Technion - Operating Systems - Wet Exercise #2

Technion - Israel Institute of Technology
Faculty of Electrical Engineering Computer Architecture - Winter 2025-2026 Course: 046267
HW 2 - Cache Memory
Short and to the Point In this exercise, you will implement your own Cache Simulator, similar to what was learned in class
1.The cache configuration will be flexible and defined at the start of the run via parameters.
2.Additionally, your simulator will read an input file detailing memory accesses.
3.Your simulator must calculate the Hit/Miss rate and the average memory access time for the defined configuration on the given trace (input file).

Cache Simulator Characteristics:
1. The simulator will simulate Data accesses only.
2. The simulator will contain two levels (L1 and L2).
3. The two cache levels will work with Write Allocate and No Write Allocate policies, and a Write Back policy.
4. The cache memory will operate according to the Inclusive principle.
5. The eviction policy is LRU.
6. At the start of the run, the cache memory is empty.
7. All accesses are 4 bytes in size and aligned to a 4-byte boundary (the two least significant bits of the address will always be 00).

Parameters Defined at Initialization:
1. Cache Memory Size (in bytes).
2. Block Size (in bytes).
3. Associativity Level.
4. Access Times (in clock cycles).
5. Policy: Write Allocate / No Write Allocate.

Input File Structure:
The lines of the input file contain a trace of memory accesses from a program run. Each line describes one access with two fields separated by a space:
1. Read (r) or Write (w).
2. The Address to read from or write to (in Hex)
Example:
w 0x1ff91ca8
r 0x20000cdc

An example input file is provided with the exercise materials.

So, what do you actually need to do?
Your program will be named "cacheSim" and its execution line will look like this:

```
./cacheSim <input file> --mem-cyc <# of cycles> --bsize <log2(block size)> --wr-alloc <0: No Write Allocate; 1: Write Allocate> --l1-size <log2(size)> --l1-assoc <log2(# of ways)> --l1-cyc <# of cycles> --l2-size <log2(size)> --l2-assoc <log2(# of ways)> --l2-cyc <# of cycles>
```

Important Notes:
1. Cache size, block size, and associativity level are given as powers of 2 (log2). The number of cycles is not a power of 2
2. All numbers are integers.
3. The cache memory size refers to the Data part only and does not include the size of the tag directory.

Example: If we want to simulate:
1. Main memory access time: 100 cycles.
2. Block size: 32B (2^5).
3. L1 Cache: 64KB (2^16) size, 8-way associativity (2^3), access time 1 cycle.
4. L2 Cache: 1MB size, 16-way associativity, access time 5 cycles.
5. Policy: Write Allocate.
6. Trace file: example.in.

We would use the following command:

```
./cacheSim example.in --mem-cyc 100 --bsize 5 --wr-alloc 1 --l1-size 16 --l1-assoc 3 --l1-cyc 1 --l2-size 20 --l2-assoc 4 --l2-cyc 5
```

Timing Calculations:
Access times for different levels do not include the access time for previous levels
L1 Miss, L2 Hit: t_access = t_L1 + t_L2
L1 Miss, L2 Miss: t_access = t_L1 + t_L2 + t_mem

Additional Rules:
- There is no need to account for Writeback overheads.
- Access from L1 to L2 resulting from a writeback does not affect the Miss Rate calculation.

Program Output:
The output must be strictly formatted as follows:

```
L1miss=<L1 miss rate> L2miss=<L2 miss rate> AccTimeAvg=<avg. acc. time>
```

- L1/L2 miss rate: Decimal fractions between 0 and 1 (not percentages), 3 digits after decimal.
- AccTimeAvg: Average access time, 3 digits after decimal.
- Rounding: Standard rounding.
- Add a newline character at the end.

ATTENTION!!!
1. Your programs will be checked automatically.

2. You must write the program in C or C++ only and provide a makefile.

Submission Requirements:
Method: Electronic submission via Moodle.
Deadline: December 27, 2025, at 23:55.
File Format: hw2_ID1_ID2.tar.gz