Project 4. Cache Simulator

Due: 23:59, June. 4th

TA: Yoonsung Kim, Guseul Heo

1. Introduction

In this project, you will build **a trace-based cache simulator**. This cache simulator is configurable to adjust capacity, associativity (i.e., n-way), and block size with command-line options. **Please read this document carefully before you start.**

For any questions related to the project, please ask on the Piazza board or via email (cs311_ta@casys.kaist.ac.kr).

Additionally, we will run 2 office hours for this project:

- **1 week before the deadline: Thursday, May 30th, 2024, 7pm ~ 9pm**
- **1 day before the deadline: Monday, June 3rd, 2024, 7pm ~ 9pm**

@ E3-1 4th Floor, Room 4429.

2. Simulator Specification

The cache simulator receives a trace file as input. This trace file contains a sequence of read and write operations along with their addresses. At each step, an entry from the trace is fed into the cache simulator to simulate the internal processes of the cache. The cache's write policy must be set to **write-allocate** and **write-back.**

Cache Parameters
The cache simulator has three configurable parameters: capacity, associativity (n-way), and block size. The range of each parameter is specified below.

- Capacity: 4B (one word) ~ 8KB
- Associativity: 1 way ~ 16 ways
- Block size: 4B ~ 32B

Capacity and block size should be specified with byte granularity, and all three parameters must be powers of two (e.g., capacity: 4KB, associativity: 4-ways, block size: 32B, which corresponds to 4096:4:32).

Replacement Policy
You should implement two replacement policies and enable the selection of a replacement policy through a command parameter:
1. LRU (Least-Recently Used): The LRU policy was covered in the CS311 lecture. Please refer to the lecture notes for details on the LRU policy.
2. LFLRU (Least-Frequently Least-Recently Used): The LFLRU policy combines the Least-Frequently Used (LFU) and Least-Recently Used (LRU) policies. Under this policy, when a cache miss occurs and a cache block needs to be evicted, the block with the lowest frequency of access is selected first. If multiple blocks have the same lowest frequency, the least recently used block among them is then evicted. This approach requires tracking both (1) the frequency of access and (2) the most recent access time for each cache block.

Input Trace File
An input trace file contains a sequence of read or write operations with addresses like below.

```
…
R 0x10000000
W 0x10003231
R 0x12341245
R 0x10003231
W 0x10023414
…
```

*R denotes ‘read’, while W means ‘write’.

Output
The simulator output includes statistics for the total number of instructions, write-backs, hits, and misses for both read and write operation. Please check the variables corresponding to these metrics in the skeleton code.

3. Simulator Options

Basic command

$ ./cs311cache -c capacity:associativity:block_size -r -x <input trace file>

Options

- c: Cache configuration specification.
- r: This optional flag enables ‘LRU only replacement policy’. Without this flag option, the simulator runs with the LFLRU policy.
- x: This optional flag dumps the ‘cache content’ at the end of the simulation.

Cache Content

Cache content does not represent actual data, but rather the address, which is aligned with the block size. For example, assume the block size is 16B (2^4B). When the address 0x10001234 is stored in the cache, the content of the cache entry is 0x10001230, as the block size bits (the lowest 4 bits) are masked by 0.

Tag bit	Index bit	Block offset (0)

4. GitLab Repository

Refer to the instructions of the Project 1, changing the link to the TA's repository of **Project 4** (https://cs311.kaist.ac.kr/root/project4). Please make sure that you do a **private fork**.

5. Grading Policy

The grading policy is similar to that of previous projects. Submissions will be evaluated based on 7 examples: **4 provided examples in the 'sample_input' directory and 3 hidden examples, for which both inputs and outputs are not provided.** The hidden examples are of similar complexity to the given examples. **You will receive the full score for each case when your outputs are correct for both policies.**

To get scores, your simulator should print the exactly same output as the reference solution. You can evaluate your code within given examples by comparing your output with files in 'sample_output' directory. **You should check the correctness of your code by executing** make test **at your working directory of this project**.

If there are any differences (including whitespaces) it will print the differences. If there is no difference for an example, it will print 'Test seems correct'.

Please make sure your outputs for given examples are **identical to the files in the sample_output directory, without any redundant prints. Every single character of the output must be identical** to the given sample output. Otherwise, you will receive **0 score** for the example.

Note that **simply hard-coding outputs for given examples would lead you to 0 score** for this project.

6. Submission (Important!!)

Make sure your code works well on our class Linux server. It is highly recommended to work on the class server, since your project will be graded on the same environment as those servers.

Add the 'submit' tag to your final commit and push your work to the gitlab server. The following commands are the flow you should take to submit your work.

If there is no 'submit' tag, your work will not be graded. Please do not forget to submit your work with the tag.

Please make sure you push your work before the deadline. If you do not "push" your work, we won't be able to see your work so that your work will be marked as unsubmitted.

For more information about submission procedure, please refer to the specification of 'Submission' in project 1 document.

7. Late Policy & Plagiarism Penalty (Important!!)

Submissions that are **one day late** (June 5th, 00:00~23:59) will lose **30%** of your score. We will **not accept** any works that are submitted after then.

Be aware of plagiarism! Although it is encouraged to discuss with others and refer to extra materials, **copying other students' or opened code is strictly banned: Not only for main routine functions, but also helper functions.**

TAs will compare your source code with open-source codes and other students' code. If you are caught, you will receive a serious penalty for plagiarism as announced in the first lecture of this course.

If you have any requests or questions regarding administrative issues (such as late submission due to an unfortunate accident, GitLab is not working), please send an email to TAs.

8. Tips

Modifying your repository via http may cause some problems.

You may use C++ instead of C if you want. Just make sure your `make test` command works properly.

In this project, you can freely define **‘additional’** functions, macros, structs and classes (C++) you need. However, **please do not modify existing code lines in the template.**

Please start as early as possible.

Please read this document and given skeleton code carefully before you start.