ECE 485/585: Microprocessor System Design

Winter 2020 TERM PROJECT

The project will be carried out in groups of two or three students. Final project report is due on Friday, March 20 11:59:59 PM.

Project Objective

The goal of the project is to simulate a single level of cache in software. You will write your own simulator in a software platform of your own choice (such as C, C++, JAVA, VHDL, Verilog etc.). The simulator will take the provided input address trace as its input. It will implement the workings of the cache. Specifically, it will keep track of which blocks are brought into the cache and which blocks are being evicted. At the completion of the trace, the simulator will provide statistics about cache hit ratio, read traffic, write traffic etc.

Note that there is no need to model the actual data stored in the cache. The input trace contains information only about the addresses that are being accessed. Since there is no data information in the trace, there is no way to model the data contents any way.

Project Details

Please read the following details carefully and follow them, when implementing your simulator:

1. **Input trace:** The simulator will take an address trace as its input. This trace captures the sequence of accesses that are being made to the cache. For each access, the trace provides the type of access and the memory address that is being accessed. Each line in the trace has the following format: *Access_type Hex_Address*

   Access_type signifies the type of the memory operation (read, write or invalidate). It is encoded as follows: *Access_type* = 0 indicates a read, *Access_type* = 1 indicates a write and *Access_type* = 2 indicates an invalidate operation.

   Hex_address is the address which is being accessed. It is encoded in the hexadecimal format. Each address is a 32-bit number (8 hexadecimal digits) and signifies the byte which is being accessed.

   The two fields in each line are separated by a whitespace.

   Example 1: Let us assume that the first line in the address trace is as follows:
   0 00A53C00
   This represents a read request to the byte address 0x00A53C00

   Example 2: Consider another line in the address trace:
   1 3C44DB20
   This represents a write request to the byte address 0x3C44DB20

   Example 3: Another line in the address trace is as follows:
   2 00A53C04
   This represents an invalidation request for address 0x00A53C04. If the cache line that contains address 00A53C04 is present in the cache, then it should be evicted from the cache and its valid bit should be reset to 0. If the invalidated address is not present in the cache, then nothing needs to be done.

2. **Cache Parameters:** Your model should support the following parameters to be configurable:

   - *num_sets*
   - *num_ways* (associativity)
   - *line_size* (in bytes)
   - *replacement_policy*

   To simplify the implementation, you should assume that *num_sets*, *num_ways* and *line_size* are always powers of 2, maximum value of *num_ways* is 8 and *line_size* ranges from 32 to 128.
   The modeled cache should support the following two replacement policies: (i) **True LRU** (*replacement_policy* = 0), and (ii) **1-bit LRU** (*replacement_policy* = 1). When a new cache line is brought into a set, it should first look to occupy an invalid cache block in that set. If there is no invalid block, then the replacement policy should choose the appropriate block as victim to make room for the new line. The details of the true LRU and 1-bit LRU replacement policies were discussed in class and are included in the lecture slides (ece585_lec10.pdf)
   As far as writes are concerned, you should assume a write-back cache with a write-allocate policy. You don't need to implement any timing details (such as hit time, miss penalty etc.) for the cache.

3. **Simulator output:** After the last access in the trace has been simulated, the simulator output should include the following statistics:

   - Total number of cache accesses
   - Number of cache reads
   - Number of cache writes
   - Number of invalidates
   - Number of cache hits
   - Number of cache misses
   - Cache hit ratio
   - Cache miss ratio
   - Number of evictions
   - Number of writebacks

4. **Project Evaluation:** Once you have understood all the above details, you are ready to implement the simulator. To help you with validation, a sample trace with known output has been posted to the course website. In addition, it is strongly recommended that you should develop your own test traces (following the trace format described above) to test different corner cases and validate the correctness of your simulator. Final project evaluation will be carried out with surprise traces that will be made available in Week 10.
   A major portion of your project grade will be based on the accuracy of your simulator output for the evaluated traces. You'll be required to include your simulator results in the final report.

Project Deadlines

- Inform instructor about your project group: Wednesday, February 19
- Project evaluations (Demo + Q&A): Wednesday, March 11
- Project report due: Friday, March 20