[CPEN-411 2023] Assignment 2:
High-Performance Cache Replacement Policies

This assignment aims to implement high-performance and industry-grade cache-replacement policies for the **shared Last Level Cache** (LLC). 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 5 different tasks and a bonus task – for 1 point. You may use a unix box, linux box, a virtual machine that runs linux, a linux within windows, etc. Additionally, this document enables you to execute your code on the ECE server. **Please read this document very carefully!**

Preparatory Steps

1. **[0 Marks]** Copy the Assignment-2.tar.gz folder to the ECE Server. If you like, you may create a folder named CPEN411 within the ECE server and then copy your tar.gz file there.

2. **[0 Marks]** untar **Assignment-2.tar.gz**
   $ tar xvf **Assignment-2.tar.gz**

```
gattaca:~/CPEN411> ls
Assignment-2.tar.gz  ChampSim-Intel
gattaca:~/CPEN411>
```

3. **[0 Marks]** Change directory into **ChampSim-Intel** folder.

```
gattaca:~/CPEN411> cd ChampSim-Intel/
gattaca:~/CPEN411/ChampSim-Intel> ls
bin     build_all.sh        build_bip.sh  build_gimmick.sh  build_ptreelru.sh    create_submitarchive.sh  LICENSE   obj         replacement  simscript  src
branch  build_baselines.sh  build_dip.sh  build_lip.sh      compile_champsim.sh  inc                      Makefile  prefetcher  results      spec2006
gattaca:~/CPEN411/ChampSim-Intel>
```

4. **[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/2023/462.libquantum.gz spec2006/462.libquantum.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/2023/471.omnetpp.gz spec2006/471.omnetpp.gz

```
ssh-linux4:~/CPEN411/ChampSim-Intel> ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/400.perlbench.gz spec2006/400.perlbench.gz
ssh-linux4:~/CPEN411/ChampSim-Intel> ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/401.bzip2.gz spec2006/401.bzip2.gz
ssh-linux4:~/CPEN411/ChampSim-Intel> ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/403.gcc.gz spec2006/403.gcc.gz
ssh-linux4:~/CPEN411/ChampSim-Intel> ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/462.libquantum.gz spec2006/462.libquantum.gz
ssh-linux4:~/CPEN411/ChampSim-Intel> ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/464.h264ref.gz spec2006/464.h264ref.gz
ssh-linux4:~/CPEN411/ChampSim-Intel> ln -s /ubc/ece/home/courses/cpen411/etc/www/2023/471.omnetpp.gz spec2006/471.omnetpp.gz
ssh-linux4:~/CPEN411/ChampSim-Intel>
```

   This should create "soft" links to these files. This would also enable you to compile and run your source code on the ece-server.

5. **[0 Marks]** Change directory into **ChampSim-Intel** folder.

```
gattaca:~/CPEN411/ChampSim-Intel> ls
bin     build_all.sh        build_bip.sh  build_gimmick.sh  build_ptreelru.sh    create_submitarchive.sh  LICENSE   obj         replacement  simscript  src
branch  build_baselines.sh  build_dip.sh  build_lip.sh      compile_champsim.sh  inc                      Makefile  prefetcher  results      spec2006
gattaca:~/CPEN411/ChampSim-Intel>
```

6. **[0 Marks]** Change directory into **replacement** folder.

```
gattaca:~/CPEN411/ChampSim-Intel> ls
bin     build_all.sh        build_bip.sh  build_gimmick.sh  build_ptreelru.sh    create_submitarchive.sh  LICENSE   obj         replacement  simscript  src
branch  build_baselines.sh  build_dip.sh  build_lip.sh      compile_champsim.sh  inc                      Makefile  prefetcher  results      spec2006
gattaca:~/CPEN411/ChampSim-Intel> cd replacement/
gattaca:~/CPEN411/ChampSim-Intel/replacement> ls
base_replacement.cc  bip.llc_repl  dip.llc_repl  drrip.llc_repl  gimmick.llc_repl  hawkeye.llc_repl  lip.llc_repl  llc_replacement.cc  lru.llc_repl  ptreelru.llc_repl  ship.llc_repl  srrip.llc_repl
gattaca:~/CPEN411/ChampSim-Intel/replacement>
```

a. The lru.llc_repl file implements the LRU policy.
b. The srrip.llc_repl file implements the SRRIP policy
c. The drrip.llc_repl file implements the DRRIP policy.
d. The ship.llc_repl file implements the SHIP policy.
e. The hawkeye.llc_repl file implements the HAWKEYE policy [1].

[1] *Akanksha Jain and Calvin Lin. 2016. Back to the future: Leveraging Belady's algorithm for improved cache replacement. In Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16). IEEE Press, 78–89. https://doi.org/10.1109/ISCA.2016.17*

7. **[0 Marks]** Please investigate the .repl 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, .repl files must be edited like they are C++ files.

8. **[0 Marks]** Change directory into **ChampSim-Intel** folder.

```
gattaca:~/CPEN411/ChampSim-Intel> ls
bin     build_all.sh        build_bip.sh  build_gimmick.sh  build_ptreelru.sh    create_submitarchive.sh  LICENSE   obj         replacement  simscript  src
branch  build_baselines.sh  build_dip.sh  build_lip.sh      compile_champsim.sh  inc                      Makefile  prefetcher  results      spec2006
gattaca:~/CPEN411/ChampSim-Intel>
```

9. **[0 Marks]** Compile these "sample" baseline cache-replacement policies.

```
gattaca:~/CPEN411/ChampSim-Intel> ls
bin     build_baselines.sh  build_dip.sh      build_lip.sh       compile_champsim.sh      inc      Makefile  prefetcher   results    spec2006
branch  build_bip.sh        build_gimmick.sh  build_ptreelru.sh  create_submitarchive.sh  LICENSE  obj       replacement  simscript  src
gattaca:~/CPEN411/ChampSim-Intel> bash build_baselines.sh
```

10. **[0 Marks]** Change directory into **simscript** folder.

```
gattaca:~/CPEN411/ChampSim-Intel> ls
bin     build_baselines.sh  build_dip.sh      build_lip.sh       compile_champsim.sh      inc      Makefile  prefetcher   results    spec2006
branch  build_bip.sh        build_gimmick.sh  build_ptreelru.sh  create_submitarchive.sh  LICENSE  obj       replacement  simscript  src
gattaca:~/CPEN411/ChampSim-Intel> cd simscript/
gattaca:~/CPEN411/ChampSim-Intel/simscript> ls
bench_common.pl  runall_bip.sh  runall_drrip.sh    runall_hawkeye.sh  runall_lru.sh  runall_ptreelru.sh  runall_srrip.sh
getdata.pl       runall_dip.sh  runall_gimmick.sh  runall_lip.sh      runall.pl      runall_ship.sh      run_champsim.sh
gattaca:~/CPEN411/ChampSim-Intel/simscript>
```

11. **[0 Marks]** This assignment has six SPEC 2006 workloads. Each instance of the workload runs on a core. Our simulation setup is a 4-core CPU with a **shared LLC**. The details of the workloads can be found inside the bench_common.pl file. **DO NOT EDIT THIS FILE in this FOLDER.**

```
gattaca:~/CPEN411/ChampSim-Intel/simscript> ls
bench_common.pl  runall_bip.sh  runall_drrip.sh    runall_hawkeye.sh  runall_lru.sh  runall_ptreelru.sh  runall_srrip.sh
getdata.pl       runall_dip.sh  runall_gimmick.sh  runall_lip.sh      runall.pl      runall_ship.sh      run_champsim.sh
gattaca:~/CPEN411/ChampSim-Intel/simscript> vim bench_common.pl
```

```
#*****************************SPEC NAMES **********************************
$SUITES{'spec_name'}      =
'400.perlbench
401.bzip2
403.gcc
462.libquantum
464.h264ref
471.omnetpp';
```

12. **[0 Marks] runall_<name_of_policy>.sh** indicates the different cache-replacement policies. Open one of these files, e.g. **runall_lru.sh**

The –f 1 indicates that 1 simulation will be fired in parallel as separate processes running in the background. This is the setting for ECE server. **If you do have a powerful local computer with several processing cores, you can change this "f" value to be > 1.**

```
ssh-linux4:~/CPEN411/ChampSim-Intel/simscript> ls
bench_common.pl  runall_bip.sh  runall_drrip.sh    runall_hawkeye.sh  runall_lru.sh  runall_ptreelru.sh  runall_srrip.sh
getdata.pl       runall_dip.sh  runall_gimmick.sh  runall_lip.sh      runall.pl      runall_ship.sh      run_champsim.sh
ssh-linux4:~/CPEN411/ChampSim-Intel/simscript>
```

13. **[0 Marks]** You will have to use "**screen**" to make sure your simulations do not get killed when you log off ece-servers. "screen" is already installed on the ece-servers.

    To understand how screen works, see: https://www.youtube.com/watch?v=I4xVn6Io5Nw

14. **[0 Marks]** Execute **runall_lru.sh**.

```
gattaca:~/CPEN411/ChampSim-Intel/simscript> bash runall_lru.sh
Name "main::trace_dir" used only once: possible typo at ./runall.pl line 5.
Name "main::myopt" used only once: possible typo at ./runall.pl line 81.
../results/lru/simPN.bin
./run_champsim.sh    hashed_perceptron-no-no-no-no-lru-4core 50 200 0 0 401.bzip2.gz 401.bzip2.gz 401.bzip2.gz 401.bzip2.gz  > ../results/lru/401.bzip2 &
./run_champsim.sh    hashed_perceptron-no-no-no-no-lru-4core 50 200 0 0 403.gcc.gz 403.gcc.gz 403.gcc.gz 403.gcc.gz  > ../results/lru/403.gcc &
./run_champsim.sh    hashed_perceptron-no-no-no-no-lru-4core 50 200 0 0 462.libquantum.gz 462.libquantum.gz 462.libquantum.gz 462.libquantum.gz  > ../results/lru/462.libquantum &
./run_champsim.sh    hashed_perceptron-no-no-no-no-lru-4core 50 200 0 0 429.mcf.gz 429.mcf.gz 429.mcf.gz 429.mcf.gz  > ../results/lru/429.mcf &
./run_champsim.sh    hashed_perceptron-no-no-no-no-lru-4core 50 200 0 0 400.perlbench.gz 400.perlbench.gz 400.perlbench.gz 400.perlbench.gz  > ../results/lru/400.perlbench &
./run_champsim.sh    hashed_perceptron-no-no-no-no-lru-4core 50 200 0 0 401.bzip2.gz 403.gcc.gz 462.libquantum.gz 429.mcf.gz  > ../results/lru/mix1 &
./run_champsim.sh    hashed_perceptron-no-no-no-no-lru-4core 50 200 0 0 403.gcc.gz 462.libquantum.gz 429.mcf.gz 400.perlbench.gz  > ../results/lru/mix2 &
./run_champsim.sh    hashed_perceptron-no-no-no-no-lru-4core 50 200 0 0 462.libquantum.gz 429.mcf.gz 400.perlbench.gz 401.bzip2.gz  > ../results/lru/mix3 &
./run_champsim.sh    hashed_perceptron-no-no-no-no-lru-4core 50 200 0 0 429.mcf.gz 400.perlbench.gz 401.bzip2.gz 403.gcc.gz  > ../results/lru/mix4
```

15. **[0 Marks]** Similarly, you can execute **runall_srrip.sh**, **runall_drrip.sh, runall_ship.sh, and runall_hawkeye.sh** at a later time. For this assignment, LRU serves as the primary baseline and you need to have its results as a point of comparison. The other sample policies are only FYI and used only during the bonus point question.

16. **[0 Marks]** These simulations take quite a lot of time. DO NOT KILL your simulations before they complete.

17. **[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.**

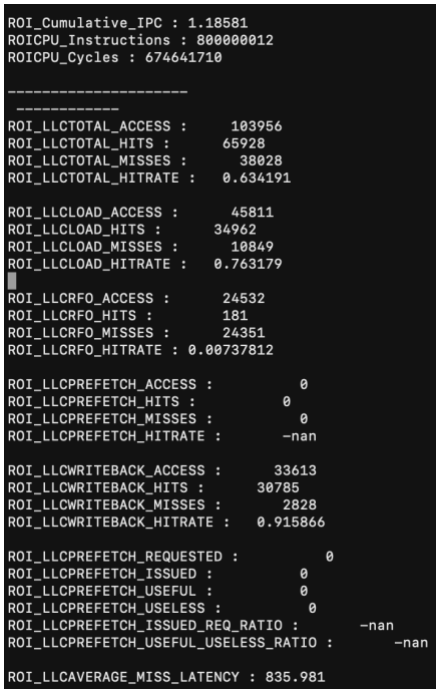
```
*** ChampSim Multicore Out-of-Order Simulator ***

Warmup Instructions: 50000000
Simulation Instructions: 200000000
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 /home/prashantnair/CPEN411/ChampSim-Intel/simscript/../spec2006/401.bzip2.gz
CPU 1 runs /home/prashantnair/CPEN411/ChampSim-Intel/simscript/../spec2006/401.bzip2.gz
CPU 2 runs /home/prashantnair/CPEN411/ChampSim-Intel/simscript/../spec2006/401.bzip2.gz
CPU 3 runs /home/prashantnair/CPEN411/ChampSim-Intel/simscript/../spec2006/401.bzip2.gz
Heartbeat CPU 1 instructions: 10000001 cycles: 4088234 heartbeat IPC: 2.44604 cumulative IPC: 2.44604 (Simulation time: 0 hr 12 min 40 sec)
Heartbeat CPU 2 instructions: 10000001 cycles: 4090622 heartbeat IPC: 2.44462 cumulative IPC: 2.44462 (Simulation time: 0 hr 12 min 40 sec)
Heartbeat CPU 0 instructions: 10000001 cycles: 4090641 heartbeat IPC: 2.4446 cumulative IPC: 2.4446 (Simulation time: 0 hr 12 min 40 sec)
Heartbeat CPU 3 instructions: 10000001 cycles: 4090672 heartbeat IPC: 2.44459 cumulative IPC: 2.44459 (Simulation time: 0 hr 12 min 40 sec)
```

18. **[0 Marks]** Once your simulations finish executing, you will need to check and note-down five key variables.
    a. **ROI_Cumulative_IPC**
    b. **ROI_LLCTOTAL_HITRATE**
    c. **ROI_LLCLOAD_HITRATE**
    d. **ROI_LLCWRITEBACK_HITRATE**
    e. **ROI_LLCAVERAGE_MISS_LATENCY**

    Note: These variables **should not be confused** with **ROI_0Cumulative_IPC, ROI_1Cumulative_IPC, etc. in the output files withi your results folder.** These variables are per-core variables whereas **ROI_Cumulative_IPC (without a number after ROI)** is an aggregate variable. Follow this logic for other variables in this list as well. We are interested ONLY in the ROI statistics.



19. **[0 Marks]** Throughout the assignment you are allowed to change only the **.repl files** in the **replacement 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. This will help you develop your algorithm quicker. You can also change the **runall_<name of your policy>.sh** files in this folder and change the **–f values**.

Evaluated Steps [5 marks]: Cache Replacement Policies

These will take you FULL 2 weeks. Please start this TODAY ITSELF!

1. **[1 Mark]** Implement the LRU Insertion Policy (LIP). You can edit the **lip_llc.repl file** within replacement folder to implement this policy. You can compile by using **build_lip.sh** (similar to step-8 in the preparatory phase). You can execute this policy by changing into the simscript

directory and executing **runall_lip.sh** (similar to step-12 in the preparatory phase). Your outputs should be available in **results/lip** folder. You may compare your output files with those of LRU in the same folder. You can compare against the metrics mentioned in step-16 of the preparatory phase.

2. **[1 Mark]** Implement the Bimodal Insertion Policy (BIP) with an *epsilon (∈) of 5%*. You can edit the **bip_llc.repl file** within replacement folder to implement this policy. You can compile by using **build_bip.sh** (similar to step-8 in the preparatory phase). You can execute this policy by changing into the simscript directory and executing **runall_bip.sh** (similar to step-12 in the preparatory phase). Your outputs should be available in **results/bip** folder. You may compare your output files with those of LRU in the same folder. You can compare against the metrics mentioned in step-16 of the preparatory phase.
3. **[1 Mark]** Implement the Dynamic Insertion Policy with BIP (*epsilon (∈) of 5%)* and LRU. To implement set-duelling and set-sampling, you may choose **64 sample sets for each policy**. You can edit the **dip_llc.repl file** within replacement folder to implement this policy. You can compile by using **build_dip.sh** (similar to step-8 in the preparatory phase). You can execute this policy by changing into the simscript directory and executing **runall_dip.sh** (similar to step-12 in the preparatory phase). Your outputs should be available in **results/dip** folder. You may compare your output files with those of LRU in the same folder. You can compare against the metrics mentioned in step-16 of the preparatory phase.

4. **[1 Mark]** Implement the Tree-Based Pseudo-LRU eviction policy. You can edit the **ptreelru_llc.repl file** within replacement folder to implement this policy. You can compile by using **build_ptreelru.sh** (similar to step-8 in the preparatory phase). You can execute this policy by changing into the simscript directory and executing **runall_ptreelru.sh** (similar to step-12 in the preparatory phase). Your outputs should be available in **results/ptreelru** folder. You may compare your output files with those of LRU in the same folder. You can compare against the metrics mentioned in step-16 of the preparatory phase.

5. **[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. **ROI_LLCTOTAL_HITRATE**
    c. **ROI_LLCLOAD_HITRATE**
    d. **ROI_LLCWRITEBACK_HITRATE**
    e. **ROI_LLCAVERAGE_MISS_LATENCY**

These graphs should include all 9 workloads and they should have executed for the configurations in the original bench_common.pl. You should also compare the IPC – with respect to the IPC of LRU for each workload and then compute their **GEOMEAN Speedup**.

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.

BONUS [1 mark]: YOUR Cache Replacement Policy called GIMMICK

6. Implement your own cache replacement policy – let us call this policy GIMMICK. You can edit the **gimmick_llc.repl file** within replacement folder to implement this policy. You can compile by using **build_gimmick.sh** (similar to step-8 in the preparatory phase). You can execute this policy by changing into the simscript directory and executing **runall_gimmick.sh** (similar to step-12 in the preparatory phase). Your outputs should be available in **results/gimmick** folder. You may compare your output files with those of LRU in the same folder. You can compare against the metrics mentioned in step-16 of the preparatory phase. **To get this BONUS point, GIMMICK should have 2%+ speedup (geomean) as compared to Hawkeye.**

7. **If you have implemented GIMMICK and it provides 2%+ speedup (geomean) as compared to Hawkeye ONLY then –** In your report (pdf) explain how you implemented GIMMICK. This pdf should also have graphs for GIMMICK and Hawkeye and you need to compare:
    a. **ROI_Cumulative_IPC**
    b. **ROI_LLCTOTAL_HITRATE**
    c. **ROI_LLCLOAD_HITRATE**
    d. **ROI_LLCWRITEBACK_HITRATE**
    e. **ROI_LLCAVERAGE_MISS_LATENCY**

These graphs should include all 9 workloads and they should have executed for the configurations in the original bench_common.pl. You should also compare the IPC of GIMMICK – with respect to the IPC of Hawkeye for each workload and then compute their **GEOMEAN Speedup**.

Submission

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

```
gattaca:~/CPEN411/ChampSim-Intel> bash create_submitarchive.sh
```

Please upload **ONLY** the **submission.tar.gz on Canvas**. If you have implemented GIMMICK which provides 2%+ speedup over Hawkeye, mention this in the comment section of your submission.

Thus, be extremely careful and double check if your results, replacement folders, etc. are the right one! We have ~90 students in this course and we will not re-evaluate your Assignment if you submit an incorrect or stale work.