SCE212 Project 2: Cache Design

Due 11:59pm, December 6th

1. Overview

This project is intended to help you understand the principle of caching by implementing a data cache. The main part of this project is to simulate a data cache from an input trace file. The cache should be configurable to adjust capacity, associativity, and block size with command-line options. You must support extra options to print out the content of the cache.

2. Simulation Details

In this project, you design a trace-based cache simulator. An input trace file contains a sequence of read or write operations with addresses. For each step, an entry of the trace is fed into the cache simulator, to simulate the internal operation of the cache. The write policy of the cache must be *write-allocate* and *write-back*. The replacement policy must be the perfect LRU.

2.1 Cache Parameters

The capacity, associativity, and block size of the cache must be configurable. The parameters are specified with '`-c`' option: `-c <capacity>:<assoc>:<blocksize>`
Configurable parameters:

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

When you specify both capacity and block size, you should specify the number with byte granularity and power of two.
e.g.,) Capacity 4KB, Associativity 4way, Block size 32B → **4096:4:32**

3. Simulator Options and Output

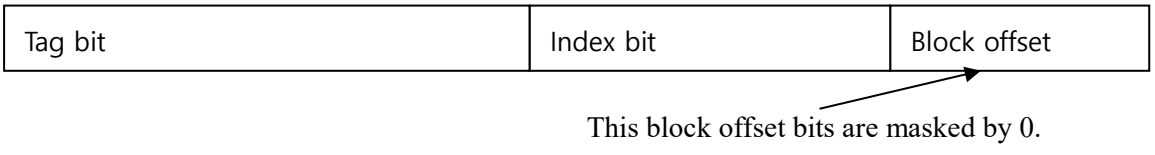
3.1 Options

```
$ ./sce212cache -c cap:assoc:block_size [-x] input_trace
```

- `-c` : cache configuration
- `-x` : dump the cache content at the end of simulation

Cache content is not data content but the address which is aligned with block size.
e.g.,) Block size 16B

When the address 0x10001234 is stored in cache, the content of the cache entry is 0x10001230.
The 4bits (block size bit) are masked by 0.



The TAs will provide skeleton code for displaying output format within `main.c`. You are free to change the parameters and corresponding parts of the code as long as it can print out the same format.

The example of output format with options `-c` and `-x` is attached as below.

```
Cache Configuration:
-------------------------------------
Capacity: 256B
Associativity: 4way
Block Size: 8B

Cache Stat:
-------------------------------------
Total reads: 0
Total writes: 0
Write-backs: 0
Read hits: 0
Write hits: 0
Read misses: 0
Write misses: 0

Cache Content:
-------------------------------------
          WAY[0]      WAY[1]      WAY[2]      WAY[3]
SET[0]:   0x00000000  0x00000000  0x00000000  0x00000000
SET[1]:   0x00000000  0x00000000  0x00000000  0x00000000
SET[2]:   0x00000000  0x00000000  0x00000000  0x00000000
SET[3]:   0x00000000  0x00000000  0x00000000  0x00000000
SET[4]:   0x00000000  0x00000000  0x00000000  0x00000000
SET[5]:   0x00000000  0x00000000  0x00000000  0x00000000
SET[6]:   0x00000000  0x00000000  0x00000000  0x00000000
SET[7]:   0x00000000  0x00000000  0x00000000  0x00000000
```

3.2 Input

The input trace contains a sequence of read or write operations with addresses as we mentioned.

e.g.,)

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

The TAs will provide several input traces which are real world traces.

3.3 Output

Your output must contain the following statistics:

- the number of total reads

- the number of total writes
- the number of write-backs
- the number of read hits
- the number of write hits
- the number of read misses
- the number of write misses

The order of output results is Cache Configuration (if `-c` option is on) → Cache Statistics → Cache Content (if `-x` option is on). The example of output format are attached as below.

4. Forking and Cloning your Repository

Like earlier Project 1, you will fork the SCE212/Project2 repo to your student ID namespace. Then you will clone your repo into your local machines to work on the project. Following instruction is identical to the Project1 and 2.

3.1 Forking the Class's repo

(1) Go to the following page: http://sce212.ajou.ac.kr/[Class ID]/project2. The page is your class repository.
(2) Click the fork button.
(3) Select your account and the repo will be forked.
(4) Your repo will have the following http://sce212.ajou.ac.kr /[your student ID]/project2
(5) NOTE: We will be running automated scripts to download your work and grade your projects. **Please do not change the name of your project paths**. (Keep the project name & path as `project2`)

3.2 Cloning your repository to your local machine (your virtual machine)

From the website of your repo, you can copy the HTTP URL of the git repository. The HTTP URL will look something like the following:

```
http://sce212.ajou.ac.kr/[your student ID]/project2.git
```

Change directory to the location you want to clone your project and clone!

```
$ git clone http://sce212.ajou.ac.kr/[your student ID]/project2.git
```

The second command (HTTP) requires below entries when you type

Username for 'http://sce212.ajou.ac.kr': <**your student ID**>
Password for 'http://[your student ID]@sce212.ajou.ac.kr': <**your password**>

Be sure to read the `README.md` file for some useful information. It includes the explanation of each file and which files you are allowed to modify for this project.

5. Grading Policy

Grades will be given based on the 4 examples provided for this project provided in the `sample_input` directory. Your simulator should print the exact same output as the files in the `sample_output` directory.

We will be automating the grading procedure by seeing if there are any difference between the files in the `sample_output` directory and the result of your simulator executions. **Please make sure that**

your outputs are identical to the files in the sample_output directory.

You are encouraged to use the *`diff`* command to compare your outputs to the provided outputs.

```
$ ./sce212cache -c 1024:8:8 sample_input/simple > my_output
$ diff -Naur my_output sample_output/simple
```

If there are any differences (including whitespaces) the diff program will print the different lines. If there are no differences, nothing will be printed. Furthermore, we have provided a simple checking mechanism in the `Makefile`. Executing the following command will automate the checking procedure.

```
$ make test
```

There are 4 code segments to be graded and you will be granted 20 points for each correct binary code and **being "Correct" means that every digit and location is the same** to the given output of the example. If a digit is not the same, you will receive **0 score** for the example.

6. Submission (Important!!)

6.1 Make sure your code works well on the virtual machine environment we provided

In fact, it is highly recommended to work on the pre-created VM image from TA throughout this class. Your project will be graded on the same environment as the VM.

6.2 Summarize the contribution

You need to summarize your contributions to each project. Add a 'contribution.txt' file in your repository (don't forget to commit it). If you use good commit messages, this can be done in a simple step. **`git shortlog`** summarizes commit titles by each user and will come in handy (especially if your commit titles have useful information).

```
$ git shortlog > contribution.txt
$ git add contribution.txt
$ git commit
```

If you want to add commit messages, please fill in the part after the option '–m' when committing.

```
$ git commit
```

A text editor will pop up with some information about the commit. Fill out your commit message at the top. The first line is the subject line of the commit. The second line should be blank, the third line and onwards will be the body of your commit message.

6.3 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.

```
$ git tag submit
$ git push
$ git push --tags
```

If there is no "`submit`" tag, your work will not be graded so please remember to submit your work with the tag. If you do not `push` your work, we will not have the visibility to your work. Please make sure you push your work before the deadline

6.4 Updating Your Submit Tag

If you decide after tagging your commit and pushing, that you want to update your submission, you will need to remove the existing tag and retag & repush your submit tag.

```
$ git tag -d submit          # Deletes the existing tag
$ git push origin :submit    # Removes the 'submit' tag on the server
```

Now you may re-tag your work and submit using the instruction in Section 6.3

7. Updates/Announcements

If there are any updates to the project, including additional tools/inputs/outputs, or changes, we will post a notice on the Ajou BB, and will send you an e-mail using the Ajou BB system. **Frequently check your AjouBB linked e-mail account or the AjouBB notice board for updates.**