7. **[0 Marks]** Change directory into **ChampSim-Intel** folder.
8. **[0 Marks]** Compile the baseline branch prediction policies.

```
ssh-linux3:~/CPEN411-2022/ChampSim-Intel> bash build_baselines.sh
```

9. **[0 Marks]** Change directory into **simscript** folder.
10. **[0 Marks]** This assignment has six workloads. These six workloads are spec2006 workloads each running on a simulated core. The details of the workloads can be found inside the bench_common.pl file. **DO NOT EDIT THIS FILE in this FOLDER.**
11. **[0 Marks] runall_<name_of_policy>.sh** indicates the different branch prediction policies.

12. **[0 Marks]** Execute **runall_hashed_perceptron.sh**.

```
ssh-linux3:~/CPEN411-2022/ChampSim-Intel> ls
bin     build_2bitcorr.sh   build_hashed_gselect.sh  create_submitarchive.sh  LICENSE   obj         replacement  simscript  src
branch  build_baselines.sh  compile_champsim.sh      inc                      Makefile  prefetcher  results      spec2006
ssh-linux3:~/CPEN411-2022/ChampSim-Intel> cd simscript/
ssh-linux3:~/CPEN411-2022/ChampSim-Intel/simscript> ls
bench_common.pl  getdata.pl  runall_2bitcorr.sh  runall_hashed_gselect.sh  runall_hashed_perceptron.sh  runall_perceptron.sh  runall.pl  run_champsim.sh
ssh-linux3:~/CPEN411-2022/ChampSim-Intel/simscript> bash runall_hashed_perceptron.sh
```

```
ssh-linux3:~/CPEN411-2022/ChampSim-Intel/simscript> bash runall_hashed_perceptron.sh
Name "main::myopt" used only once: possible typo at ./runall.pl line 81.
Name "main::trace_dir" used only once: possible typo at ./runall.pl line 5.
../results/hashed_perceptron/simPN.bin
./run_champsim.sh    hashed_perceptron-no-no-no-no-lru-4core 10 40 0 0 401.bzip2.gz 401.bzip2.gz 401.bzip2.gz 401.bzip2.gz  > ../results/hashed_perceptron/401.bzip2
```

13. **[0 Marks]** Similarly, you can execute **runall_perceptron.sh** at a later time. For this assignment, hashed_perceptron serves as the primary baseline and you need to have its results as a point of comparison.
14. **[0 Marks]** These simulations take quite a lot of time – between 1-2 hours per workload. DO NOT KILL your simulations by shutting down your terminal. Please use tools like "screen" to maintain your simulations on the ece server.
15. **[0 Marks]** You can check your results by changing to the **ChampSim-Intel** folder and then changing into the **results/<name_of_your_policy> folder**. This folder will contain files with the names of your benchmarks. You will also see the names of four mixed workloads. Once your open any file in this folder, say 401.bzip2, the top of the file would mention the workloads that are being executed and the configuration (broadly). **DO NOT EDIT ANY OF THESE FILES.**

```
ssh-linux3:~/CPEN411-2022/ChampSim-Intel/results/hashed_perceptron> cat 401.bzip2

*** ChampSim Multicore Out-of-Order Simulator ***

Warmup Instructions: 10000000
Simulation Instructions: 40000000
Ceviche: 0
Strategy: 0
Number of CPUs: 4
LLC sets: 8192
LLC ways: 8
Off-chip DRAM Size: 16384 MB Channels: 1 Width: 64-bit Data Rate: 800 MT/s

CPU 0 runs /ubc/ece/home/pnj/faculty/prashantnair/CPEN411-2022/ChampSim-Intel/simscript/../spec2006/401.bzip2.gz
CPU 1 runs /ubc/ece/home/pnj/faculty/prashantnair/CPEN411-2022/ChampSim-Intel/simscript/../spec2006/401.bzip2.gz
CPU 2 runs /ubc/ece/home/pnj/faculty/prashantnair/CPEN411-2022/ChampSim-Intel/simscript/../spec2006/401.bzip2.gz
CPU 3 runs /ubc/ece/home/pnj/faculty/prashantnair/CPEN411-2022/ChampSim-Intel/simscript/../spec2006/401.bzip2.gz
Heartbeat CPU 1 instructions: 10000001 cycles: 4088079 heartbeat IPC: 2.44614 cumulative IPC: 2.44614 (Simulation time: 0 hr 5 min 2 sec)
Heartbeat CPU 0 instructions: 10000001 cycles: 4090373 heartbeat IPC: 2.44477 cumulative IPC: 2.44477 (Simulation time: 0 hr 5 min 2 sec)
Heartbeat CPU 3 instructions: 10000001 cycles: 4090456 heartbeat IPC: 2.44472 cumulative IPC: 2.44472 (Simulation time: 0 hr 5 min 2 sec)
Heartbeat CPU 2 instructions: 10000001 cycles: 4090547 heartbeat IPC: 2.44466 cumulative IPC: 2.44466 (Simulation time: 0 hr 5 min 2 sec)

Warmup_CPU0_instructions : 10000322
Warmup_CPU0_cycles : 4090547
 (Simulation time: 0 hr 5 min 2 sec)
Warmup_CPU1_instructions : 10002826
Warmup_CPU1_cycles : 4090547
 (Simulation time: 0 hr 5 min 2 sec)
Warmup_CPU2_instructions : 10000001
Warmup_CPU2_cycles : 4090547
 (Simulation time: 0 hr 5 min 2 sec)
Warmup_CPU3_instructions : 10000279
Warmup_CPU3_cycles : 4090546
 (Simulation time: 0 hr 5 min 2 sec)

Heartbeat CPU 3 instructions: 20000003 cycles: 21442031 heartbeat IPC: 0.576317 cumulative IPC: 0.576304 (Simulation time: 0 hr 9 min 26 sec)
Heartbeat CPU 1 instructions: 20000003 cycles: 21445519 heartbeat IPC: 0.576122 cumulative IPC: 0.576041 (Simulation time: 0 hr 9 min 26 sec)
Heartbeat CPU 0 instructions: 20000003 cycles: 21446036 heartbeat IPC: 0.576181 cumulative IPC: 0.576168 (Simulation time: 0 hr 9 min 26 sec)
Heartbeat CPU 2 instructions: 20000003 cycles: 21447348 heartbeat IPC: 0.576143 cumulative IPC: 0.576143 (Simulation time: 0 hr 9 min 27 sec)
```

16. **[0 Marks]** Once your simulations finish executing, you will need to check and note-down five key variables.
    a. **ROI_Cumulative_IPC**
    b. **CPU_0_Branch Prediction Accuracy**
    c. **CPU_0_MPKIPERCENT**
    d. **CPU_0_Average_ROB_Occupancy_at_Mispredict**

17. **[0 Marks]** Throughout the assignment you are allowed to change only the **.bpred files** in the **branch folder** and the files in the **simscript-testing folder**. You can change the **bench_common.pl** in the **simscript-testing folder** and run these benchmarks for smaller number of instructions.

Evaluated Steps [5 marks]: Branch Prediction Policies

1. **[2 Marks]** Implement a 2-bit correlated branch predictor with a single bit of history. You can edit the **2bitcorr.bpred file** within branch folder to implement this policy. You can compile by using **build_2bitcorr.sh**. You can execute this policy by changing into the simscript directory and executing **runall_2bitcorr.sh**. Your outputs should be available in **results/2bitcorr** folder. You may compare your output files with those of hashed_perceptron.
   Note that: Your branch predictor has a total budget of 16384 entries in a table with each entry being 2 bits in size (max value of 3). You cannot change this restriction; you can slice and dice this table however you wish. We have also given you a "hash" that operates on the "ip". Please index your tables using this "hash". The "prediction" variable is your final prediction: it can either be "taken", i.e. 1 or "not taken" i.e. 0. We have also indicated which part of this file you need to fill (see the comments in the file).

2. **[2 Marks]** Implement a hashed-gselect branch predictor. You can edit the **hashed_gselect.bpred file** within branch folder to implement this policy. You can compile by using **build_hashed_gselect.sh**. You can execute this policy by changing into the simscript directory and executing **runall_hashed_gselect.sh**. Your outputs should be available in **results/hashed_gselect** folder. You may compare your output files with those of hashed_perceptron.
   Note that: Your branch predictor has a total budget of 16384 entries in a table with each entry being 2 bits in size (max value of 3). You cannot change this restriction; you can slice and dice this table however you wish. We have also given you a "hash" that operates on the "ip". Please index your tables using this "hash". Your **hashed-gselect** also uses history. We store history using the "history" variable. For this assignment, please use **5 bits of branch history**. You will need to "XOR" the hash value with the branch history to implement the **hashed-gselect** predictor**.** You will need to use this as the indexing function. The "prediction" variable is your final prediction: it can either be "taken", i.e. 1 or "not taken" i.e. 0. We have also indicated which part of this file you need to fill (see the comments in the file).

3. **[1 Mark]** Write a report in **pdf format** explaining how you implemented these policies. This pdf should also have graphs that compare:
   a. **ROI_Cumulative_IPC**
   b. **CPU_0_Branch Prediction Accuracy**
   c. **CPU_0_MPKIPERCENT**
   d. **CPU_0_Average_ROB_Occupancy_at_Mispredict**

These graphs should include all 5 workloads and they should have executed for the configurations in the original bench_common.pl. You should also compare these – with respect to hashed_perceptron for each workload. You also need to compute their **GEOMEAN Speedup** using the IPC values.

Please place your pdf report in the main ChampSim-Intel folder. The report should have a .pdf extension. If you do not do this, the submission script will not include your report.

Submission

To submit, please execute the following command within **ChampSim-Intel folder.**

```
[ssh-linux3:~/CPEN411-2022/ChampSim-Intel> bash create_submitarchive.sh
```

Please upload **ONLY** the **submission.tar.gz on Canvas**.

Thus, be extremely careful and double check if your results, branch folder, etc. are the right one! We have ~80 students in this course and we will not re-evaluate your Assignment if you submit an incorrect or stale work.

Bonus Points [2 points]

You can earn up to 2 bonus points. For this, you will have to participate in a class-wide competition.

The top 3 students with the highest IPC get 2 bonus points. The next 3 students get one bonus points. To gain these points,

1. You are allowed to create a file called gimmick.bpred in the branch predictor folder
2. You are allowed to edit this file and create your own branch predictor.
3. **Your total budget for this branch predictor is 32K bits**. This budget includes everything in the branch predictor – tables, counters, etc.
4. You are to report your geomean IPC on a google sheet file with a pseudonym. This step is mandatory. The link to the sheet is here: https://docs.google.com/spreadsheets/d/12-ajms21B2kGY-x8mviX4ZdMbnfnT1cEaLFURVbtXEo/edit?usp=sharing
5. You attach **ONLY your gimmick.bpred file** in the bonus part of assignment. During submission, **as a comment on Canvas, please write your pseudonym and the IPC**
6. You will be given four extra days to do this task.
7. We will only evaluate the top 6 candidates. **This is done manually as we need to check if you followed the 32K bit budget**. So please report accurate numbers in the google sheet and update these numbers as you change your design.
8. **If your pseudonym reports an incorrect IPC on sheets and/or your submission also contains the IPC that is incorrect, you will be given a 0 for the entire assignment! We cannot simply evaluate your gimmick.bpred file if you have incorrectly reported your numbers.**