Lab4: Cache simulator for a multi-level cache hierarchy

As part of this lab assignment you will build a cache simulator, with multiple cache levels, and configurable for cache size, linesize, and replacement policies. We will build this simulator in two parts. The first part concentrates only on the first level date cache (DCACHE). The second part deals with taking the cache object and simulating a two-level cache hierarchy and connecting it to main memory.

Part 4A:

In this part, you will estimate the cache miss rate for the DCACHE. As we are only interested in cache hit/miss information we will not be storing data values in our cache. We will provide you with traces for 100M instructions from three SPEC2006 benchmarks: bzip2, lbm, and mcf.

Each trace record consists of 4 bytes of instruction PC, 1 byte of instruction type (0: ALU 1:LOAD 2:STORE) and 4 bytes of virtual address for LD/ST instructions.

We will provide you with the trace reader and the ability to change cache parameters from command line. The tracer calls the memory system for LD ST instructions and the function in the memory system in turn calls the **cache_access** function for the DCACHE. If the line is not found in the DCACHE the memsys function calls the **cache_install** function for the DCACHE. At the end, the memory system also calls the cache_print_stats function for the DCACHE. The cache init function and cache print stats functions are already written for you. Your job is to write two functions:

1. **cache_access** – If the line is present in the cache if yes return HIT
2. **cache_install** – install the line in the cache, and track evicted line

You will modify ONLY these two functions in cache.c and leave all other files untouched (except for run.sh). The cache functions must be written such that it will work for different associativity, linesize, cache size, and for different replacement policies (such as LRU and Random).

How to begin:

1. Download the tarball "a4.tar.gz"
2. Do "tar –xvzf a4.tar.gz"
3. Cd a4
4. Cd src.4A
5. Make
6. ./sim –h (to see the simulator options)
7. check cache.h to see the cache data structures
8. check cache.c to implement the two functions
9. Do ./sim ../traces/bzip2.mtr.gz – to test the default config
10. Compare the results with the one in the report
11. Check ./run.sh to see how to run different configurations on the three trace files
12. Update the report 4A and submit only two files "cache.c and REPORT.4A.pptx". You are not allowed to change any other files or functions

Part 4B/C:

You will implement an ICACHE, DCACHE, and connect it to a unified L2 cache and DRAM. You will also estimate timing for each request in this part. More details on this part as we get close to the deadline for 4A.

FAQ:

1. You can use the timestamp method to implement LRU replacement. We have provided cycle_count as a means for tracking the time.
2. You will need the last_evicted line for part 4B, when you will need to schedule a writeback from the Dcache to the L2cache, and from the L2cache to DRAM.
3. For RANDOM replacement, you may get a minor change in the miss rate compared to what is shown in the report.