**Microprocessor System Design**

**ECE – 485/585**

**Fall 2018**

**Final Project – Cache Simulation**

Vinitha Baddam <[vbaddam@pdx.edu](mailto:vbaddam@pdx.edu)>

Michael Bourquin <[bourquin@pdx.edu](mailto:bourquin@pdx.edu)>

Hima Ethakota <hima@pdx.edu>

# Contents

1. OVERVIEW 2

2. SPECIFICATIONS 4

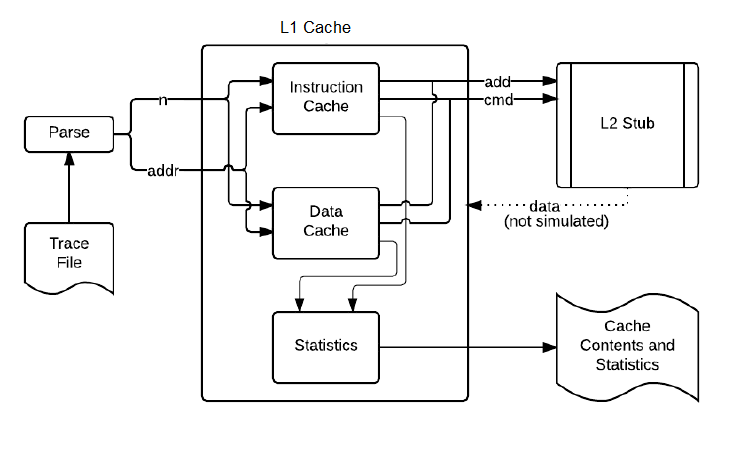
3. ASSUMPTIONS 4

4. SOURCE CODE 5

5. TEST CASES WITH OUTPUT 27

# **OVERVIEW**

This project requires the simulation of a split L1 cache for a 32-bit processor with having multiple processors. The data cache is 8-way set associative having 16k sets and 64-byte lines and the instruction cache is 4-way set associative having 16k sets and 64-byte lines. The cache employs MESI protocol to maintain cache coherence. The L1 data cache is write-back using 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 number of reads, writes, hits, misses and the hit rate are calculated and recorded.



# **SPECIFICATIONS**

1. The 32-bit address is broken into the offset bits, index bits and tag bits. The data cache is 8-way set associative having 16K sets and 64-byte lines and the instruction cache is a 4-way set associative having 16K sets and 64-byte lines.

**31:20 = 12-bit tag**

**19:6 = 14-bit index**

**5:0 = 6-bit offset**

1. We give a command to perform various functions in L1 and L2 cache as the project specifies. The output is simulated and the cache contents are displayed in the report/statistics. There is a command for print which will display the contents in the cache that comprise of offset bits, index bits, tag bits, MESI and LRU bits. For the data cache as it is 8-way associative, we have 4 bits for LRU and for the instruction cache being 4-way associative, we have 2 bits for the LRU. MESI bits are represented as 2 bits.
2. When printing the contents and state of the cache in response to a 9 in the trace file, we shows only the valid lines in the cache along with way and appropriate state and LRU bits.
3. Following is the trace file format

**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 - Invalidate command from L2

4 - Data request from L2 (in response to snoop)

8 - Clear the cache and reset all state (and statistics)

9 - Print contents and state of the cache (allow subsequent trace activity)

And the address will be a hex value.

There will be no address for commands (n) 8 and 9

1. In order to maintain inclusivity and implement the MESI protocol 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).
2. Return data to L2 <address>

In response to a 4 in the trace file your cache should signal that it’s returning the data for that line (if present and modified)

1. Write to L2 <address>

This operation is used to write back a modified line to L2 upon eviction from the L1 cache. It is also used for an initial write through when a cache line is written for the first time so that the L2 knows it’s been modified and has the correct data

1. Read from L2 <address>

This operation is used to obtain the data from L2 on an L1 cache miss

1. Read for Ownership from L2 <address>

This operation is used to obtain the data from L2 on an L1 cache write miss

1. 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

# **ASSUMPTIONS**

In case of designing the L1 cache, the following assumptions were made:

1. The cache employs MESI protocol and also follow inclusivity hierarchy to maintain cache coherence. We use 2 bits to represent the MESI bits in the caches.
2. LRU replacement policy is used to update the cache after every transaction takes place. The LRU policy we use is the counter method.
3. There is no data hence the cache contents would display the LRU, MESI, tag, index and offset bits.
4. The data cache is a write through cache in the first transaction and from the next transaction is a write back one.
5. All read and write operations are referred to single byte locations.
6. A read request from L2 cache can be because of i) Read miss in other processor’s cache ii) Write miss in other processor’s cache, hence we assume that the processors employ write allocate.
7. The MESI bits which are of 2-bit size are encoded as:

Invalid = 2'b00

Exclusive = 2'b01

Shared = 2'b10

Modified = 2'b11

1. The LRU replacement policy is done using the counter method and the most recently used bit is encoded as 111 for data cache (11 for instruction cache) and the least recently used bit is encoded as 000 for data cache (00 for instruction cache). The bits higher than the most recently used bits are decremented by 1 and then updated accordingly.
2. In our simulation we have considered two mode output

MODE=0: In this mode, our simulation displays only the required summary of usage statistics and responses to 9s in the trace file and nothing else

MODE=1: In this mode, our simulation should display everything from mode 0 but also display the communication messages to the L2 as described in the specification and nothing else