CS311 Project 4: Cache Design

Due 11:59pm, December 22nd

1. Purpose

This project is intended to help you understand the principle of caching by implementing a data cache.

2. Overview

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.

3. Cache Implementation

3.1 Standalone Data Cache Simulation

In this project, you must 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 to 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.

3.2 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.
Ex) Capacity 4KB, Associativity 4way, Block size 32B → **4096:4:32**

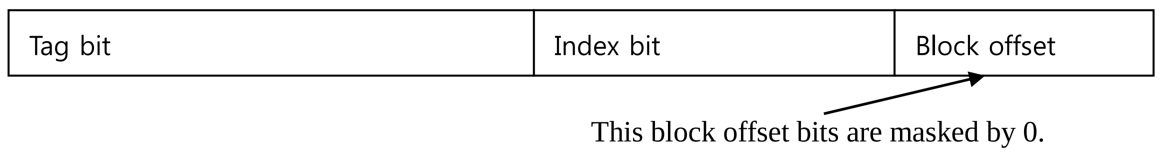
4. Simulator Option

```
cs311cache [-c cap:assoc:bsize] [-x] input_trace
```

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

Cache content is not data content but the address which is aligned with block size.
Ex) 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.



4.1 Input

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

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

The TAs will provide several input traces which are both real world traces and the traces which we generate in order to check the cache operations.

4.2 Output

Your output must contain the following statistics:

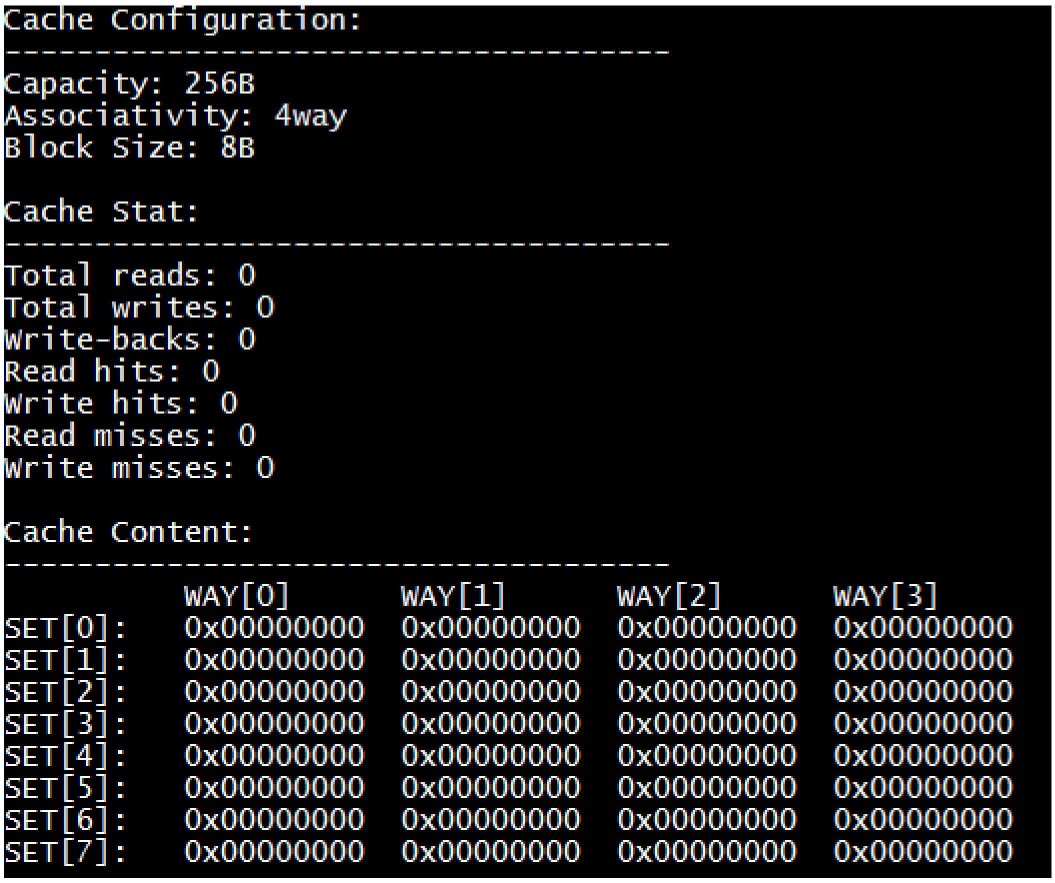
- the number of total reads
- the number of total writes
- the number of write-backs
- the number of read hits
- the number of write hits
- the number of read misses
- the number of write misses

The order of output results is Cache Configuration → Cache Statistics → Cache Content (if –x option is on).

The TAs will provide the skeleton code for displaying output format like lab3.
The example of output format are attached as below.

5. Hand in

You should submit the compressed file of the source files through tar or zip program.
Please make the compressed file name with your team name. (team_name.tar or team_name.zip)
Ex) If you are team14, you should make the compressed file with "team14.tar" or "team14.zip"
Please send the email to cs311_ta@calab.kaist.ac.kr with attaching the compressed file until the due date.



```
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
```

Figure 1 Output format at the initial state