Homework #4: Cache

November 6th, 2019

Jae W. Lee (jaewlee@snu.ac.kr)
Computer Science and Engineering
Seoul National University

TA: Yunho Jin, Jeonghun Gong

Goal of This Project

You will be asked to write a cache simulator

- Understand cache structure
- Implement cache policy

Setup

You need Linux environment
Install valgrind tool for extracting memory traces

```
linux> sudo apt-get install valgrind
```

Download PA4.tar from eTL

- `linux> tar -xvf PA4.tar`
- `linux> cd PA4`
- `linux> make clean; make`

What You Need to Do

Your Task: Implement a cache simulator producing the same output as the reference simulator (=`csim-ref`)

Reference cache simulator can simulate a cache with

- Arbitrary cache size
- Arbitrary associativity
- LRU replacement policy

File to modify (and submit): `csim.c`

Test your code: An Example

```
yunho@yunho-XPS-8700:~/PA4$ make clean; make
rm -rf *.o
rm -f *.tar
rm -f csim
rm -f .csim_results .marker
gcc -g -Wall -Werror -std=c99 -m64 -o csim csim.c cachelab.c -lm
yunho@yunho-XPS-8700:~/PA4$ ./test-csim
                        Your simulator     Reference simulator
Points (s,E,b)    Hits  Misses  Evicts    Hits  Misses  Evicts
     0 (1,1,1)       0       0       0       9       8       6  traces/yi2.trace
     0 (4,2,4)       0       0       0       4       5       2  traces/yi.trace
     0 (2,1,4)       0       0       0       2       3       1  traces/dave.trace
     0 (2,1,3)       0       0       0     167      71      67  traces/trans.trace
     0 (2,2,3)       0       0       0     201      37      29  traces/trans.trace
     0 (2,4,3)       0       0       0     212      26      10  traces/trans.trace
     2 (5,1,5)       0       0       0     231       7       0  traces/trans.trace
     0 (5,1,5)       0       0       0  265189   21775   21743  traces/long.trace
     2

TEST_CSIM_RESULTS=2
yunho@yunho-XPS-8700:~/PA4$
```

Helper Programs

valgrind : generate trace files

```
linux> valgrind --log-fd=1 --tool=lackey -v --trace-mem=yes ls -l > <filename.trace>
```

valgrind output

```
I 0400d7d4,8
 M 0421c7f0,4
 L 04f6b868,8
 S 7ff0005c8,8
```

<[operation] address, size >

I: instruction load (no space before [operation])

L: data load

S: data store

M: data modify (store after load)

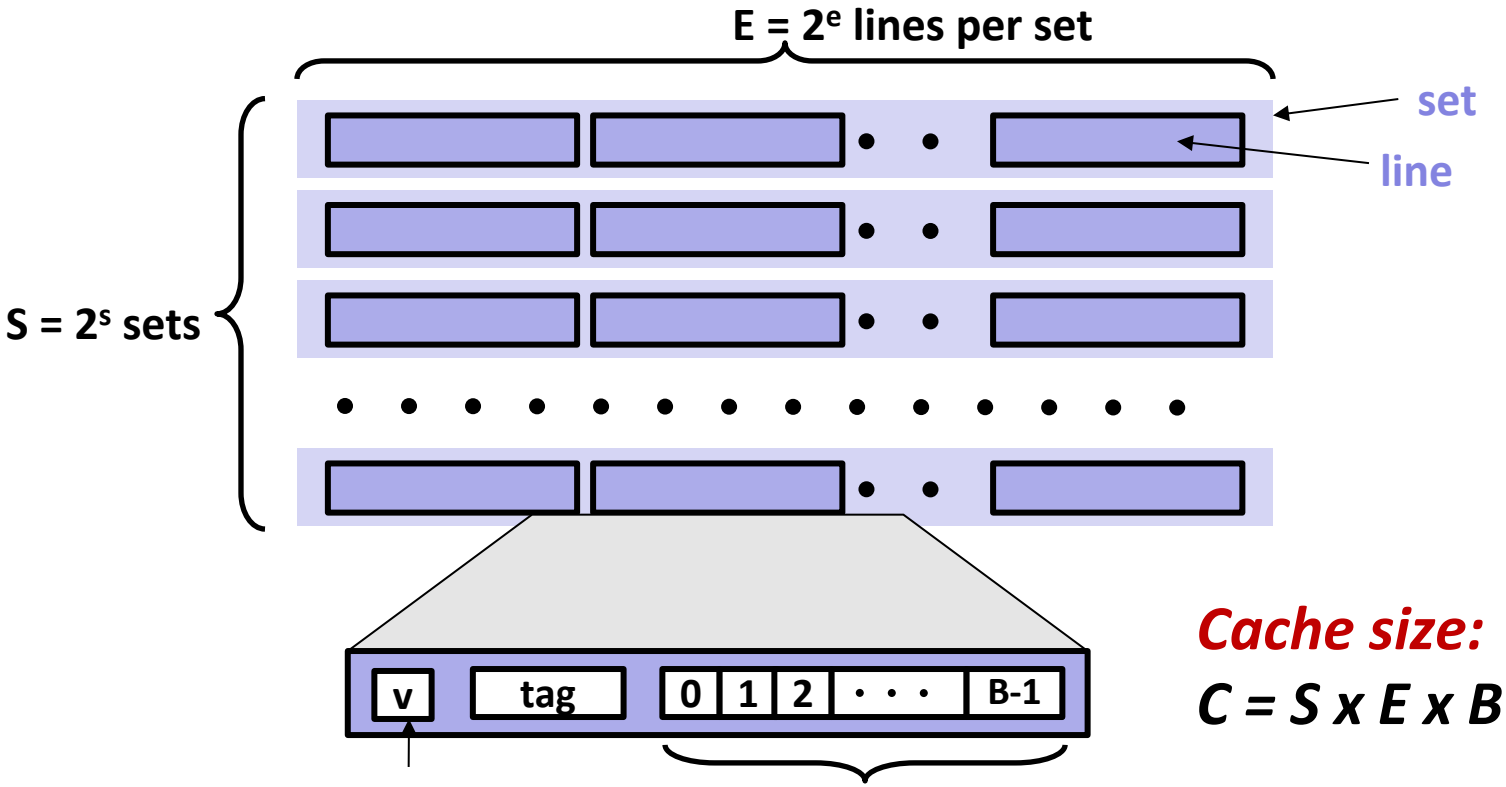
Helper Programs

csim-ref: reference cache simulator program

```
linux> ./csim-ref [-hv] -s <s> -E <E> -b <b> -t <trace file>
```

- -h: optional help flag
- -v: optional verbose flag
- -s: # of set index bits($S = 2^s$ number of sets)
- -E: number of lines per set (=associativity)
- -b: block bits ($B = 2^b$ bytes per block)
- -t: trace file

General Cache Organization (S, E, B)



Cache size:

C = S x E x B data bytes

Program rules

csim.c should run without failure

Should work for arbitrary *s, E, b*

Focus only on data cache

- ignore “I” operations in traces
- “I” operations start in the first column while others start in second (should help you parse trace files)

Call `printSummary()` in cachelab.c at the end of your main function

- printSummary(hit_count, miss_count, eviction_count);

All memory accesses are aligned properly

- Single memory access does not cross block boundary

Submission

Compress your source code and write-up into a single zip file.

- Compress csim.c and your report.
- Filename should be [student_id].zip (example: 2019-12345.zip).
- Please submit it in ZIP format. Other formats are not accepted.

Submission deadline: By 2019. 11. 20 (Wed) 23:59 KST

Grades

Writeup: 10%

Late Submission:

- Within next 24 hours: 10% deduction
- Within next 48 hours: 30% deduction
- Within next 72 hours: 50% deduction
- After next 72 hours: Submission not accepted