ECE 585: Fall'17 Final Project Report

Group 5:Manjush P V, Rithvik M B, Suprita Kulkarni, Vadiraja M N

December 7, 2017

Abstract

A split L1 cache for a 32 bit processor which is used with upto 3 other processors in a shared memory configuration maintains cache coherence using MESI protocol is simulated using C++ code.

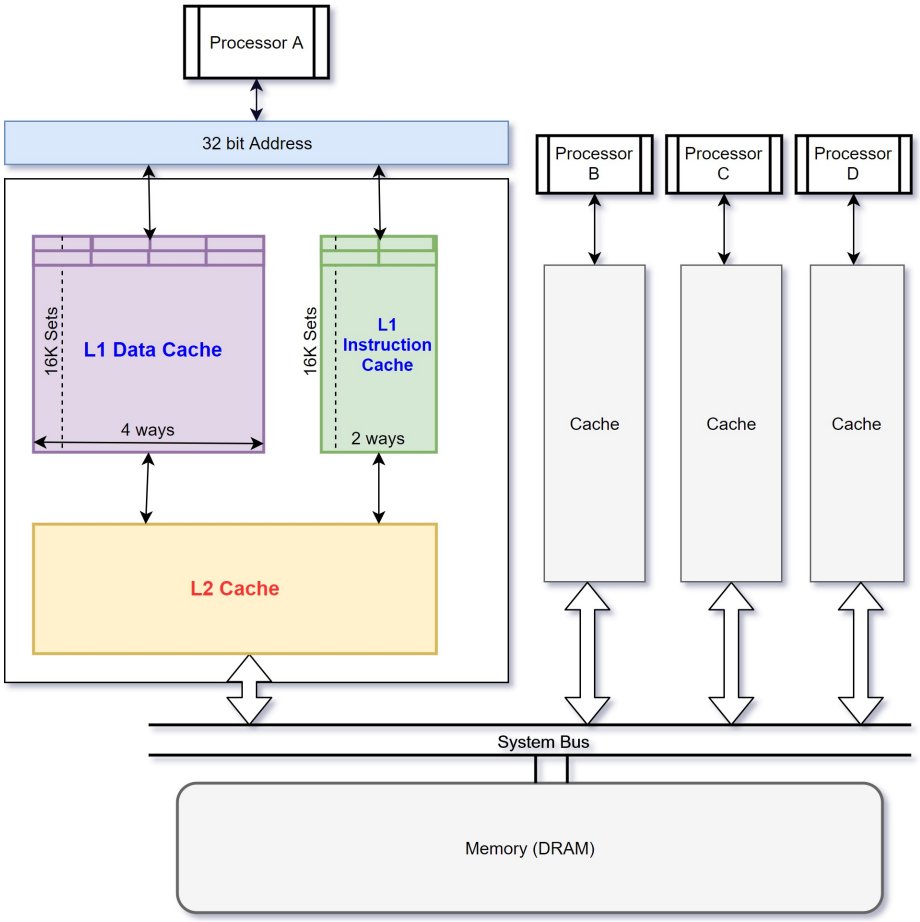


Figure 1: Design Description

1 Implementation Overview

To implement the split L1 cache, memory is allocated as two arrays of structure `Data_Cache[set][Associativity]`,`Instruction_Cache[set][Associativity]`. Each array cell represents a cache line containing the Tag, the LRU bits and the MESI state.

1.1 Accepting the Inputs

The `argc[],argv[]` accept two inputs, the mode and a trace file from the command line. The trace file consists of a number of memory references or operations to be performed on the cache. The hexadecimal address in each memory reference is then passed to `Address_Bifurcation()`, where it is converted to binary and the byte offset, the index and the tag are obtained.

1.2 Determining the Cache Line Status

Once the index bits and tag bits are separated from the input address, we determine the following information for the given memory reference:
1 If it is a hit or a miss using the `hit()` function.
2 If the set isn't full, the way of an invalid line is obtained with the `invalidLine()` function.

1.3 Replacement Policy

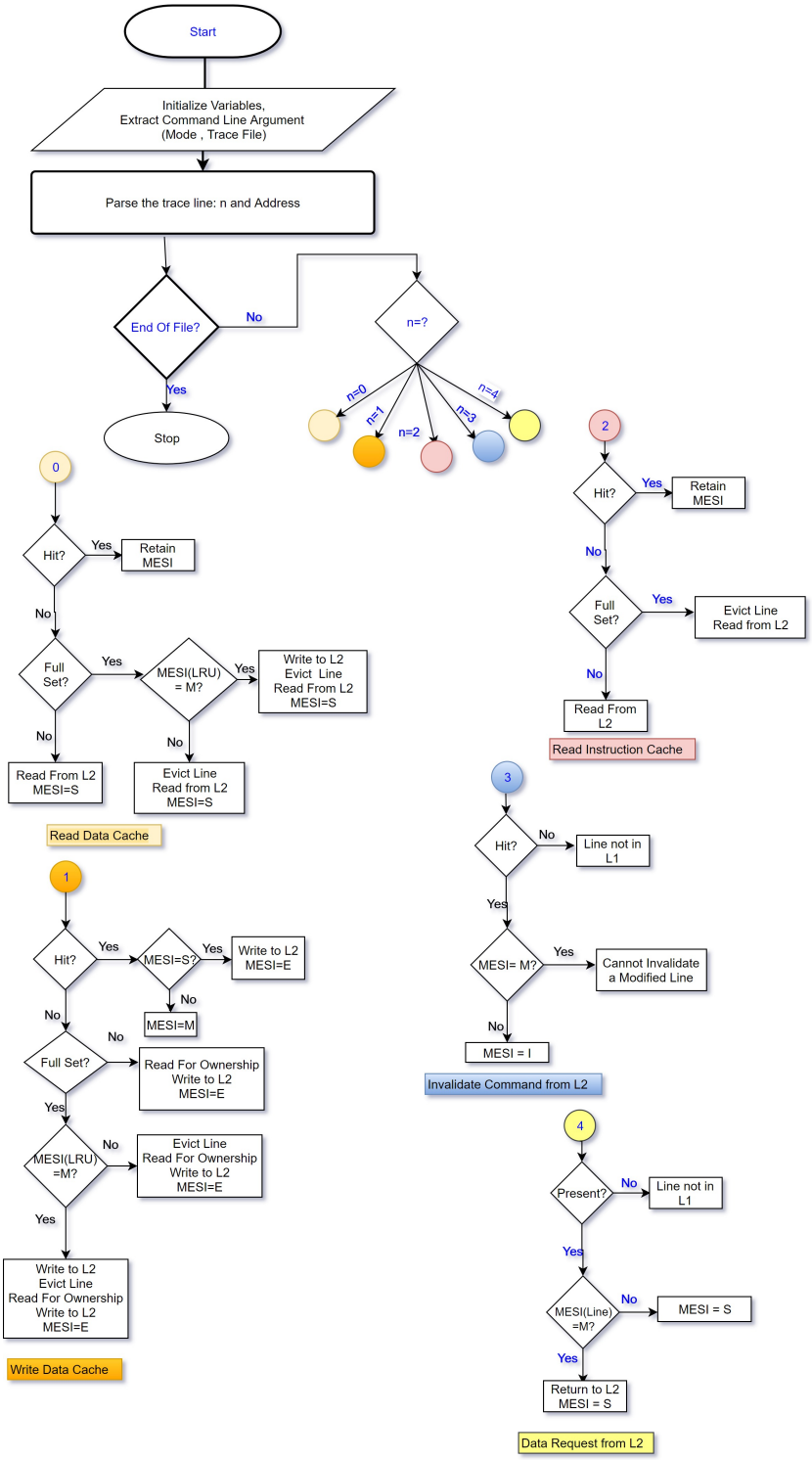
LRU replacement policy using the counter method is implemented by means of three functions.
1. `highestLRU()` : For a given set, this function determines the line with the highest LRU count, i.e the least recently used cache line.
2. `LRU_update()`: In case of a hit, the count of that line is set to 0, and the count of lines with value less than the hit line is incremented. In case of a miss and the set being full, the way of the LRU line i.e. the line with highest LRU count value is fetched from the `highestLRU()` function, and is set to zero(evicted). The count of all the other lines are incremented. In case of a set not being full, the LRU count of the line being read or written to is set to zero, all the other counts are incremented by one.
3. `LRU_update_Invalidate()`: If a line in a given set is invalidated, the LRU count of the lines with value higher than the invalidated line are decremented by one to observe proper functionality.

1.4 Clear Cache and Display Cache Contents

For input n=8, the `initializeCache()` and the `clearStats()` functions are used to invalidate all lines in the cache and to reset all the statistics which are `Read_Count`, `Write_Count`, `Hit_Count, Miss_Count` respectively.

For input n=9, `Display_Cache_Contents()` prints the tag, the MESI states and the LRU count values of all the valid cache lines.
After the execution of each trace, `Display_Stats()` prints out the statistics with respect to the cache type.

1.5 Flow chart



1.6 Assumptions

1. L1 and L2 maintain inclusivity without having to invalidate the L1-Instruction cache.
2. For n=8, the cache lines that have MESI state as M, are written back to L2.

Trace Output Table

Memory reference	n	Address	Status	Remarks	MESI states	Message	Update LRU function
Read	0	1ddf40	Miss	Way is not full	I to S	Read from L2 1ddf40	LRU_Update_Miss(...)
	0	2ddf73	Hit	Way is not full	S	NA	LRU_Update_Hit(...)
	0	ddde4b	Miss	The ways are full because of which there will be an eviction of line with highest LRU value. If the MESI State is E then the state changes to S upon eviction.	S	Read from L2 ddde40	LRU_Update_Miss(...)
	0	4dde75	Miss	The ways are full because of which there will be an eviction to a modified line (since it has the highest LRU value)	M to S	Write to L2 fdde40 Read from L2 4dde40	LRU_Update_Miss(...)
Write	1	addf43	Miss	Way is not full	I to E	Read for Ownership addf40 Write data to L2 addf40	LRU_Update_Miss(...)
	1	addf42	Hit	Way is not full. If the MESI State is M then the state will be M upon eviction.	E to M	NA	LRU_Update_Hit(...)
	1	1ddf40	Hit	Way is full	S to E	Write data to L2 1ddf40	LRU_Update_Hit(...)
	1	2ddf73	Miss	The ways are full and because of which there will be an eviction to a modified line (since it has the highest LRU value).	M to E	Write data to L2 addf40 Read for ownership 2ddf40 Write data to L2 2ddf40	LRU_Update_Miss(...)
	1	4ddf66	Miss	The ways are full because of which there will be an eviction of line with highest LRU value. If the MESI State is S then the state will go to E upon eviction.	E	Read for ownership 4ddf40 Write data to L2 4ddf40	LRU_Update_Miss(...)
Instruction	2	1dde52	Miss	Way is not full	I to S	Read from L2 1dde40	LRU_Update_Miss(...)
	2	1dde40	Hit	Way is not full	S	NA	LRU_Update_Hit(...)
	2	bdde7d	Miss	The ways are full because of which there will be an eviction of line with highest LRU value.	S	Read from L2 bdec40	LRU_Update_Miss(...)
Invalidate	3	adde71	NA	Invalidating an exclusive/shared line. If the MESI State is S then the state will go to I upon eviction.	E to I	NA	LRU_Update_Invalidate(...)
	3	adde71	NA	Invalidating an invalid line	NA	Cache Line not in L1 Data cache with address adde40	NA

	3	ddde4b	NA	Invalidating a modified line	NA	Modified Line shouldn't be invalidated -> Send "Return Data to L2" before invalidating...	NA
Data request from L2	4	2ddf73	NA	L2 is requesting data from L1	S	NA	NA
	4	2ddf73	NA	L2 is requesting data from L1(after a write hit)	E to S	NA	NA
	4	2ddf73	NA	L2 is requesting a data from L1(for a modified line)	M to S	Return data to L2 2ddf40	NA
	4	bddf41	NA	Requesting for a line that is not present in L1	NA	Cache Line not in L1 Data cache with address adde40	NA