Lab4 implementation details – 25-1 Computer Organization

2023-11534 조현우

Part I

In our cache_base_c::access() function, each incoming address is first split into (offset, set index, tag) using the cache's line size and number of sets. We look up the tag in the target cache_set_c by calling its find method, which naively looks through all of the set's entries for a valid match. On a hit appropriate stat variables are updated and LRU_update is called to move that line to the MRU position in the set's LRU_queue. On a miss, we call evict which returns an invalid slot index if it exists, or evicts the LRU entry and returns its index. If the victim was dirty, we increment the writeback counter. We then overwrite the victim's tag and valid bit. At the very end we appropriately change the values of hit/miss and write counters. The stats for 16kB, 1/2/4/8 way, 64 line size cache is given as follows.

```
./run_base ../traces/sample.trace 16384 1 64
------------------------------
L1 Hit Rate: 95.1 %
------------------------------
number of accesses: 1000000
number of hits: 951000
number of misses: 49000
number of writes: 80669
number of writebacks: 5743
```

```
./run_base ../traces/sample.trace 16384 2 64
------------------------------
L1 Hit Rate: 96.4078 %
------------------------------
number of accesses: 1000000
number of hits: 964078
number of misses: 35922
number of writes: 80669
number of writebacks: 4377
```

```
./run_base ../traces/sample.trace 16384 4 64
------------------------------
L1 Hit Rate: 96.8525 %
------------------------------
number of accesses: 1000000
number of hits: 968525
number of misses: 31475
number of writes: 80669
number of writebacks: 2950
```

```
./run_base ../traces/sample.trace 16384 8 64
------------------------------
L1 Hit Rate: 97.1274 %
------------------------------
number of accesses: 1000000
number of hits: 971274
number of misses: 28726
number of writes: 80669
number of writebacks: 2783
```