

COP-701: Major Exam

Instructions:

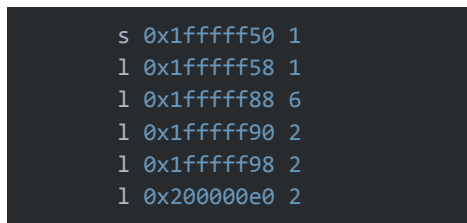
1. The duration of the exam is 12 hours. It will be from 10:00 AM to 10:00 PM.
2. You are not allowed to discuss or consult anyone throughout the exam.
3. For reviving your computer architecture knowledge, you can refer to the handout attached with the moodle activity.
4. You are not allowed to take any help from the internet. If any degree of coping is caught, you will be awarded an F grade in the course, Obviously with a Disciplinary Committee action.
5. You are only allowed to use C language for this assignment.
6. This exam is auto-graded; please refer to the template submission attached in the moodle activity. It has the evaluation script, sample test cases and sample submission. But, you only need to upload your submission, not with the evaluation script. Refraining yourself from maintaining that would result in 0 marks for the exam.

Problem Statement:

Being a research enthusiastic M-Tech research student, you have decided to design an intelligent prefetching scheme for the cache. But before that, you need to find the **optimal cache configuration** for defining the baseline for your research. To set up the baseline, you need to design a cache simulator for finding the best overall cache configuration for the given memory traces.

You are expected to design and implement a cache simulator to compare and study the effectiveness of various cache configurations. Your simulator would be provided with the **memory access trace** from the trace file and expected to simulate the cache operations in response to the memory access pattern. Subsequently, furnish the statistics to an output file.

Let us discuss the memory access pattern (shown in Figure-1) provided as input for simulation,



```
s 0xffffffff50 1
l 0xffffffff58 1
l 0xffffffff88 6
l 0xffffffff90 2
l 0xffffffff98 2
l 0x200000e0 2
```

Figure-1: Snippet of Execution Trace

Each line in the memory access pattern has three fields namely: memory access type, memory address and data corresponding either fetched from the disk or needs to be written into the disk. The first field is either *l* or *s*, where *l* is for load and *s* is for store operation. The second field is a 32-bit memory address provided in **hexadecimal**. You can use the third field for populating the cache data structure.

Note: We assume that each load or store in the trace accesses data of **at most 4 bytes of data** and do not read data that spans over multiple lines in the cache.

Your cache simulator can be configured with the following design parameters, which would be provided as command-line arguments while executing “*make run*”:

- number of sets in the cache (a positive power-of-2)
- number of blocks in each set (a positive power-of-2)
- number of bytes in each block (a positive power-of-2, at least 4)
- LFU (least-frequently-used) or LRU (least-recently-used) or FIFO (first in first out) evictions

With only the statistics of the cache hits and cache misses we will not be able to evaluate our cache design. We have assumed that we are using a **write allocate scheme** and **write back scheme** for handling the data transfer between the cache and the memory. To find the latency required by your designed cache simulator, we assume the following latencies:

1. Load / Store from/to the cache takes **1** processor cycle.
2. Load / Store from/to the memory takes **200** processor cycles **for each 4-byte word**.

We can ignore the other delays associated with the cache as we cannot design a cycle-accurate simulator with limited information.

Submission Guidelines:

You can **only use C** for this problem. You’re allowed only to use the standard library C as much as you would like to, but you are **not allowed** to use any additional (non-standard) libraries. We highly encourage you to write modular, well-designed code and develop data types and functions to manage the program’s complexity. **Strive for simplicity.**

Create a zip file named “KerberosID_Major.zip” with your Makefile, source code and header files, and README file describing your approach and observation. All of the files should be in the top-level directory of the zip file. You must provide a Makefile such that:

- **make clean**, remove all object files and executables.
- **make sim**, compile and link your program, producing an executable called sim.
- **make run num_sets=<number of sets> num_blocks=<number of blocks in each set> size_block=<size of block, at least 4> eviction_policy=<LRU || LFU || FIFO> trace_file=<trace_file_location> output_file=<output_file_location>**, runs the cache simulator with the defined design parameters.

```
make run num_sets=256 num_blocks=1024 size_block=32 eviction_policy=LFU  
trace_file=read01.trace output_file=output.txt
```

- Your code should compile cleanly with GCC-9 compiler flags: -Wall -Wextra --pedantic.

After the simulation is complete, your cache simulator is expected to print the following statistics in exactly the format given below in an output file:

```
Total loads: count
Total stores: count
Load hits: count
Load misses: count
Store hits: count
Store misses: count
Total cycles: count
```

Note:

1. In case you are **not able to** find a particular statistic, you can **print -1**, corresponding to it. Failing to do so, might create issues in the evaluation script.
2. *In case you are not able to implement this assignment. You may choose to submit a report on your ideas or methodologies which could have been used to design the simulator.*