ECE 485
DUT 2019
Final Project Description

Your team is responsible for the design and simulation of a split L1 cache for a new 32-bit processor. The processor is not used in a multicore or multiprocessor configuration and does not need to support cache coherence.

Your L1 instruction cache is two-way set associative and consists of 16K sets and 64-byte lines. Your L1 data cache is four-way set associative and consists of 16K sets of 64-byte lines. The L1 data cache uses write allocate and is write-back except for the first write to a line which is write-through. Both caches employ LRU replacement policy and are backed by a shared L2 cache. The cache hierarchy is inclusive.

Describe and simulate your cache in Verilog, C, or C++. If using Verilog, your design does not need to be synthesizable. Your simulation does not need to be clock accurate and does not need to store/retrieve data.

Maintain and report the following key statistics of cache usage for each cache and display them upon completion of execution of each trace:

- Number of cache reads
- Number of cache writes
- Number of cache hits
- Number of cache misses
- Cache hit ratio

In order to maintain inclusivity the L1 caches may have to communicate with the shared L2 cache. To simulate this, you should display the following messages (where <address> is a hexadecimal address).

- Write to L2 <address>
  This operation is used to write back a modified line to L2 upon eviction from the L1 cache.
- Read from L2 <address>
  This operation is used to obtain the data from L2 on an L1 cache miss
- Read for Ownership from L2 <address>
  This operation is used to obtain the data from L2 on an L1 cache write miss

Modes

Your simulation must support at least two modes (without the need to recompile). In the first mode, your simulation displays only the required summary of usage statistics and responses to 9s in the trace file and nothing else. In the second mode, your simulation should display everything from mode 0 but also display the communication messages to the L2 described above (and nothing else). You may choose to have additional modes or conditional compilation that display debug information but these should not be present in the version that you demonstrate to me.

Your simulation should not require recompilation to change the name of the input trace file. Ideally, if you're writing your simulation in C, you should use command line arguments (e.g. argc, argv[]) to specify the mode and input file. If you're using Verilog, consider using the $value$plusargs function.

Project report

You must submit a *brief* report that includes relevant internal design documentation, assumptions and design decisions, the source code for your cache, any associated modules used in the validation of the cache (including the testbench if using Verilog), and your simulation results along with the usage statistics.

Traces

Your testbench must read cache accesses/events from a text file of the following format. You should not make any assumptions about alignment of memory addresses. You can assume that memory references do not span cache line boundaries.

```
n address
```

Where n is

0	read data request to L1 data cache
1	write data request to L1 data cache
2	instruction fetch (a read request to L1 instruction cache)
3	evict command from L2 (for L2 evictions and inclusivity)
8	clear the cache and reset all state (and statistics) but continue reading trace file
9	print contents and state of the cache (continue reading traces, don't clear cache)

The address will be a hex value. For example:

```
2 408ed4
0 10019d94
2 408ed8
1 10019d88
2 408edc
```

When printing the contents and state of the cache in response to a 9 in the trace file, use a concise but readable form that shows only the valid lines in the cache along with way and appropriate state and LRU bits.

Grading

The project is worth 100 points.   Your grade will be based upon:

- External specification
- Completeness of the solution (adherence to the requirements above)
- Correctness of the solution
- Quality and readability of the project report (e.g. specifications, design decisions)
- Validity of design decisions
- Quality of implementation
- Structure and clarity of design
- Readability
- Maintainability
- Testing
- Presentation of results