Ian Buitrago
Miguel Diaz
11-30-2011

CS 352 Project: L1 Cache Simulator Report

Problem Statement

The goal of the project is to create a level 1 data cache simulator in C++. Using object oriented programming, source code is written to clearly illustrate how a data cache works. Also, by passing different parameters for the dimensions of the cache/memory, statistics are printed by the simulator that can be compared. Varying parameters cache size (KiB), block size (Bytes), and associativity will affect the cache miss rates.

Solution

.cc files can be made to include data structures for the cache and main memory.
Inside the cache data structure, an array of sets can be made. Each set would contain n blocks, depending on what the associativity is.

The cache would then hold all blocks of words until a block has to be evicted to the cache, depending on the cache replacement policy implemented (in this case, LRU).

The main memory would be given space for 16 megabytes (4 megawords, since 1 word = 4 bytes). This memory would only be written to when cache blocks need to be evicted from the cache or when all input of reads/writes are done.

Experiments and Data

The top 3 optimal configurations to minimize total cache miss rate:

1) -c64 -b512 -a16 (Total Cache Miss Rate: 0.027391)
2) -c64 -b512 -a8 (Total Cache Miss Rate: 0.0274506)
3) -c64 -b512 -a4 (Total Cache Miss Rate: 0.0274903)

The top 3 optimal configurations to minimize read cache miss rate:

1) -c64 -b512 -a16 (Read Cache Miss Rate: 0.0143009)
2) -c64 -b512 -a8 (Read Cache Miss Rate: 0.0148339)
3) -c64 -b512 -a4 (Read Cache Miss Rate: 0.0151892)

The top 3 optimal configurations to minimize write cache miss rate:

1) -c64 -b512 -a16 (Write Cache Miss Rate: 0.0290388)
2) -c64 -b512 -a8 (Write Cache Miss Rate: 0.0290388)
3) -c64 -b512 -a4 (Write Cache Miss Rate: 0.0290388)

To do this, a loop with two inner loops were written to simulate through different setting for cache_associativity, cache_blocksize, and cache_capacity, with the values being

cache_associativity: 1, 2, 4, 8, 16
cache_blocksize (in bytes): 4, 8, 16, 32, 64, 128, 256, and 512
cache_capacity (in kilobytes): 1, 2, 4, 16, 64

Ian Buitrago
Miguel Diaz
11-30-2011

Experiments and Data (continued from page 1)

The values that were output in the statistics were used to make 48 graphs that have been posted at the end of this report, 2 per page. Each page contains a “Cache Associativity vs. Total Cache Miss Rate” and a “Cache Size vs. Total Cache Miss Rate”. Every three pages changes block size, and every page changes between either total cache miss rate, read cache miss rate, and write cache miss rate.

Writing out contents of each page as a mini table of contents...

Ian Buitrago
Miguel Diaz
11-30-2011

Conclusion of Experiments

The results of these runs show that...

- as block size increases, cache miss rates tend to decrease.
- as associativity increases, cache miss rates tend to decrease.
- as cache size increases, cache miss rates tend to decrease.

Class Design

The cache simulator with 4 classes (including 2 structs). MainMemory, CacheMemory, Set, and CacheLine. Each aims to simulate that hardware compenent. Most of the implementation is in the CacheMemory class. It contains a MainMemory object and an array of Set objects. It also contains all the statistic variables and other useful variables as private members.

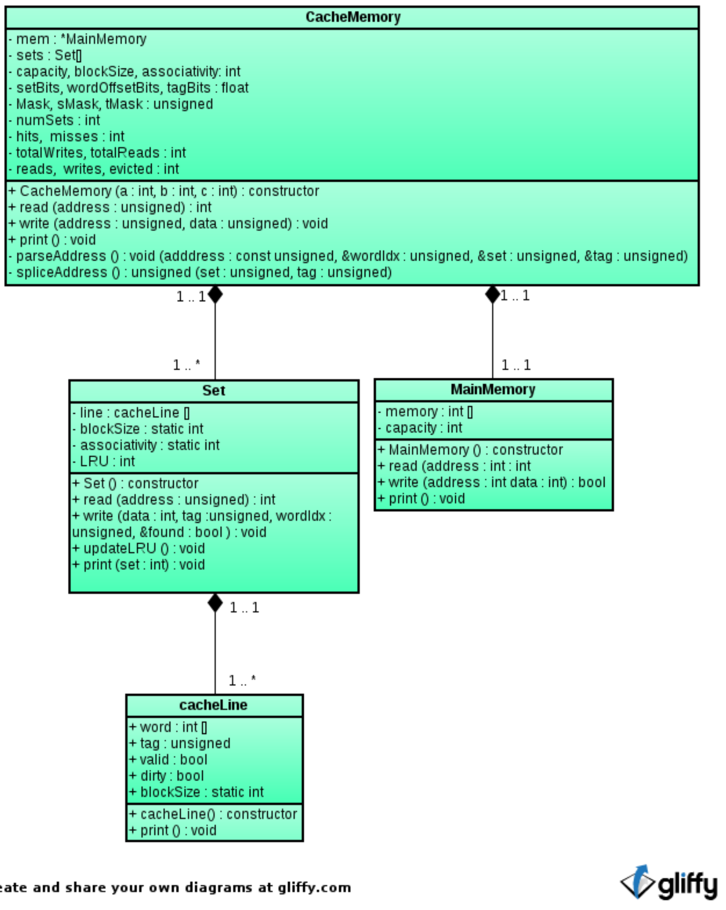
The MainMemory class has an integer array to hold all the data. It has simple accessors and mutators to interact with that data along with a print() method called by CacheMemory's print().

The Set struct, essentially a class, contains an array of CacheLine objects, the LRU algorithm, and read, write and print methods that are used only by the CacheMemory object.

The CacheLine struct contains and array of integer words, the tag and the valid and dirty bits. The print method is only called by Set's print method.

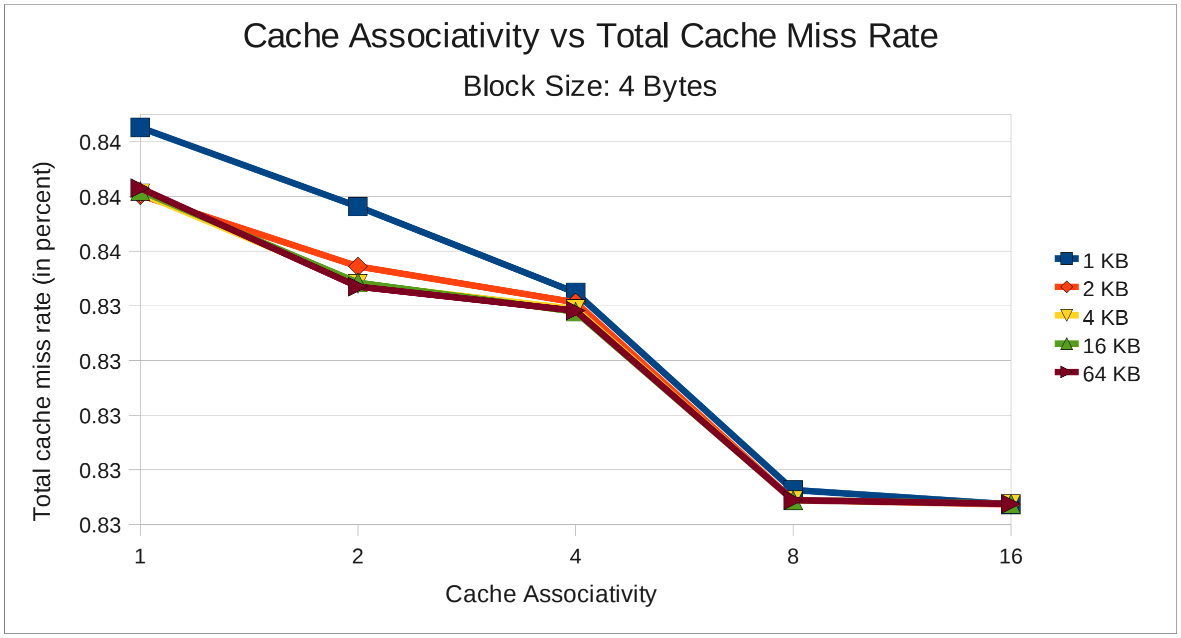
All the constructors help to initialize the data automatically when the objects are created.

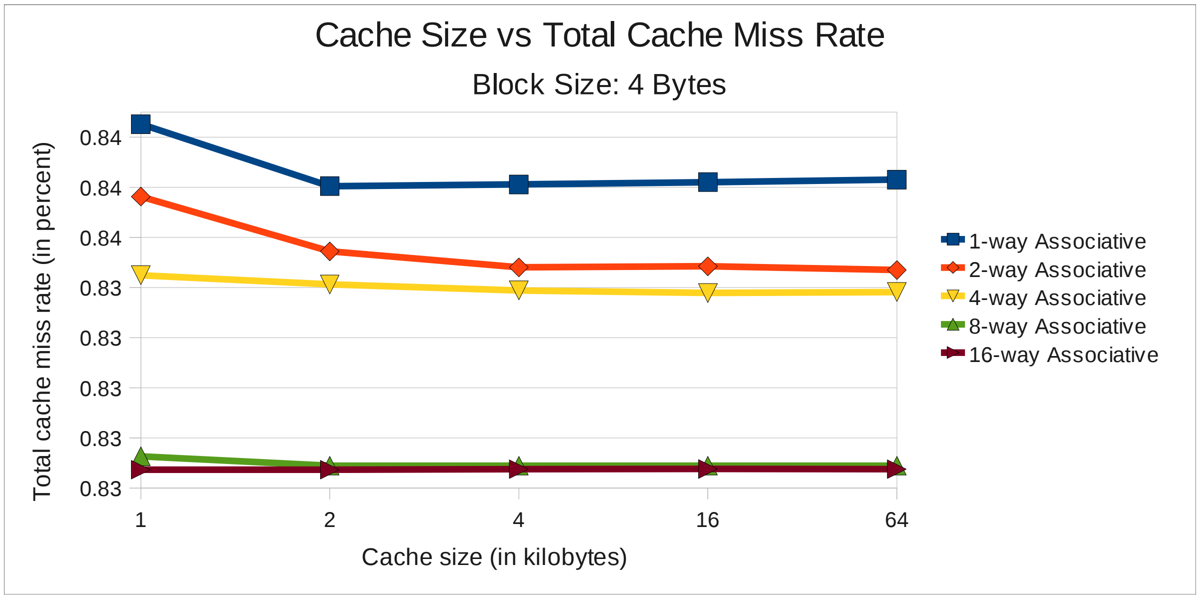
This designed helped keep methods small and readable. CacheMemory relied on Set to operate on it's data and Set relied on CacheLine to operate on data.


CacheMemory
- mem : *MainMemory
- sets : Set[]
- capacity, blockSize, associativity: int
- setBits, wordOffsetBits, tagBits : float
- Mask, sMask, tMask : unsigned
- numSets : int
- hits, misses : int
- totalWrites, totalReads : int
- reads, writes, evicted : int
+ CacheMemory (a : int, b : int, c : int) : constructor
+ read (address : unsigned) : int
+ write (address : unsigned, data : unsigned) : void
+ print () : void
- parseAddress () : void (adddress : const unsigned, &wordIdx : unsigned, &set : unsigned, &tag : unsigned)
- spliceAddress () : unsigned (set : unsigned, tag : unsigned)
1 .. 1
1 .. *
1 .. 1
1 .. 1
Set
- line : cacheLine []
- blockSize : static int
- associativity : static int
- LRU : int
+ Set () : constructor
+ read (address : unsigned) : int
+ write (data : int, tag :unsigned, wordIdx : unsigned, &found : bool) : void
+ updateLRU () : void
+ print (set : int) : void
MainMemory
- memory : int []
- capacity : int
+ MainMemory () : constructor
+ read (address : int : int
+ write (address : int data : int) : bool
+ print () : void
1 .. 1
1 .. *
cacheLine
+ word : int []
+ tag : unsigned
+ valid : bool
+ dirty : bool
+ blockSize : static int
+ cacheLine() : constructor
+ print () : void
create and share your own diagrams at gliffy.com
gliffy

Ian Buitrago
Miguel Diaz
11-30-2011

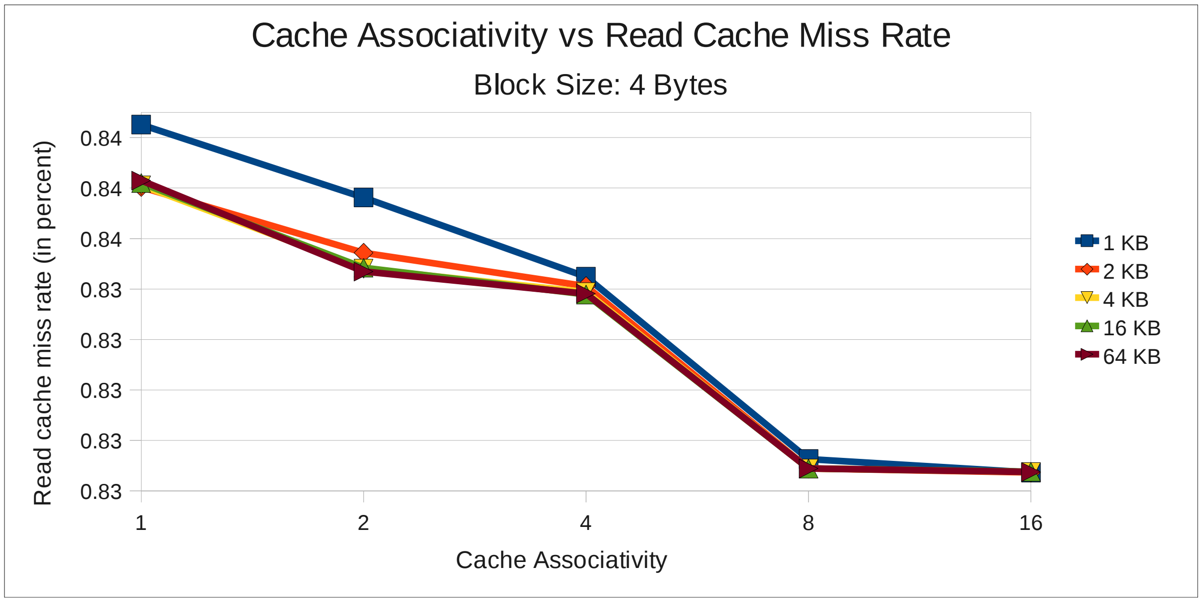
Block Size 4
Total Cache Miss Rate

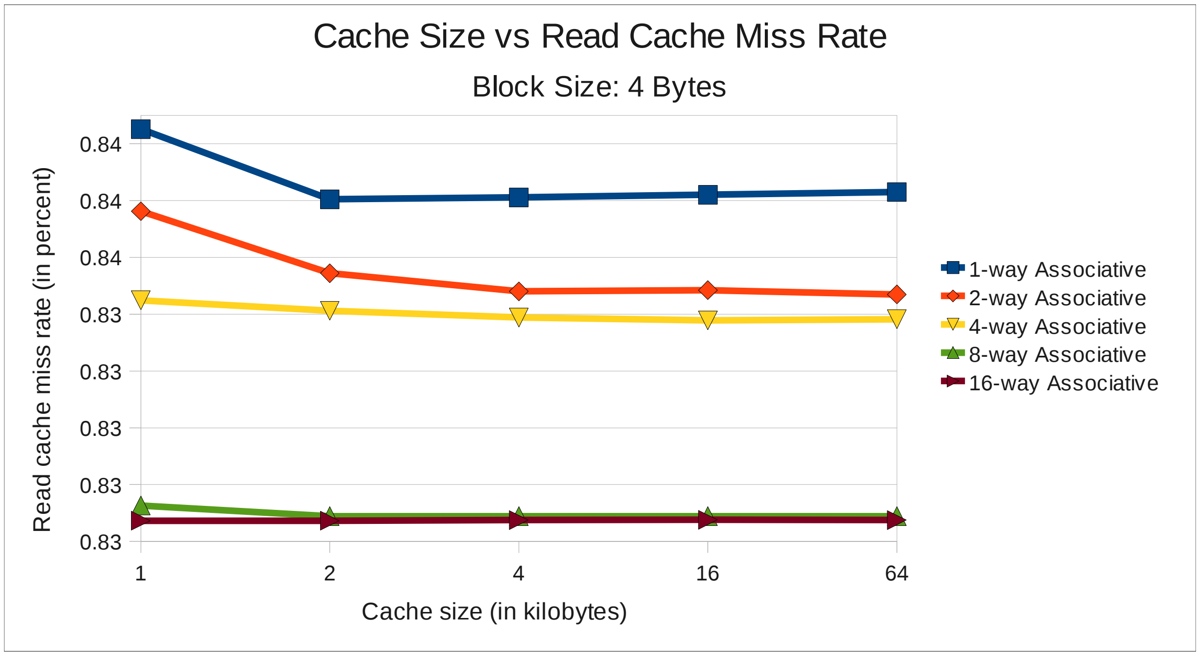




Ian Buitrago
Miguel Diaz
11-30-2011

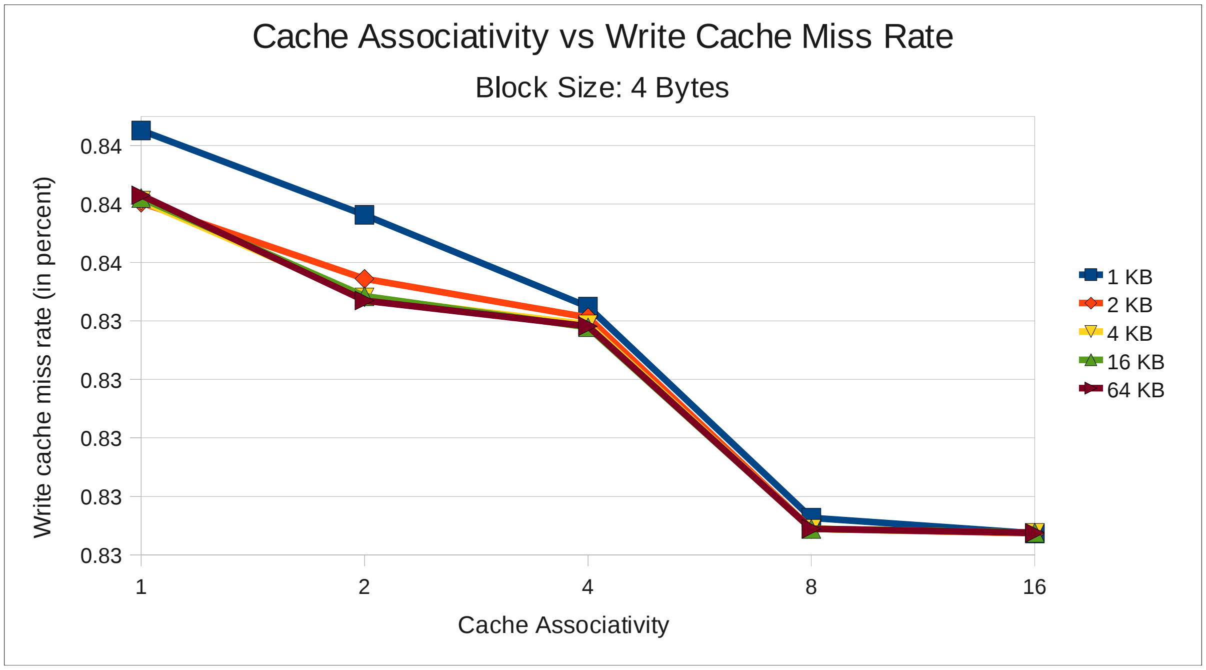
Block Size 4
Reads Cache Miss Rate

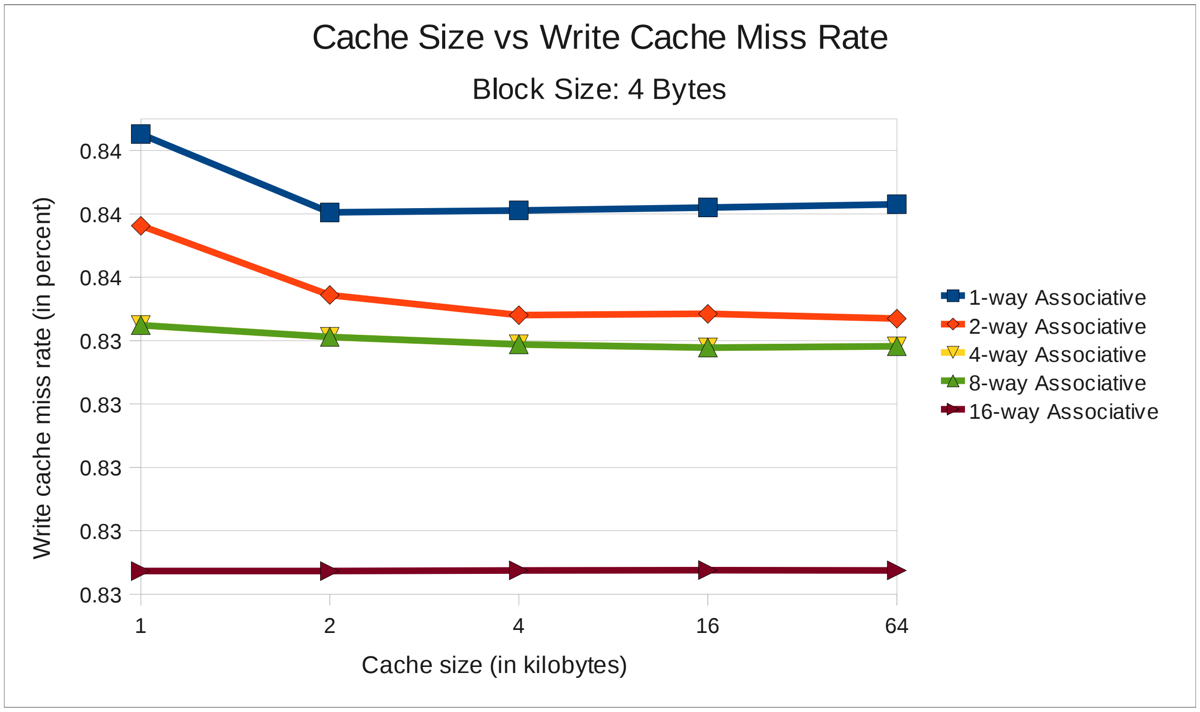




Ian Buitrago
Miguel Diaz
11-30-2011

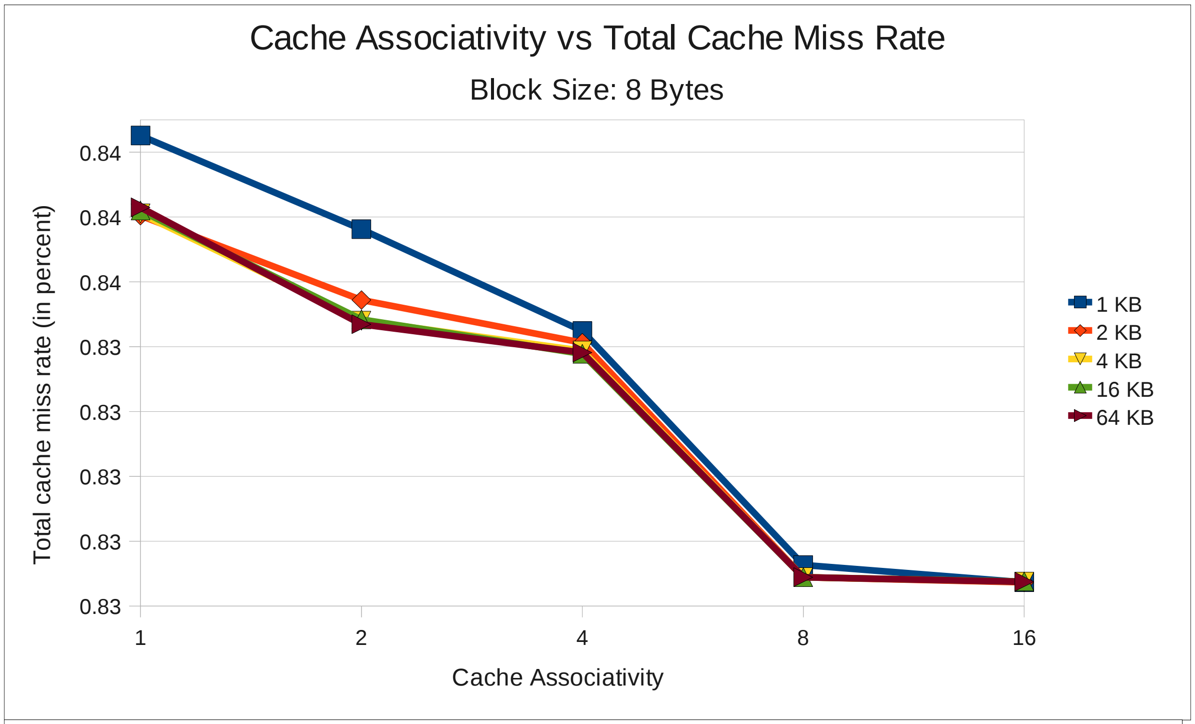
Block Size 4
Writes Cache Miss Rate

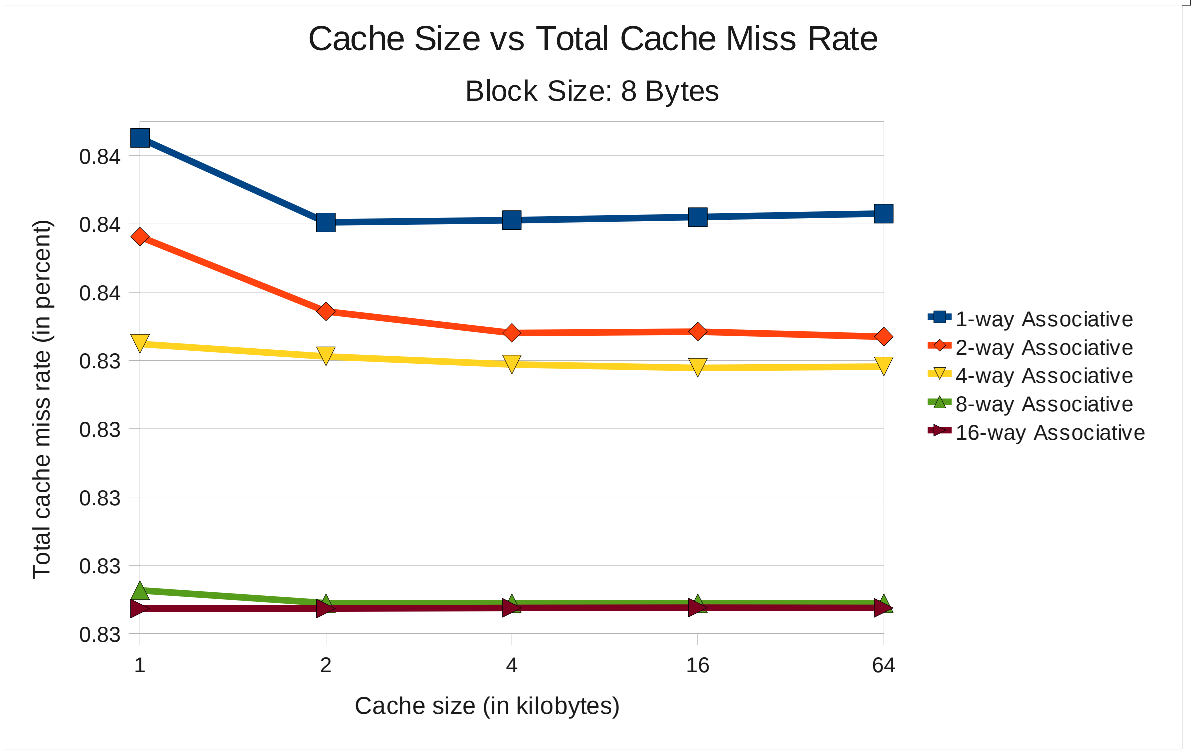




Ian Buitrago
Miguel Diaz
11-30-2011

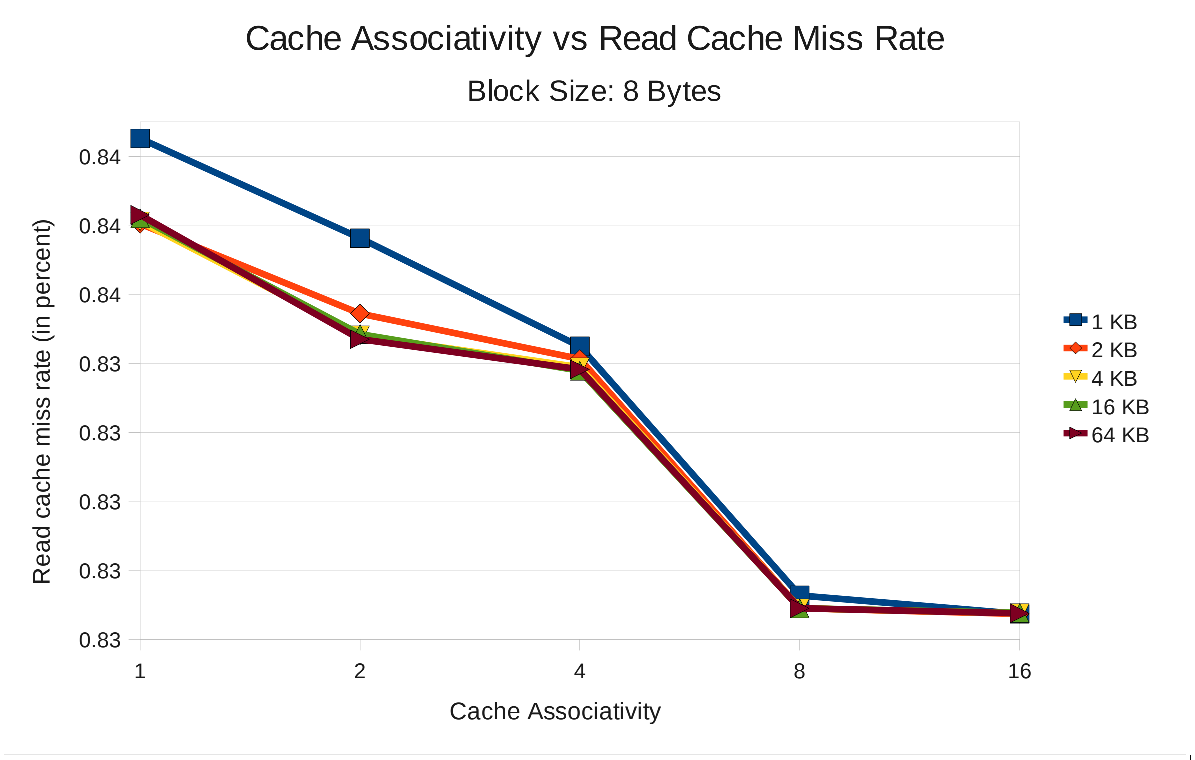
Block Size 8
Total Cache Miss Rate

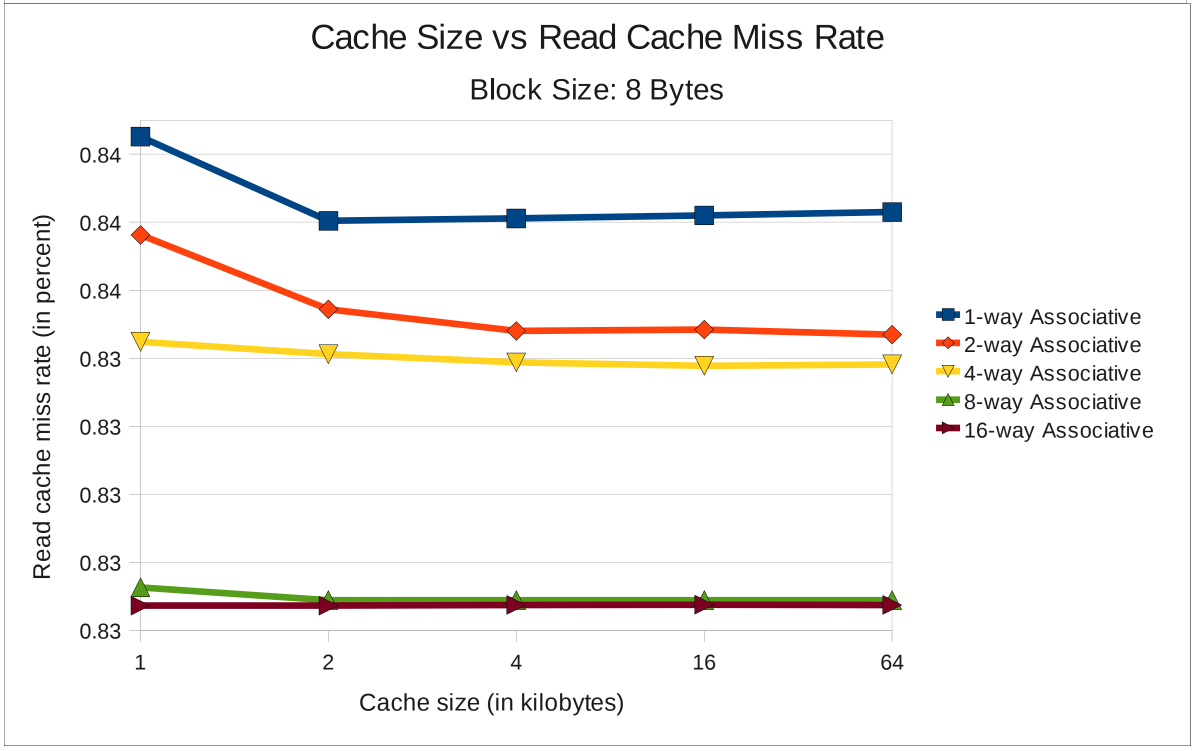




Ian Buitrago
Miguel Diaz
11-30-2011

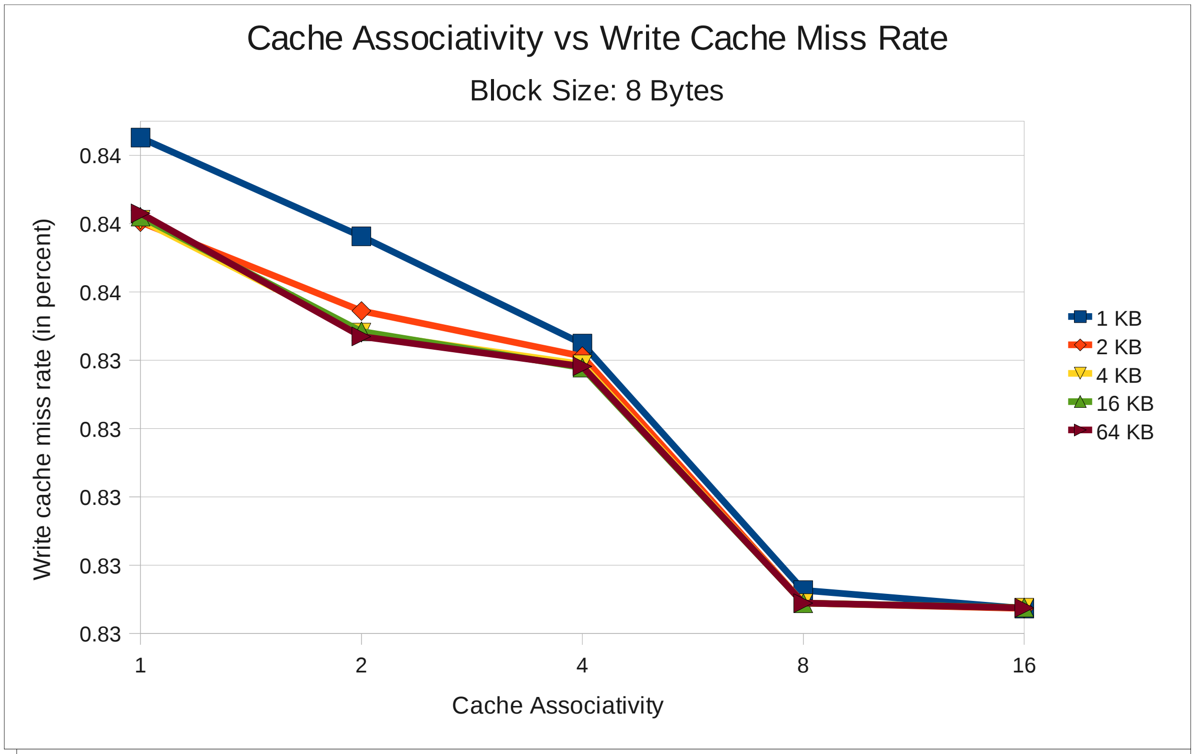
Block Size 8
Reads Cache Miss Rate

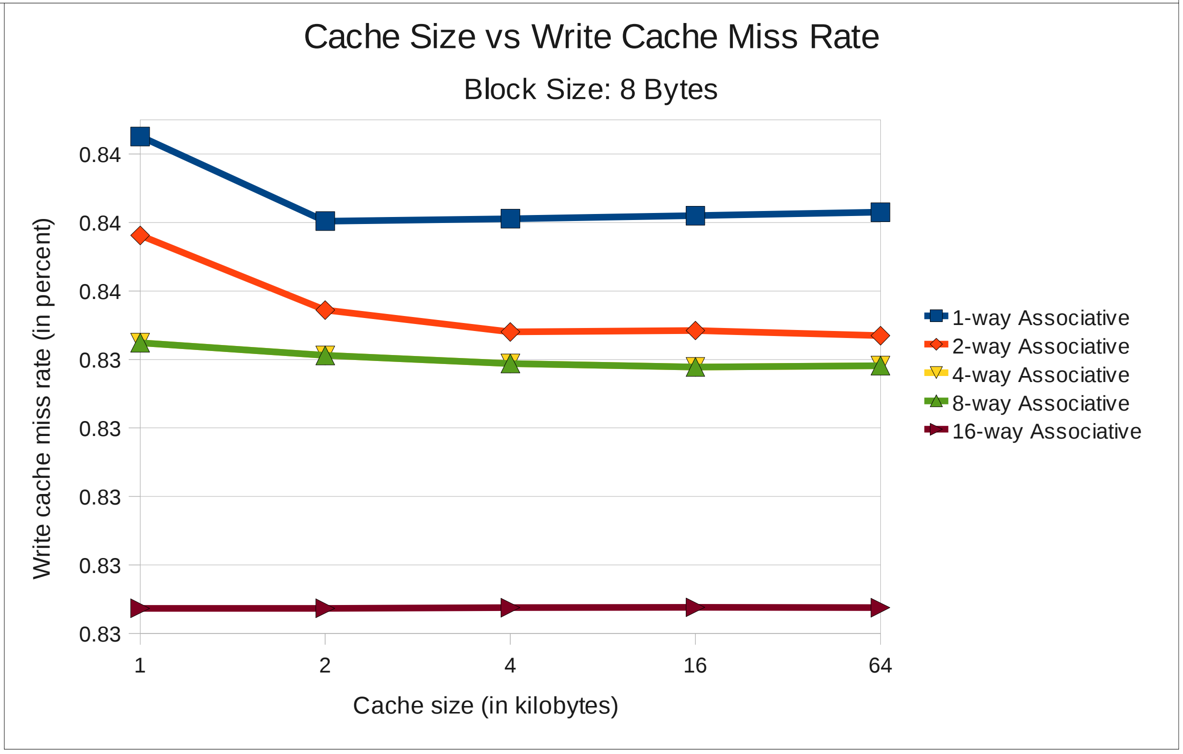




Ian Buitrago
Miguel Diaz
11-30-2011

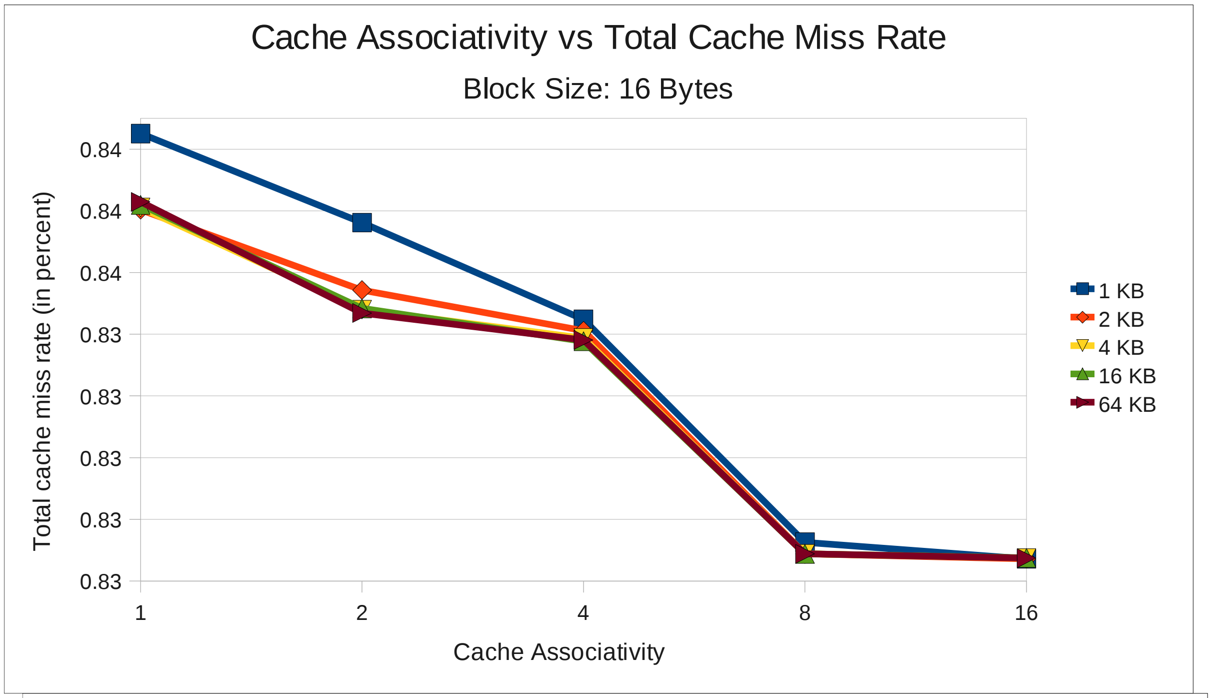
Block Size 8
Writes Cache Miss Rate

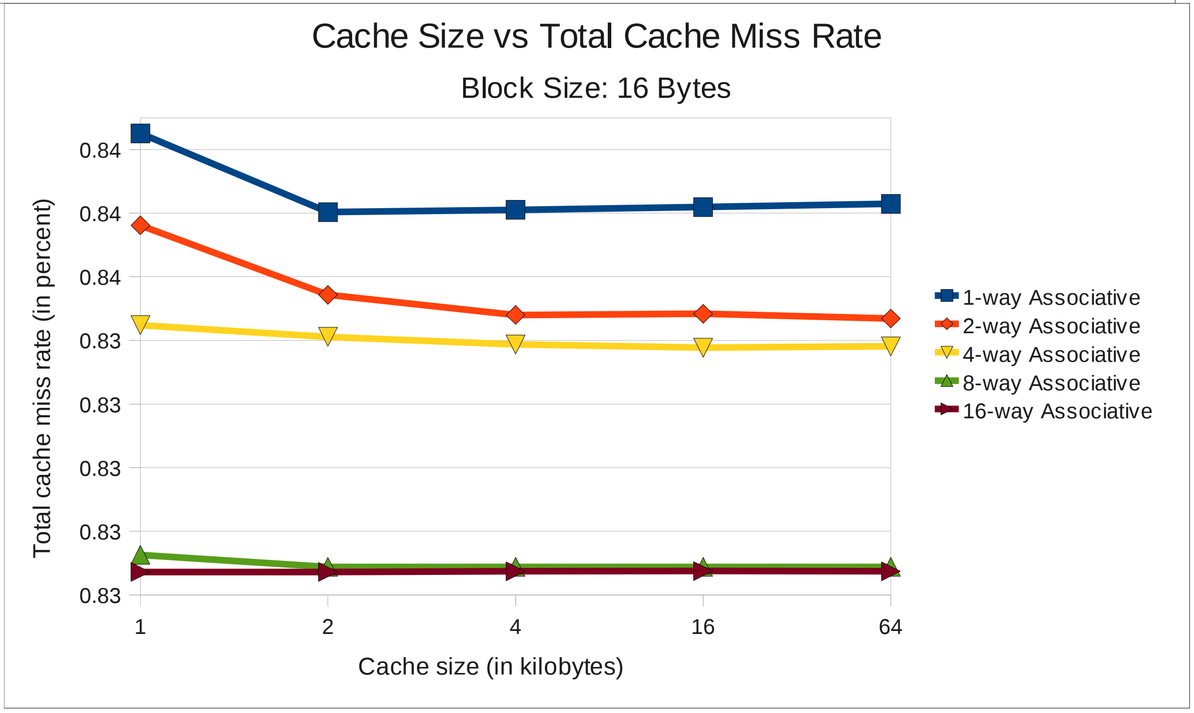




Ian Buitrago
Miguel Diaz
11-30-2011

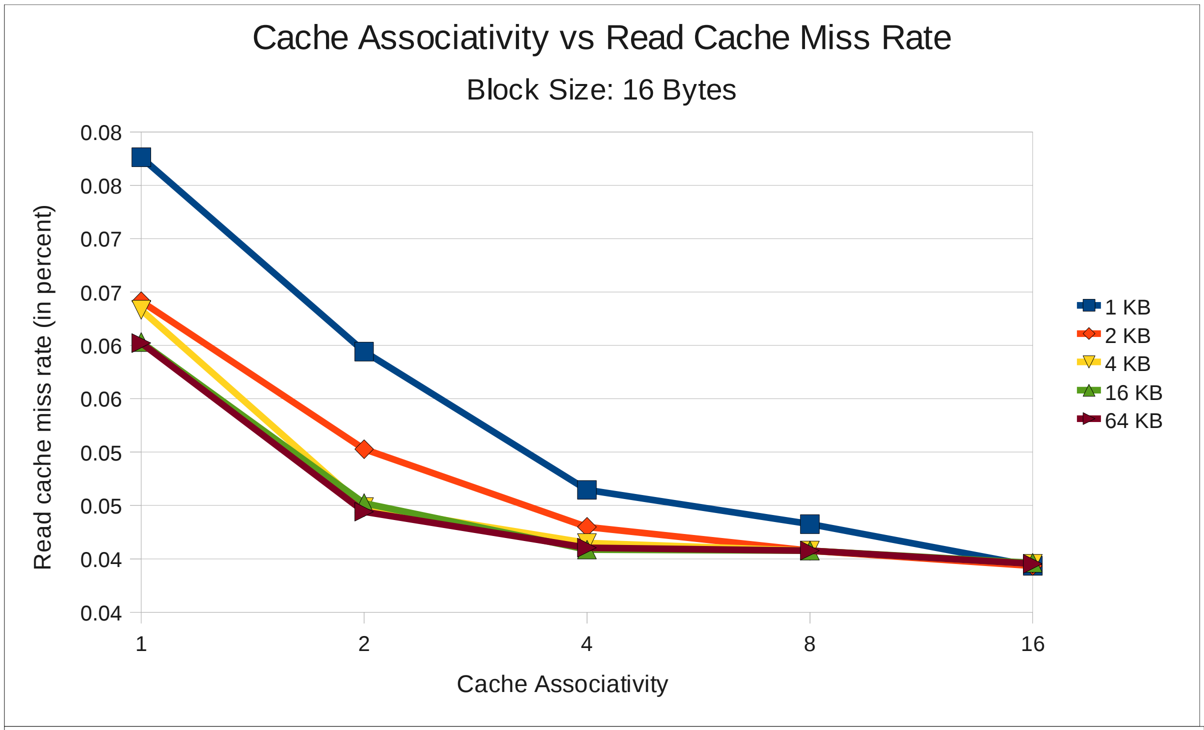
Block Size 16
Total Cache Miss Rate

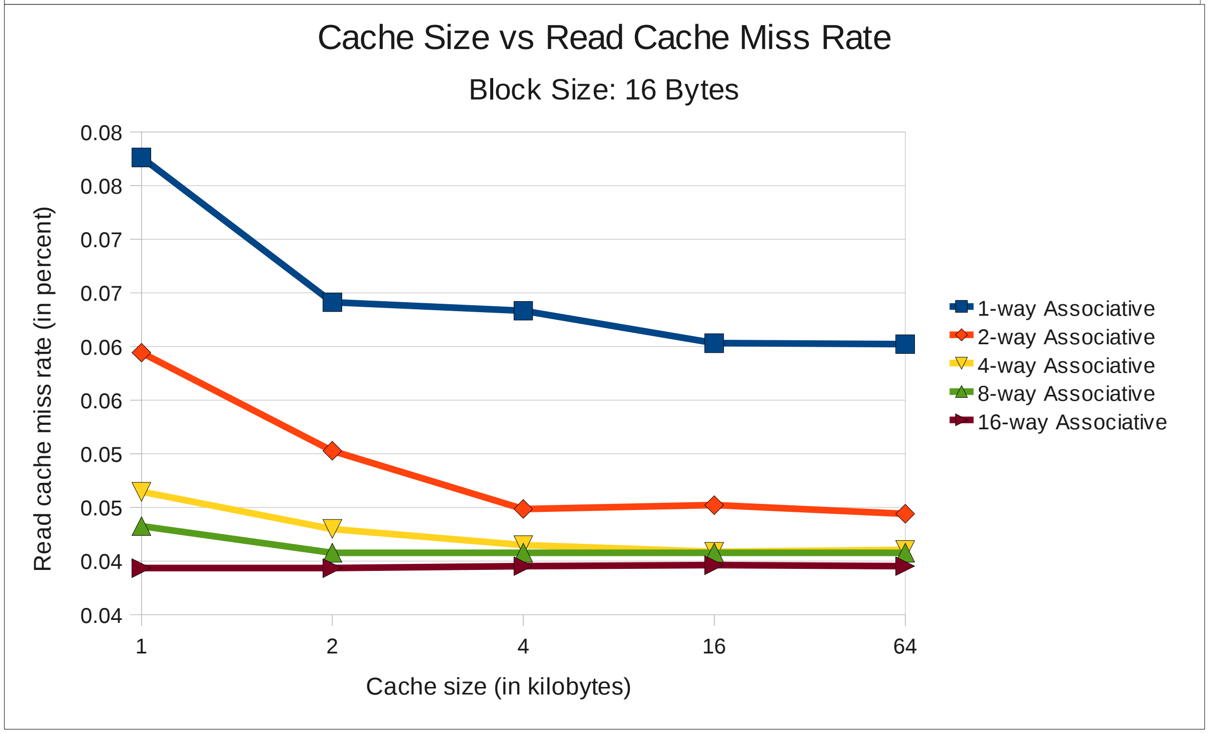




Ian Buitrago
Miguel Diaz
11-30-2011

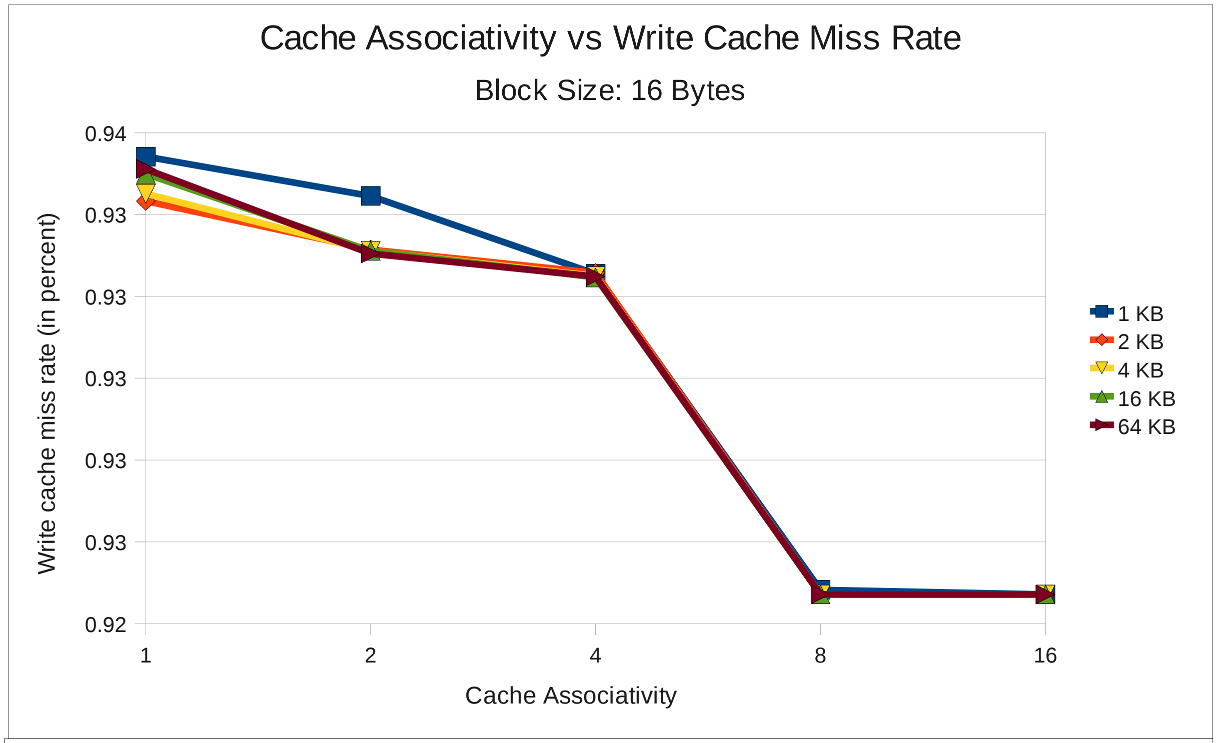
Block Size 16
Reads Cache Miss Rate

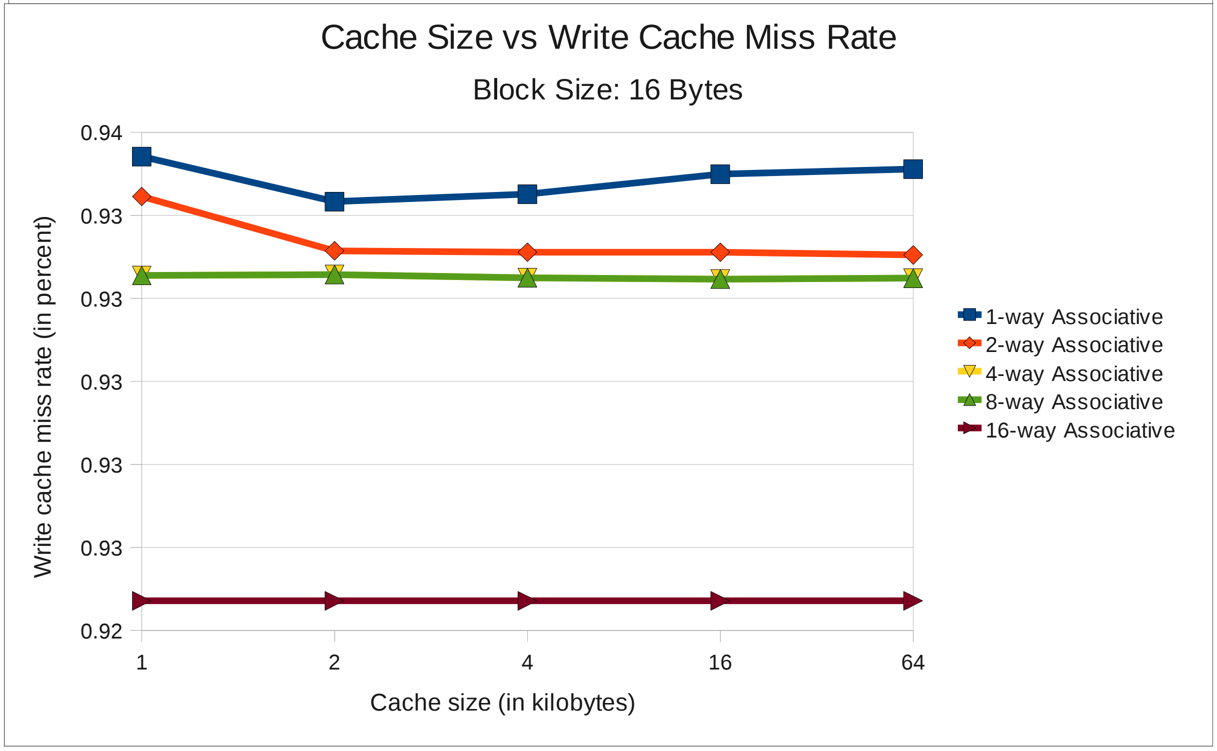




Ian Buitrago
Miguel Diaz
11-30-2011

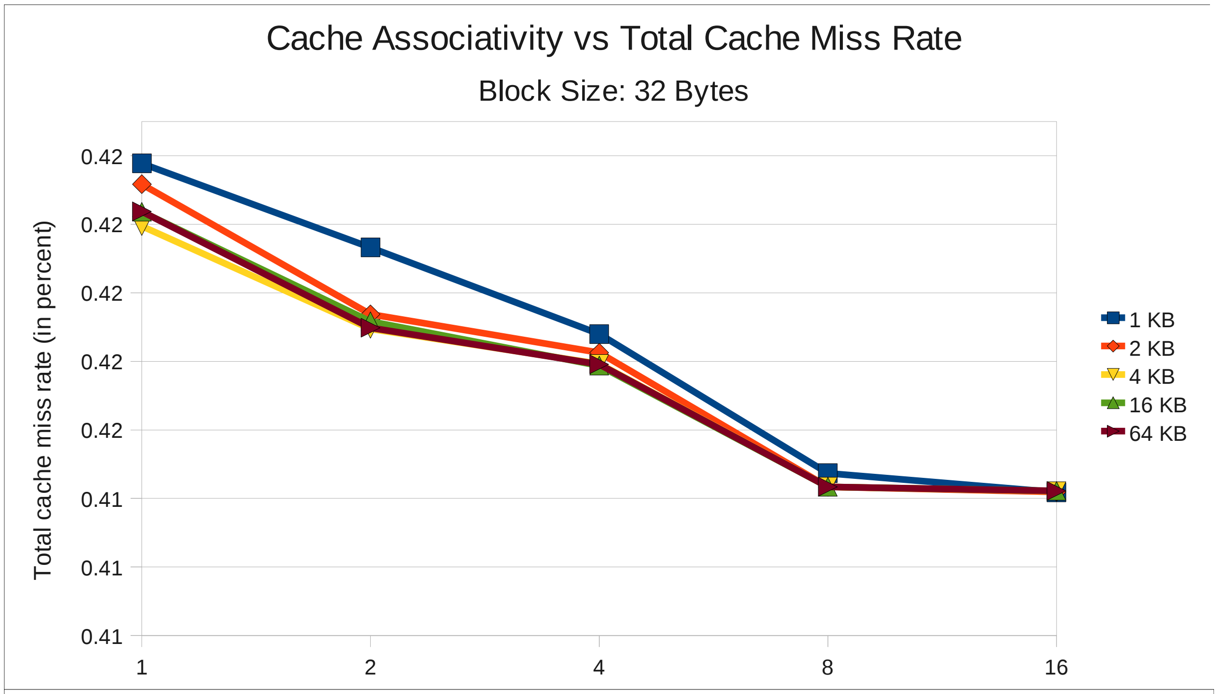
Block Size 16
Writes Cache Miss Rate

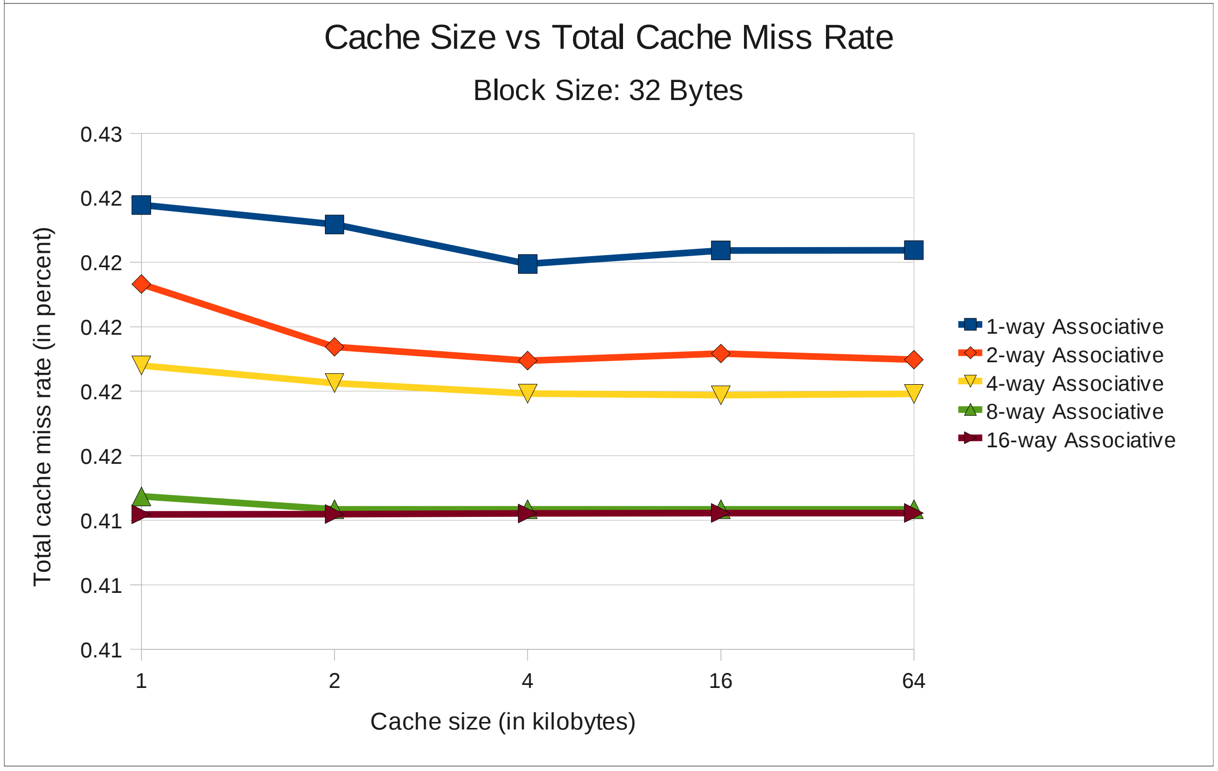




Ian Buitrago
Miguel Diaz
11-30-2011

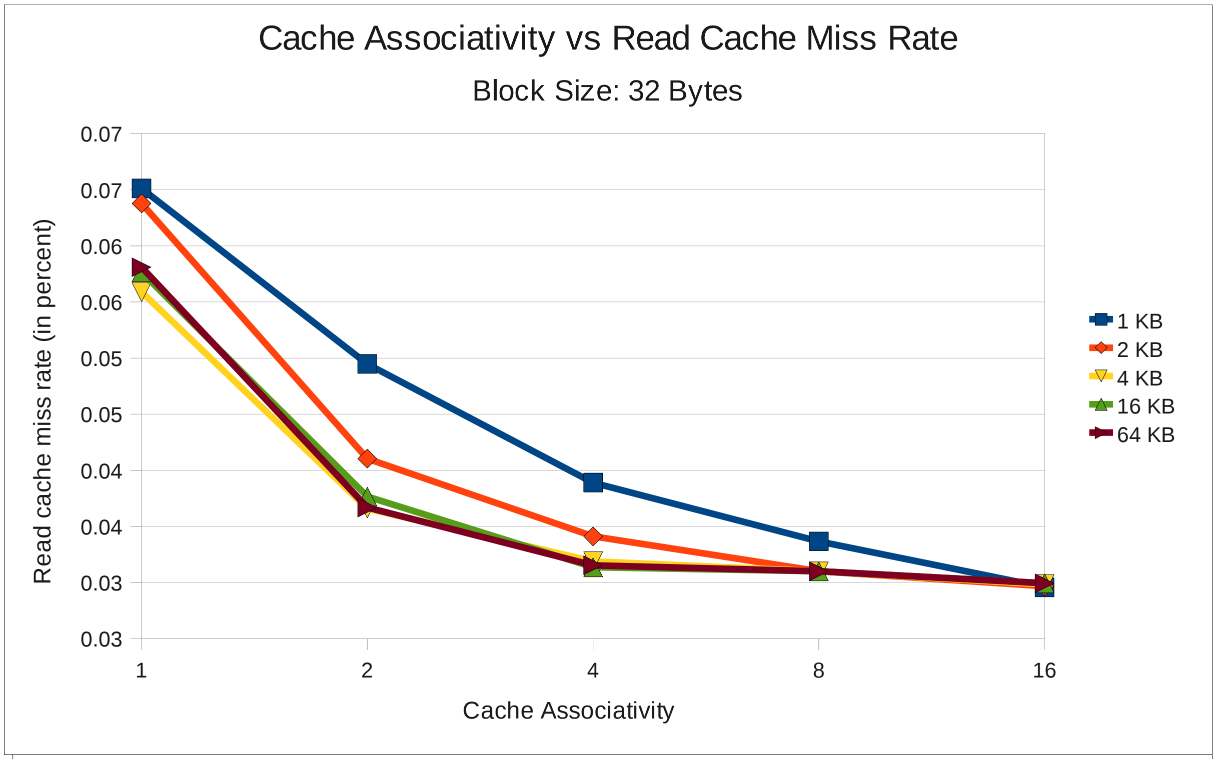
Block Size 32
Total Cache Miss Rate

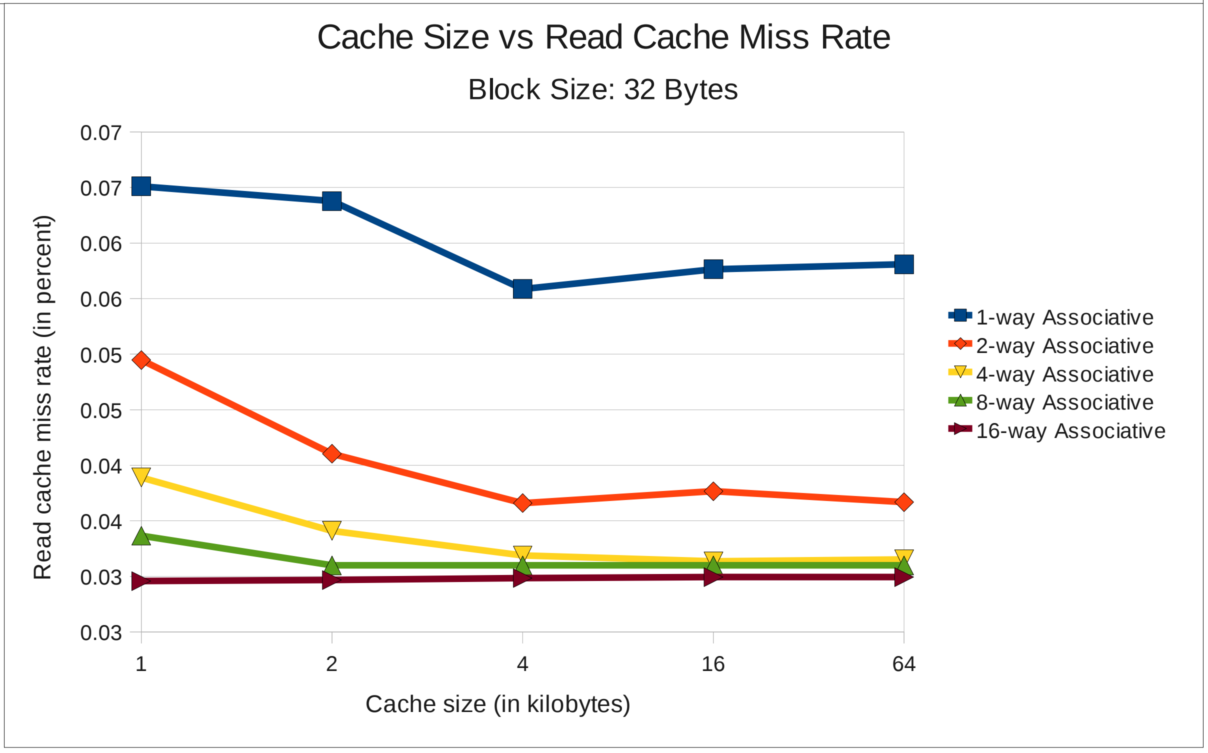




Ian Buitrago
Miguel Diaz
11-30-2011

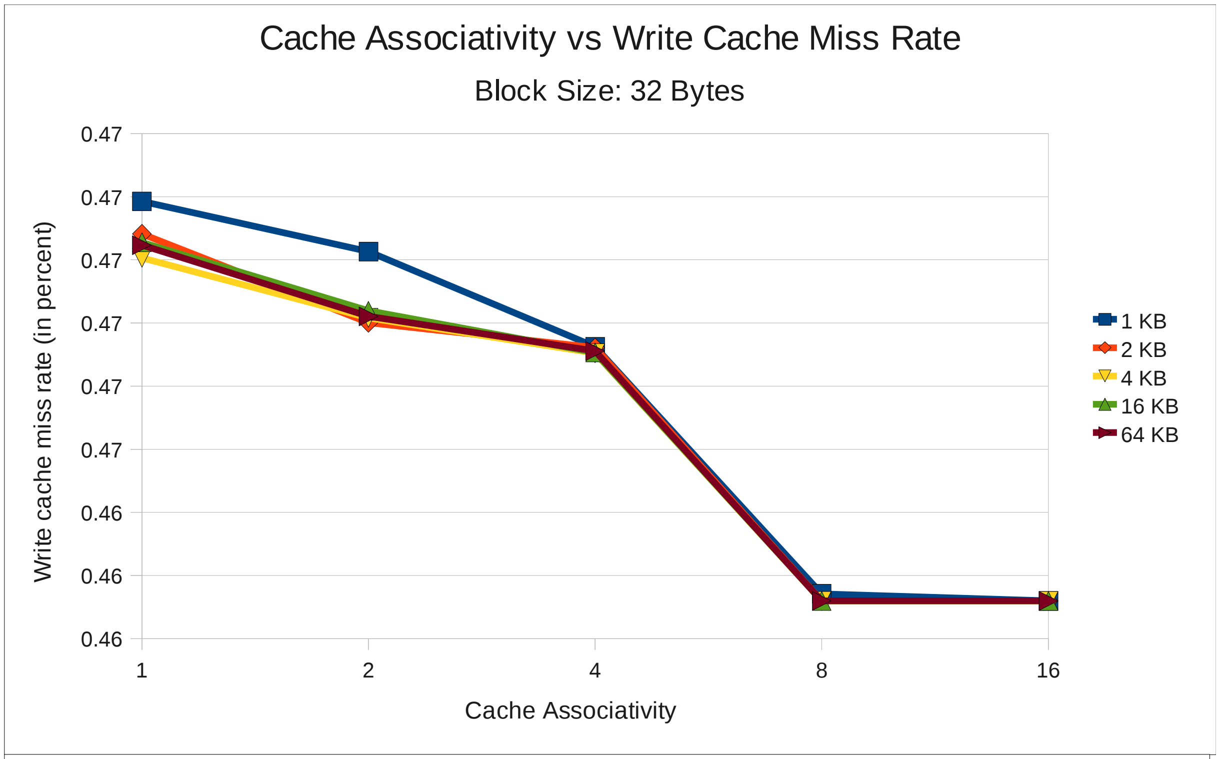
Block Size 32
Reads Cache Miss Rate

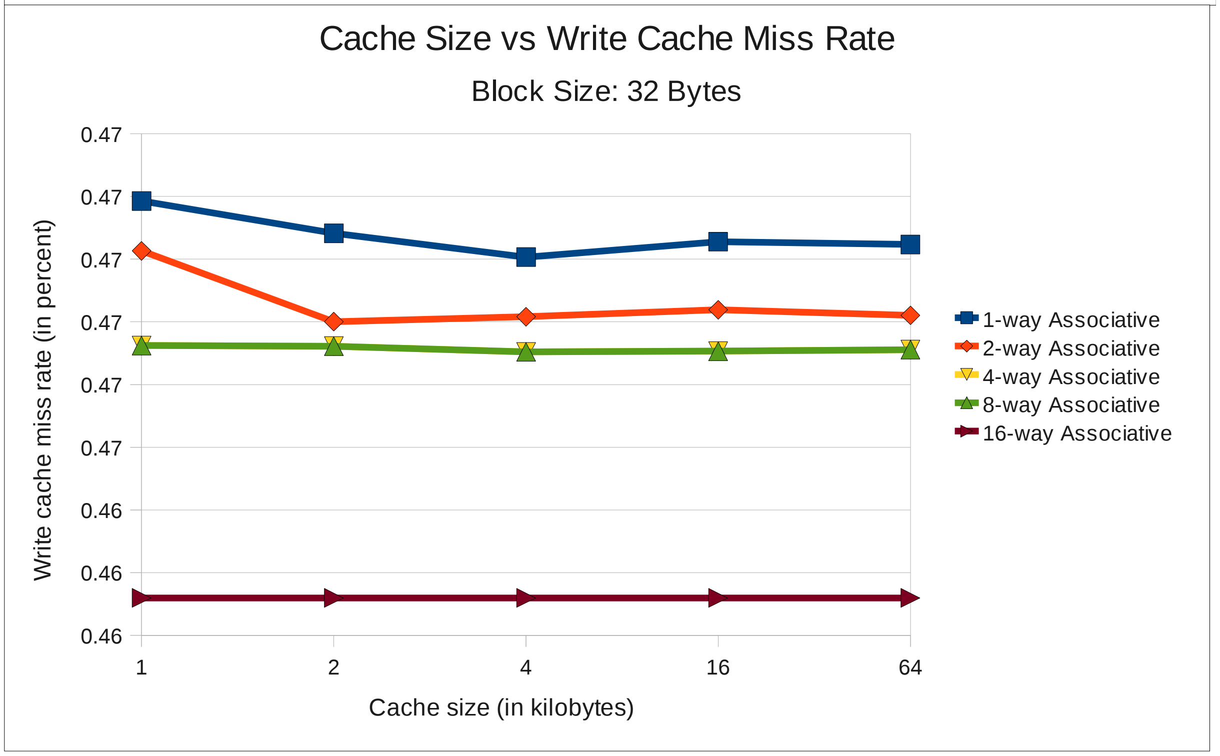




Ian Buitrago
Miguel Diaz
11-30-2011

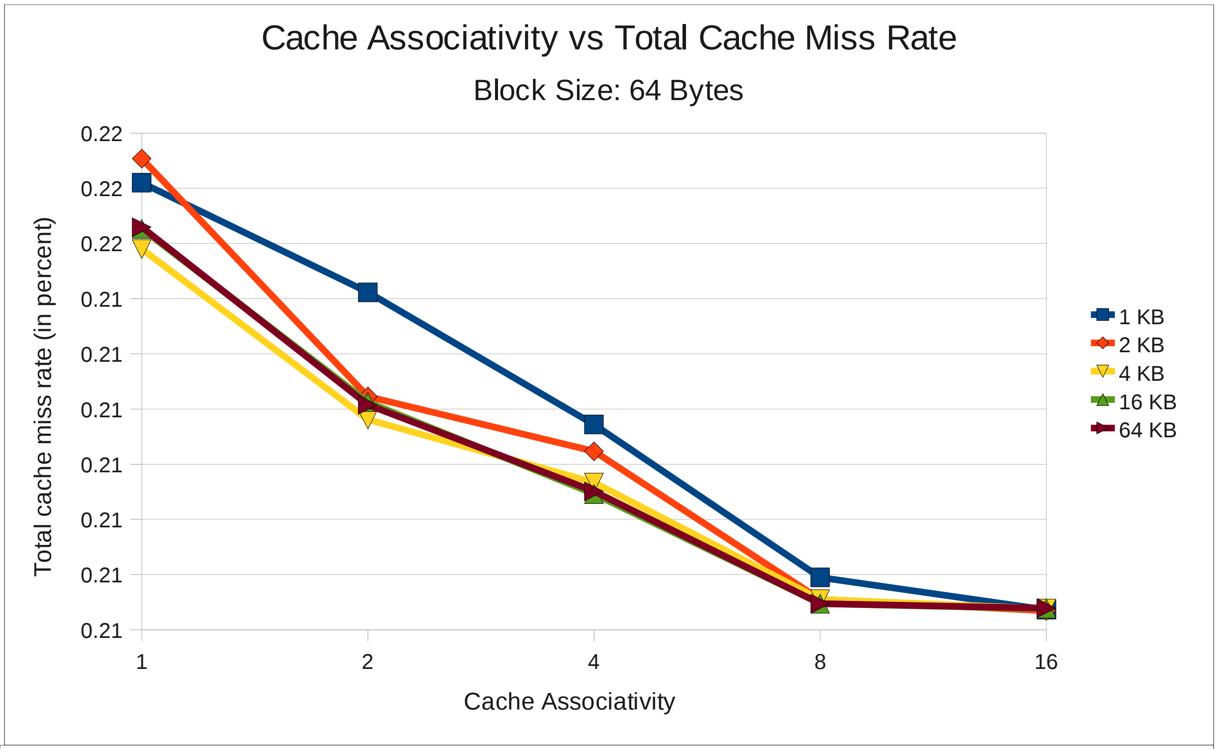
Block Size 32
Writes Cache Miss Rate

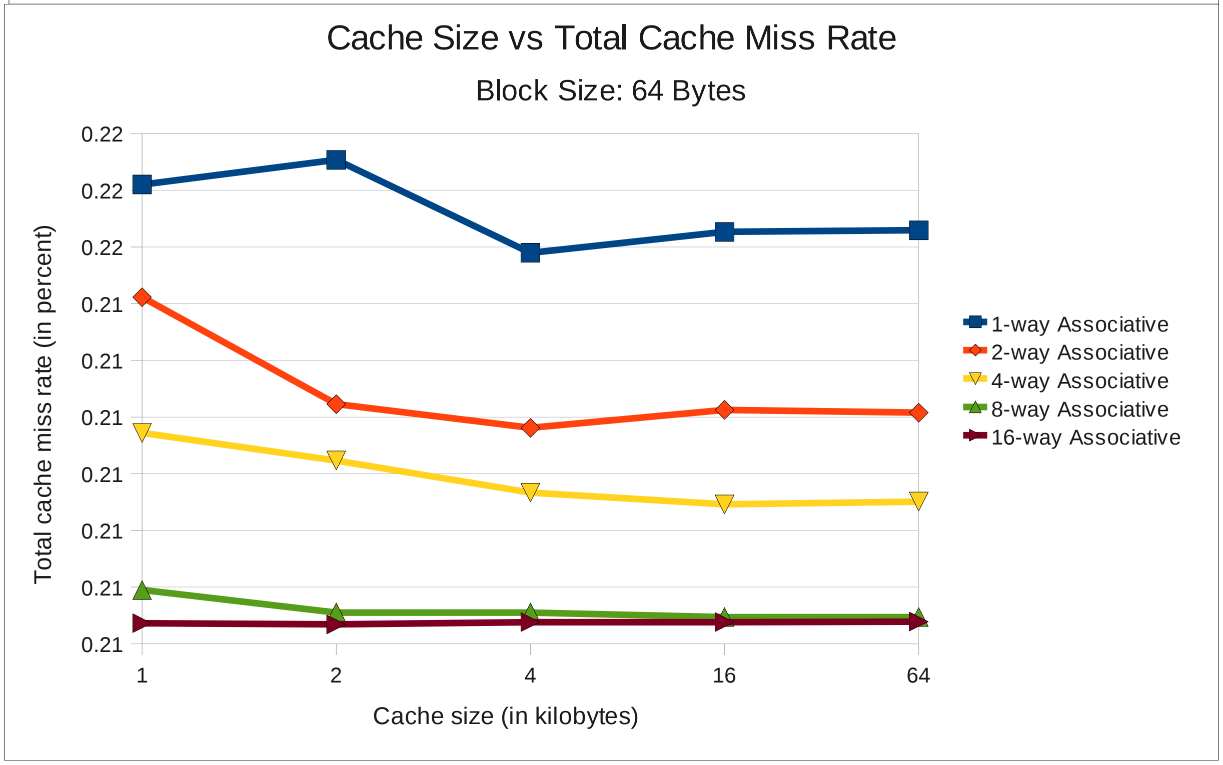




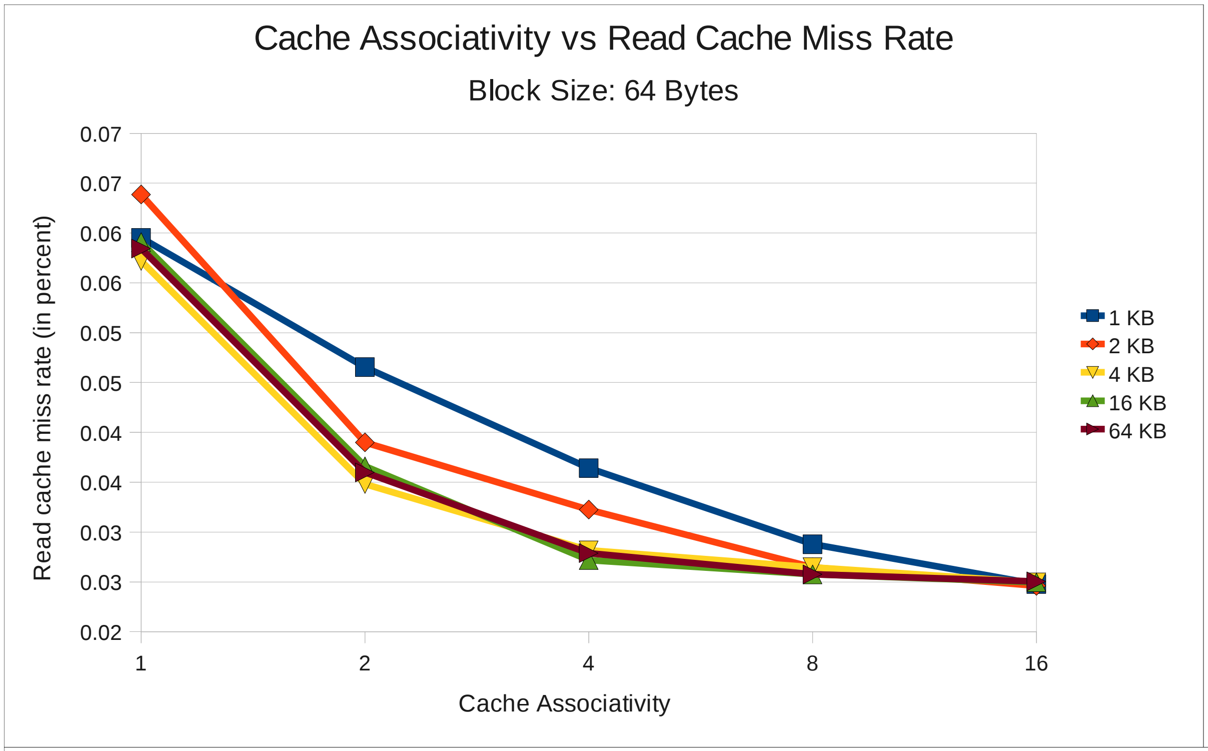
Ian Buitrago
Miguel Diaz
11-30-2011

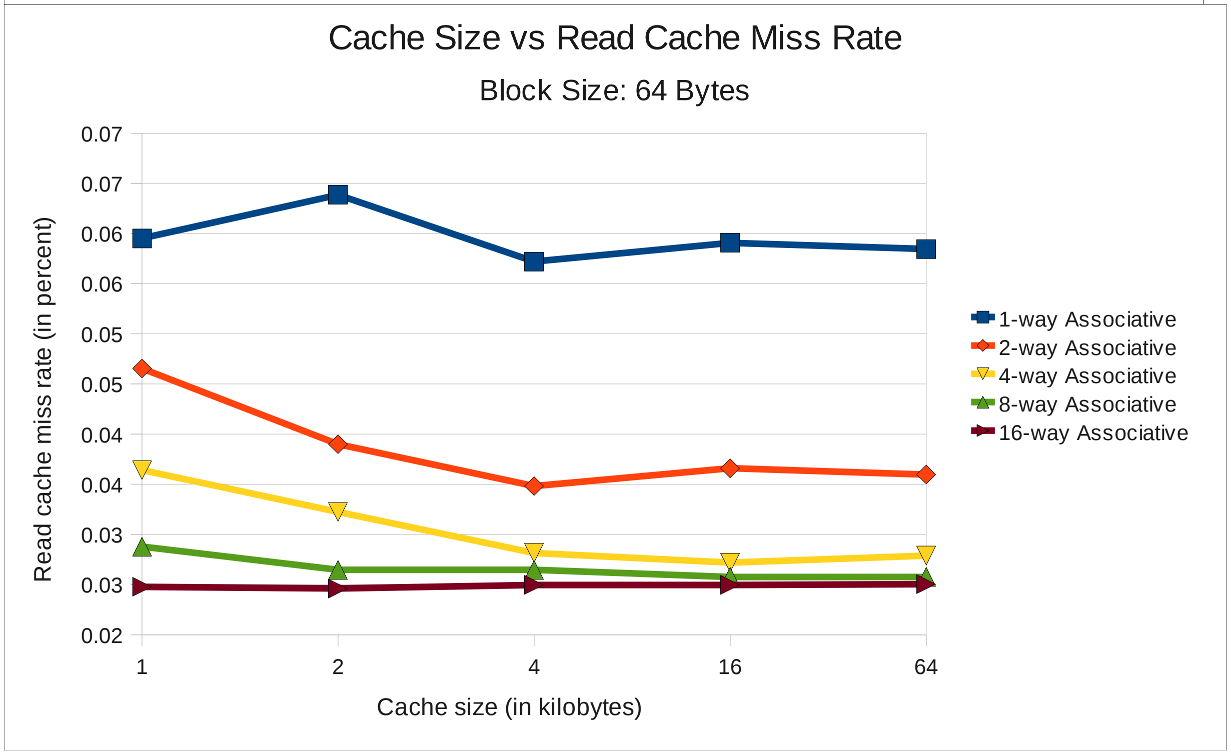
Block Size 64
Total Cache Miss Rate





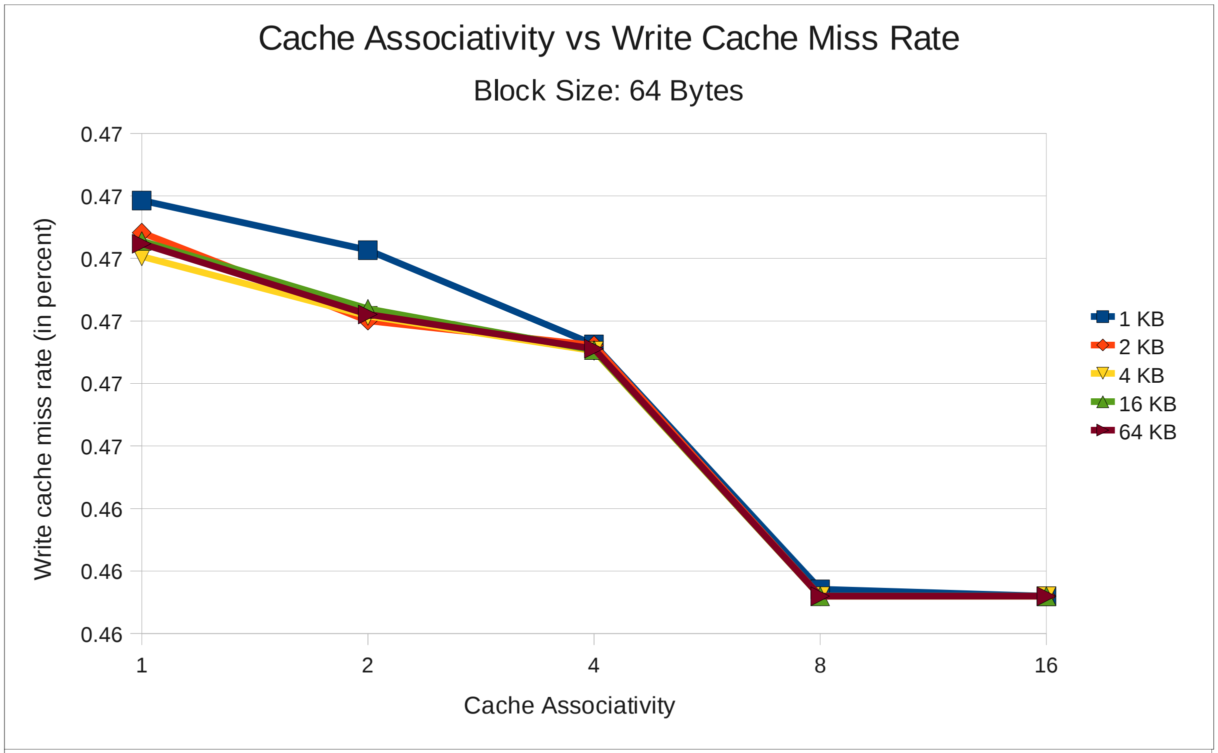
Block Size 64
Reads Cache Miss Rate

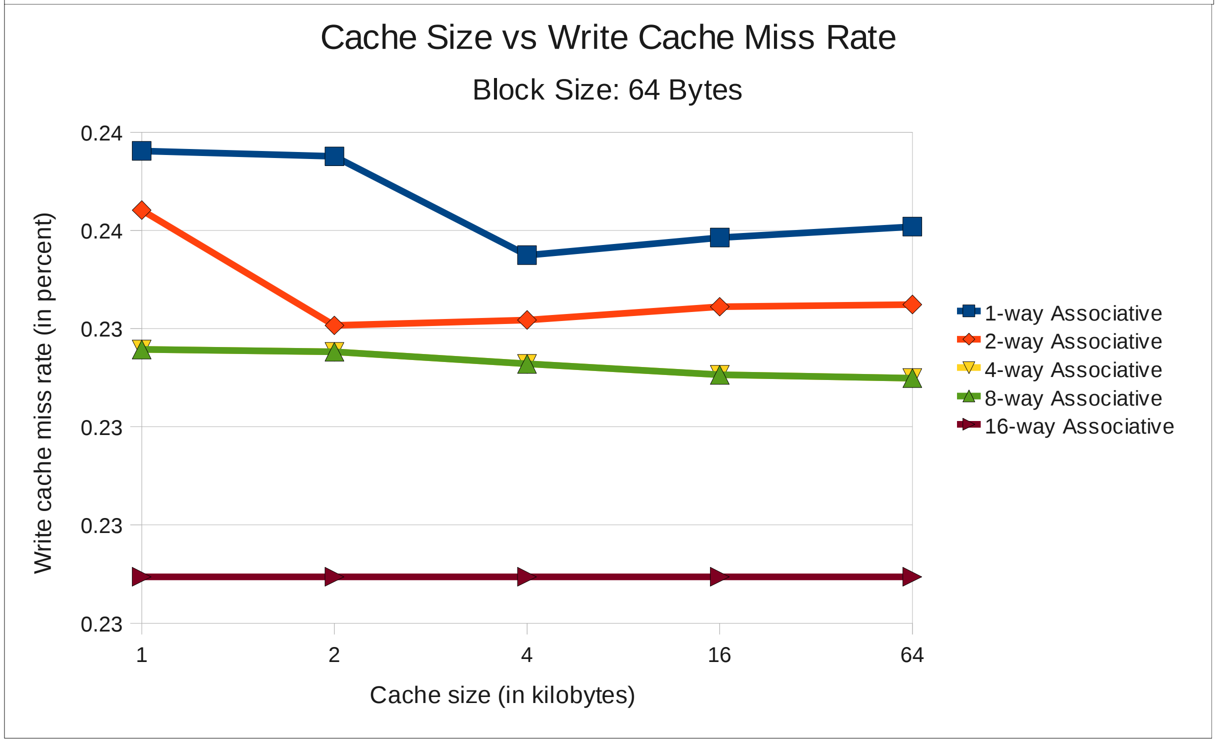




Ian Buitrago
Miguel Diaz
11-30-2011

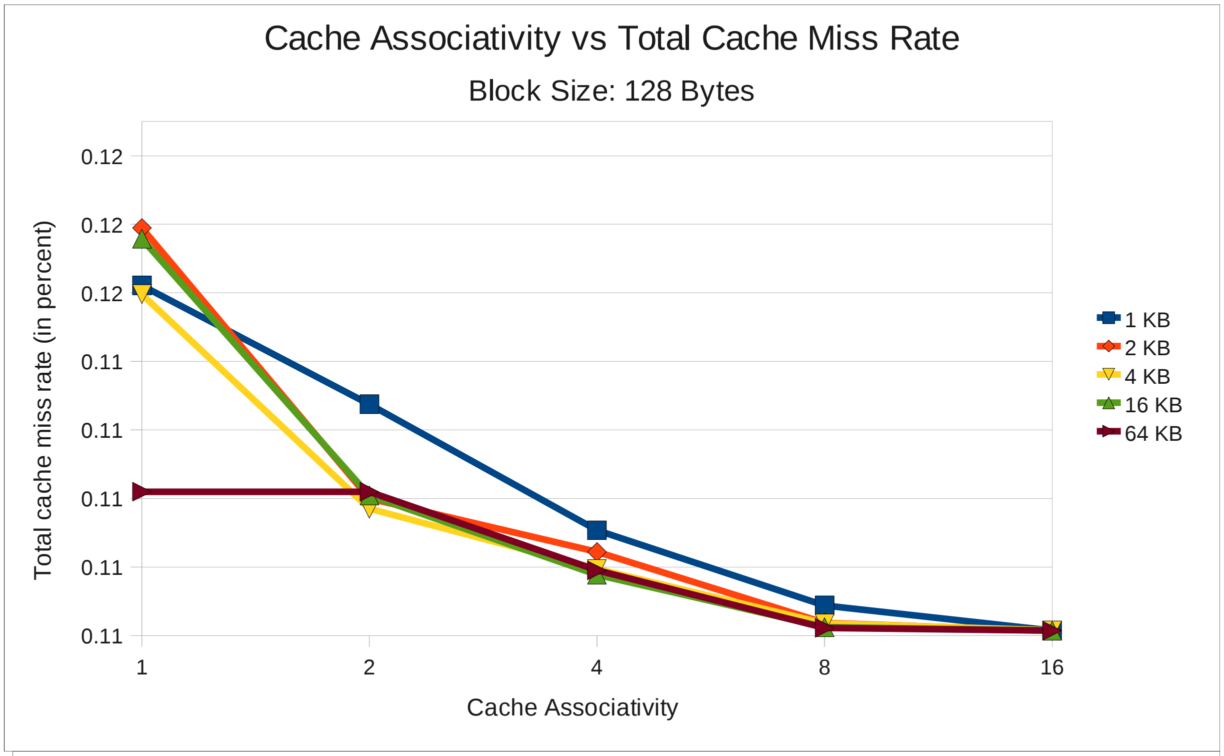
Block Size 64
Writes Cache Miss Rate

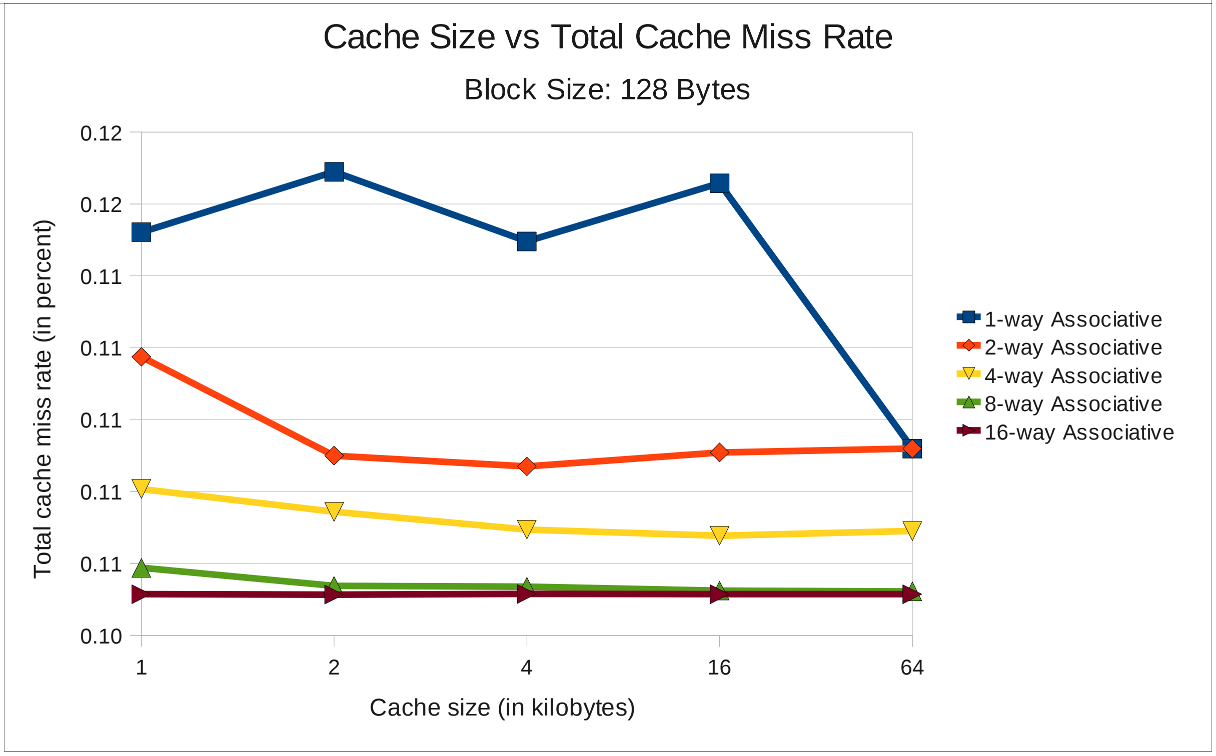




Ian Buitrago
Miguel Diaz
11-30-2011

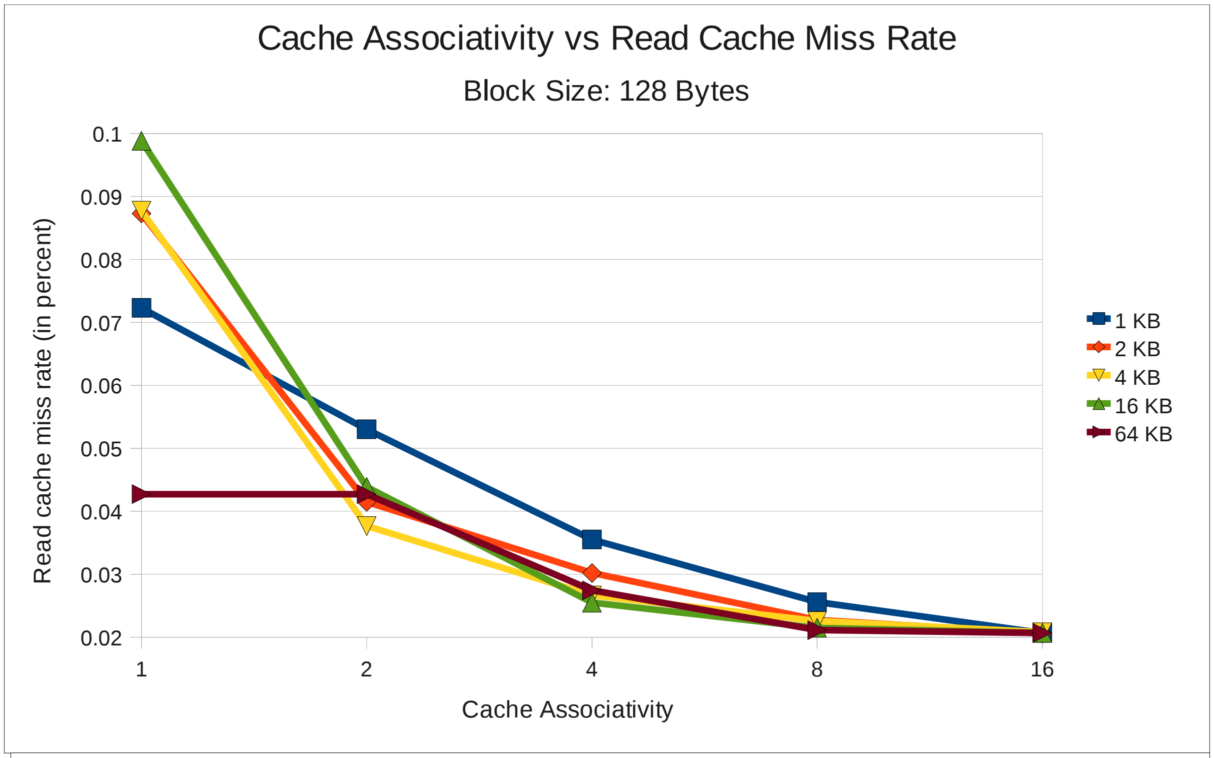
Block Size 128
Total Cache Miss Rate

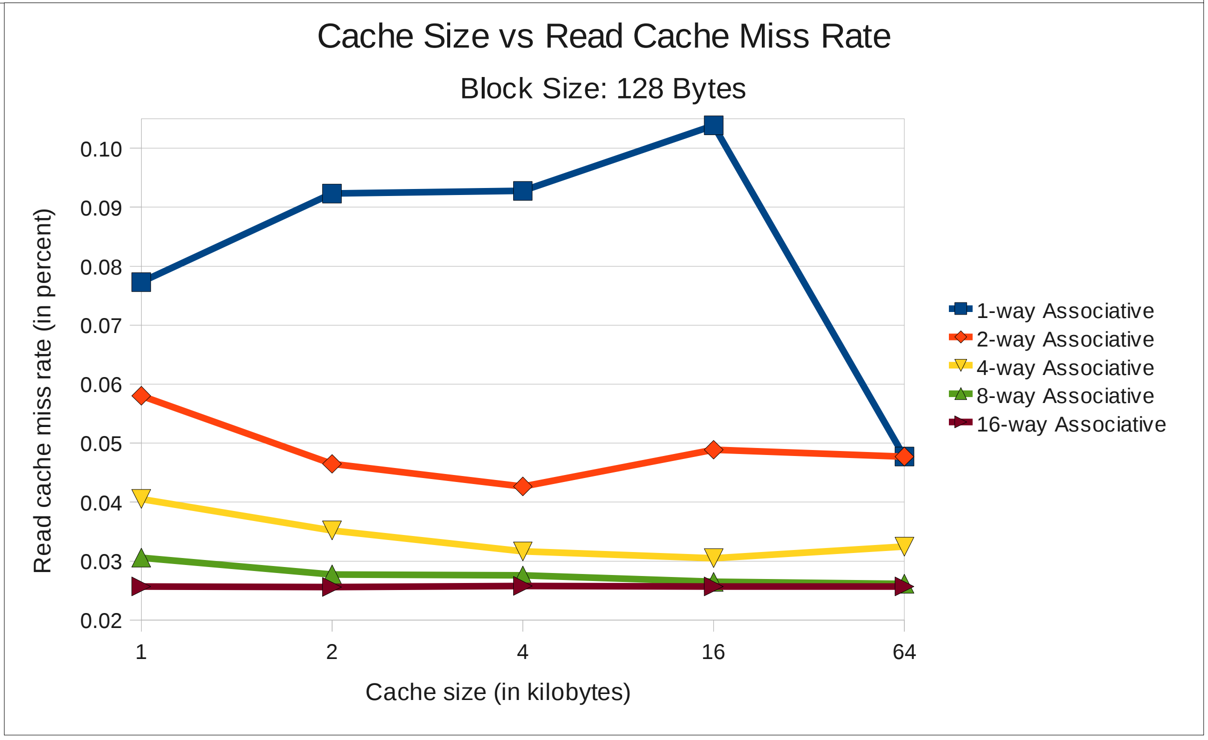




Ian Buitrago
Miguel Diaz
11-30-2011

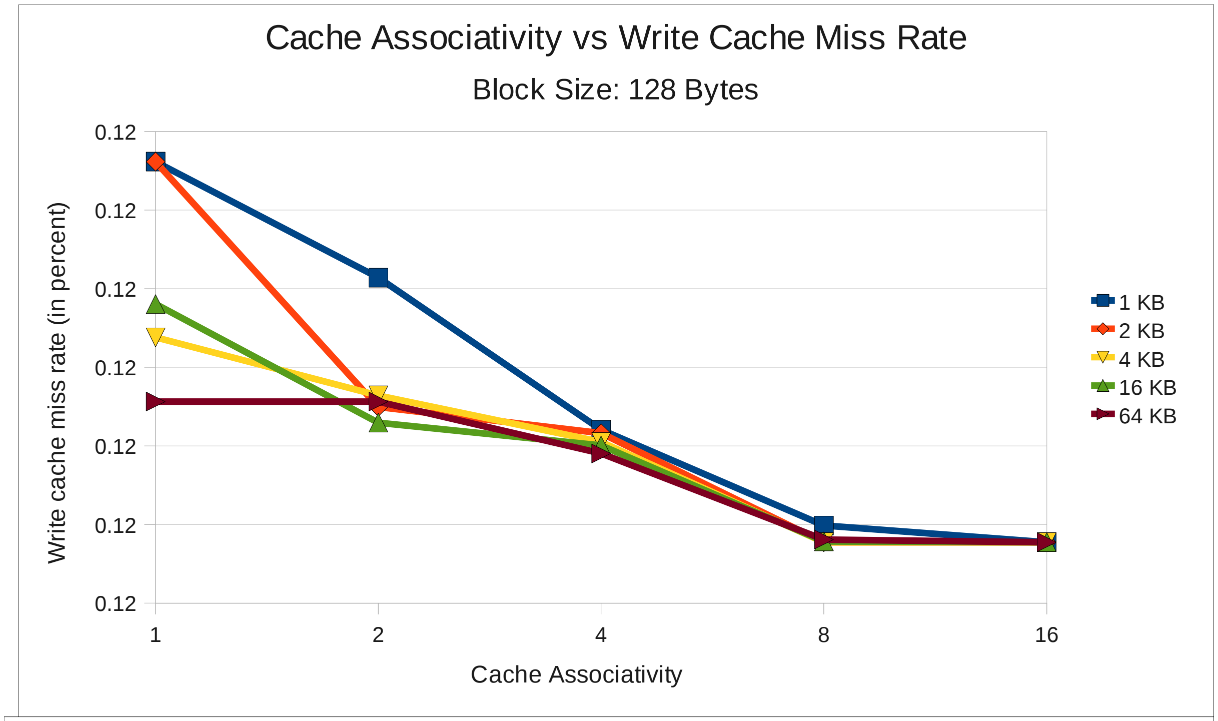
Block Size 128
Reads Cache Miss Rate

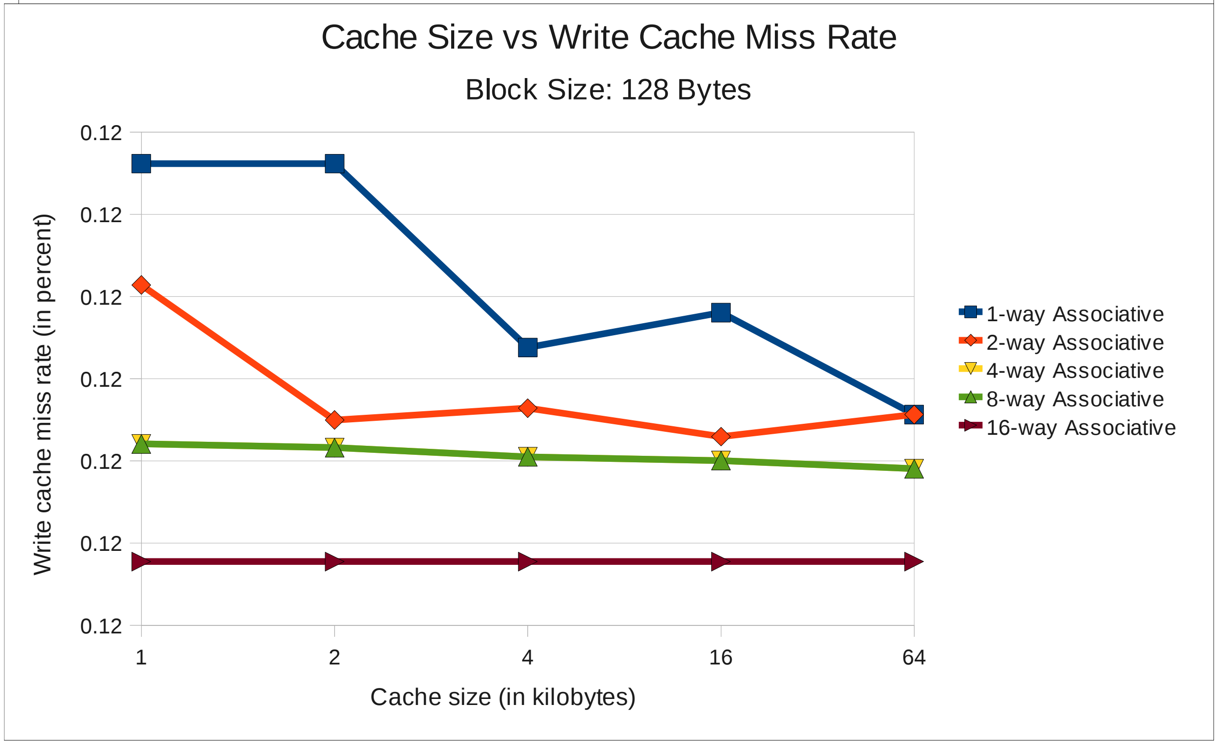




Ian Buitrago
Miguel Diaz
11-30-2011

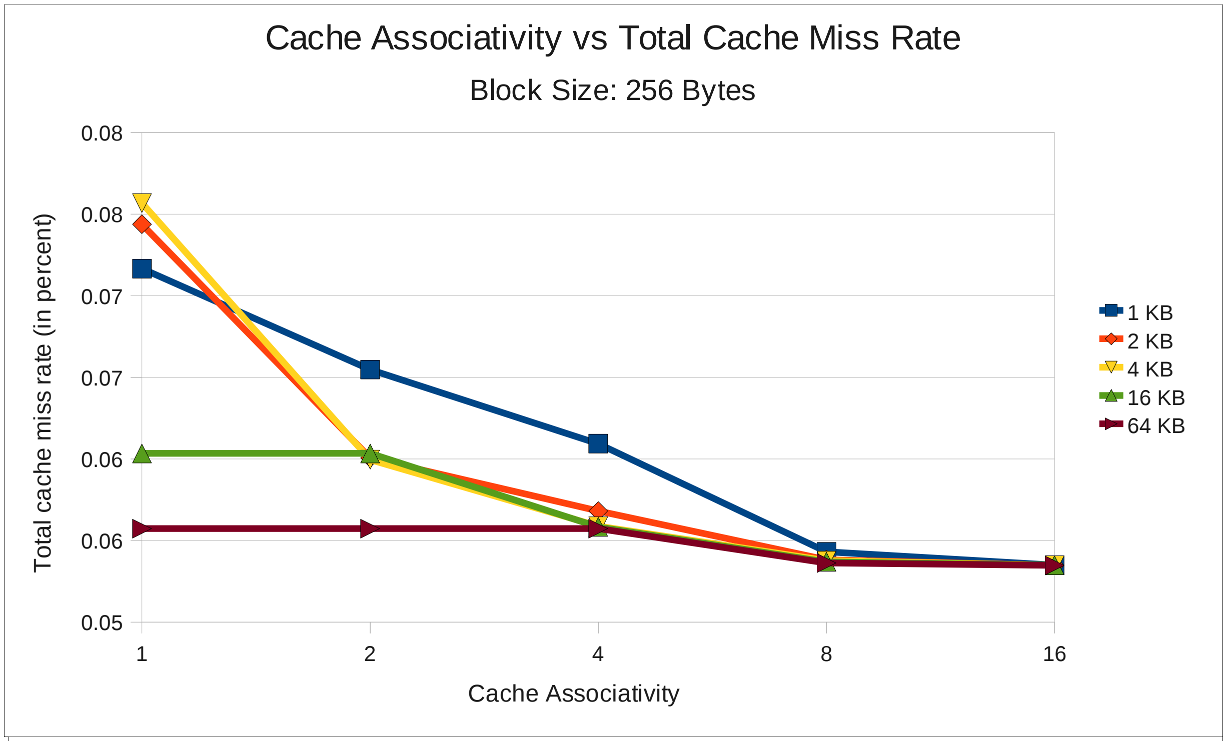
Block Size 128
Writes Cache Miss Rate

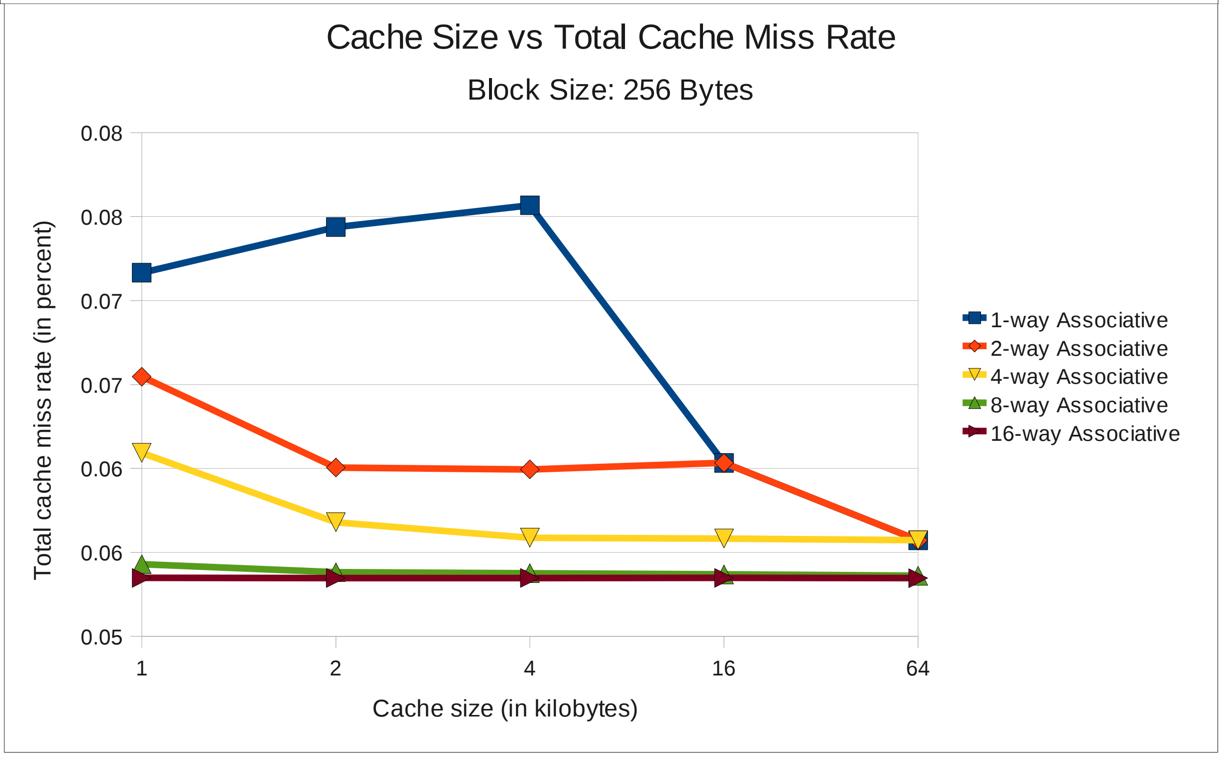




Ian Buitrago
Miguel Diaz
11-30-2011

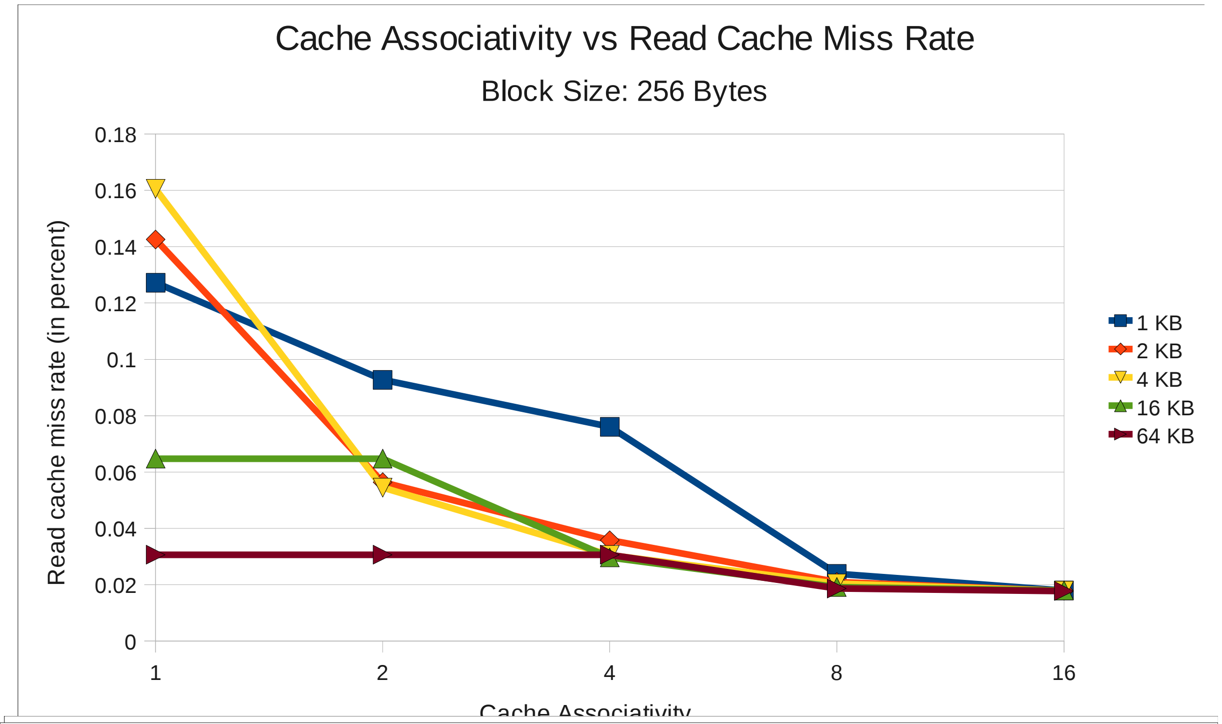
Block Size 256
Total Cache Miss Rate

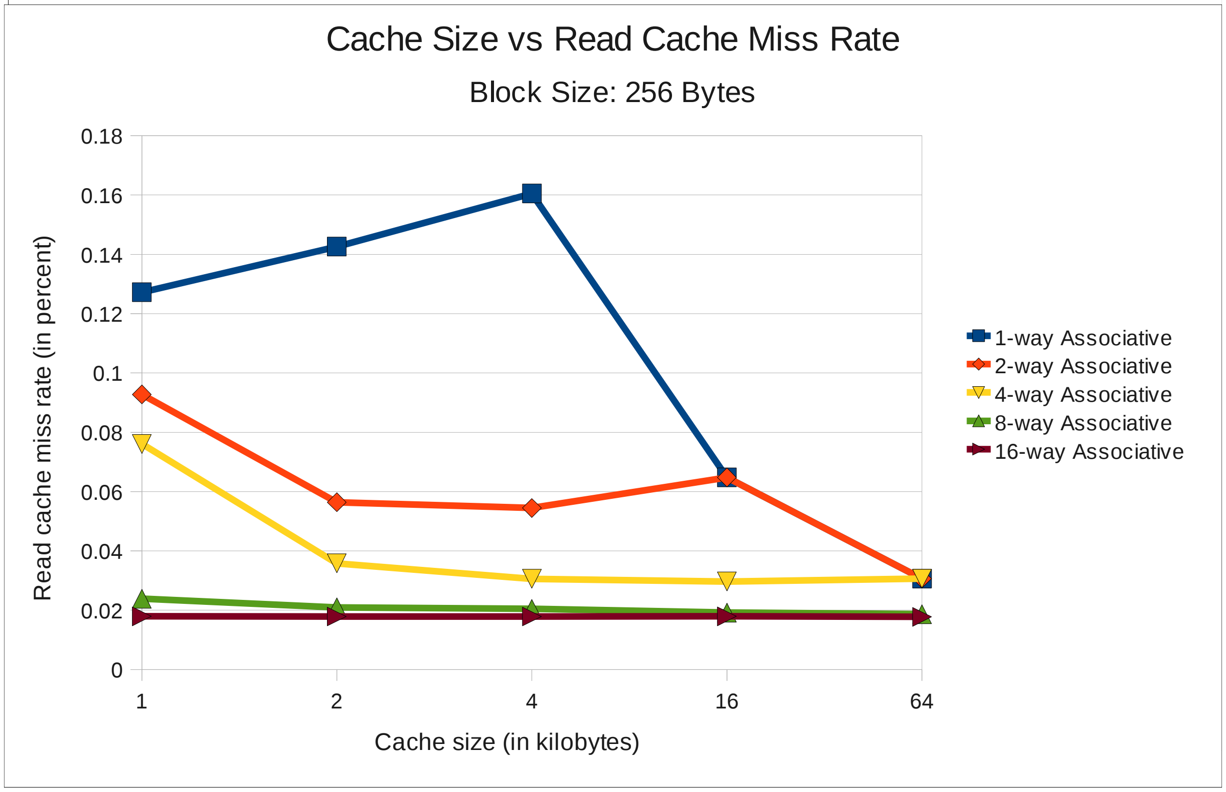




Ian Buitrago
Miguel Diaz
11-30-2011

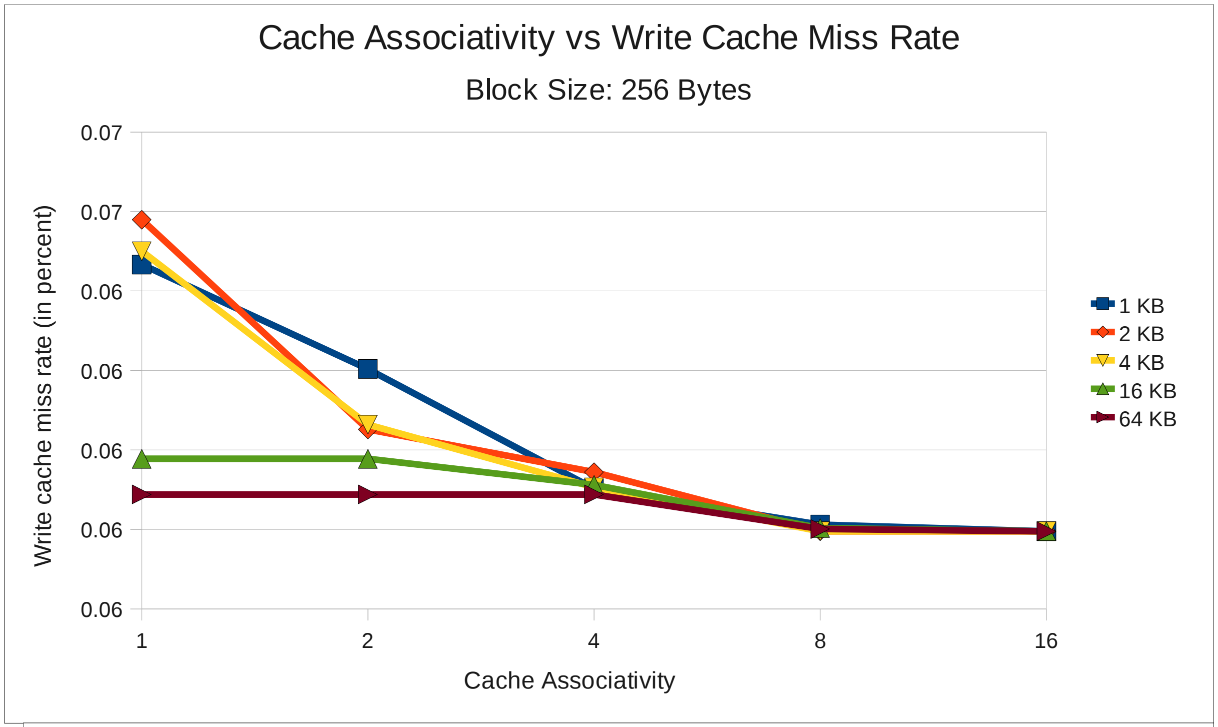
Block Size 256
Reads Cache Miss Rate

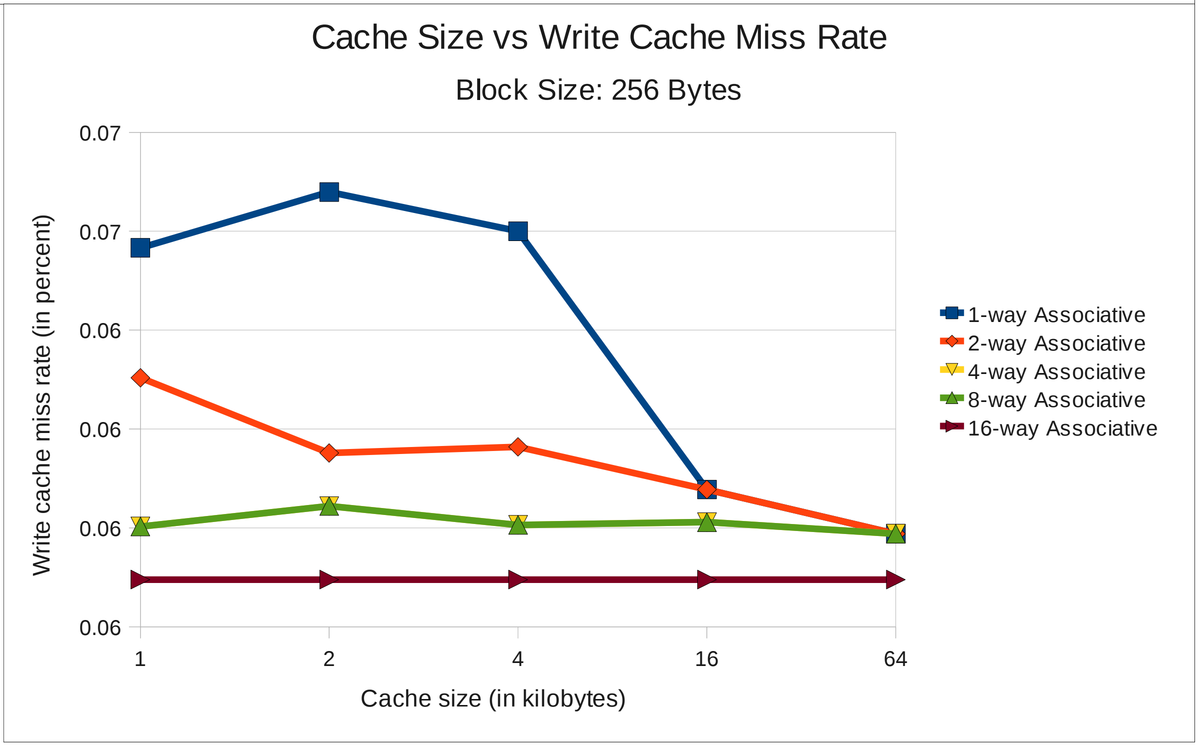




Ian Buitrago
Miguel Diaz
11-30-2011

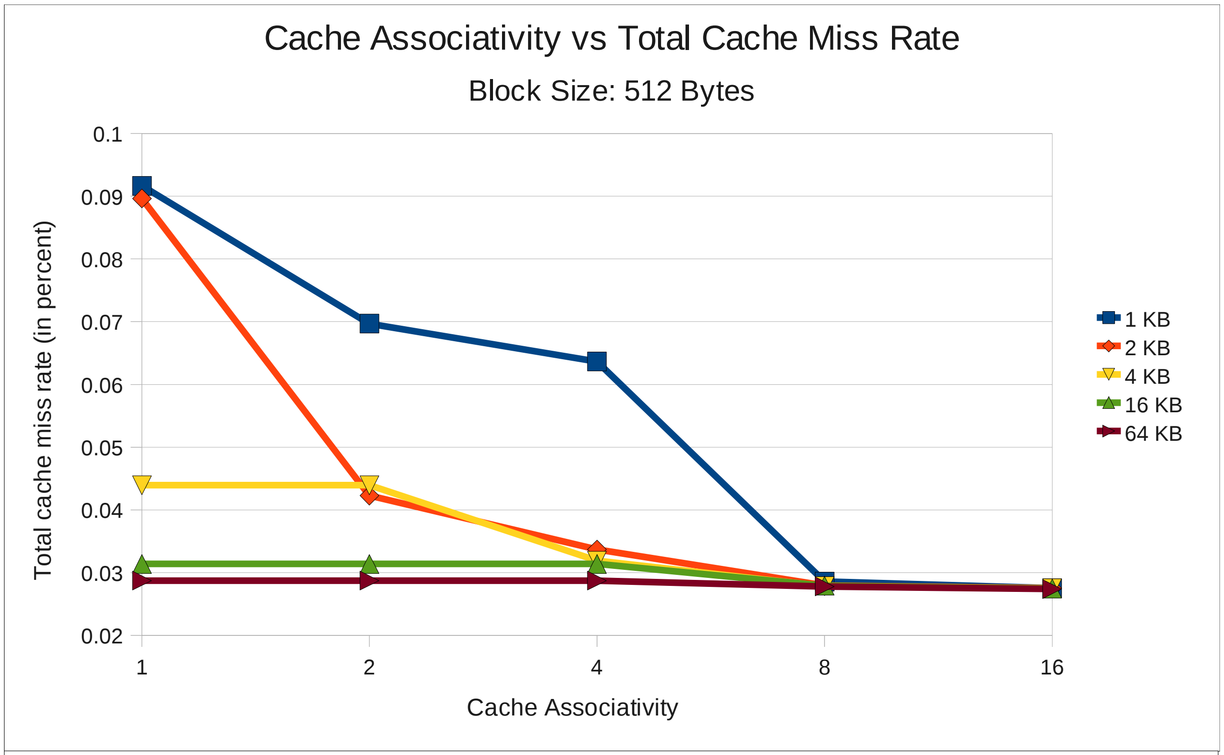
Block Size 256
Writes Cache Miss Rate

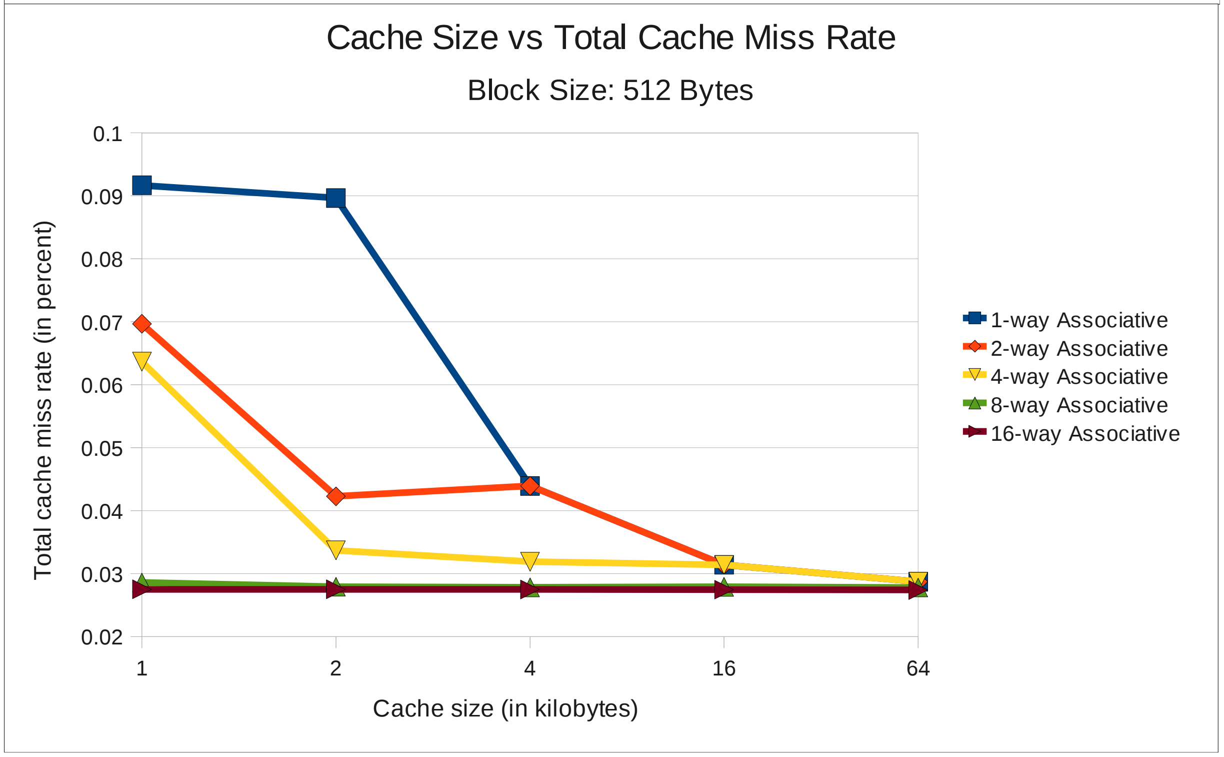




Ian Buitrago
Miguel Diaz
11-30-2011

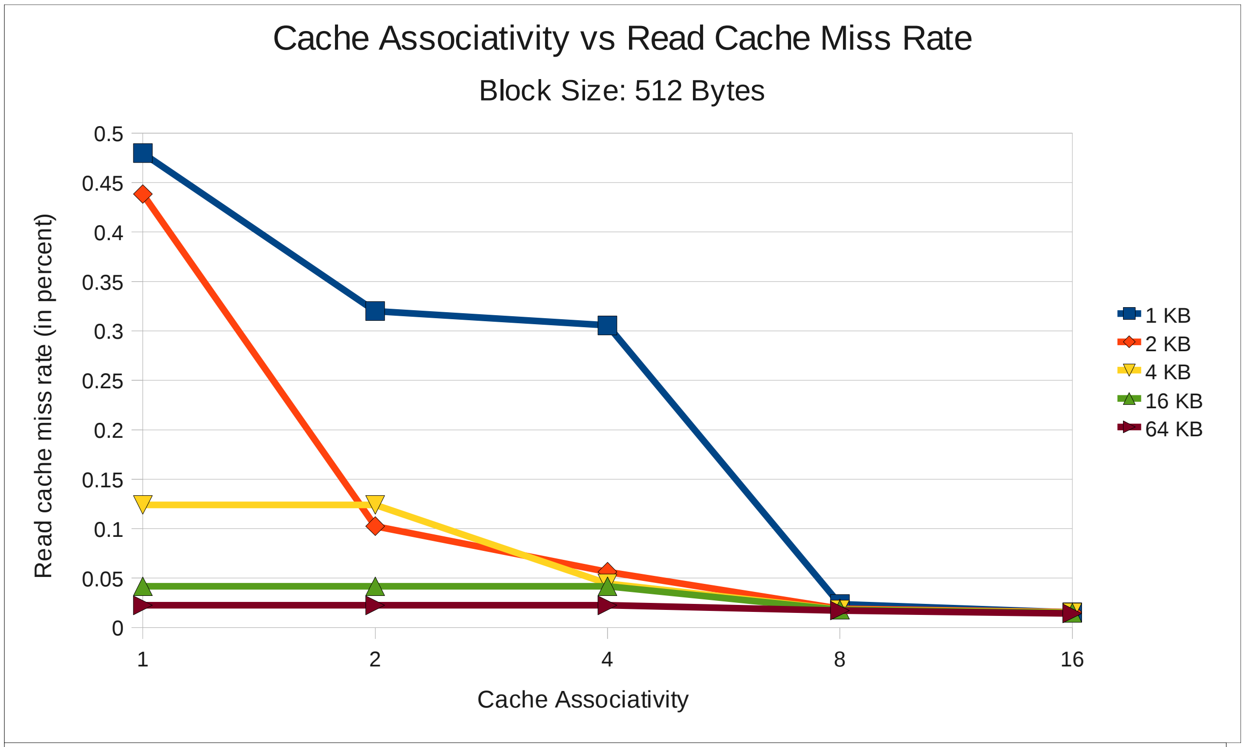
Block Size 512
Total Cache Miss Rate

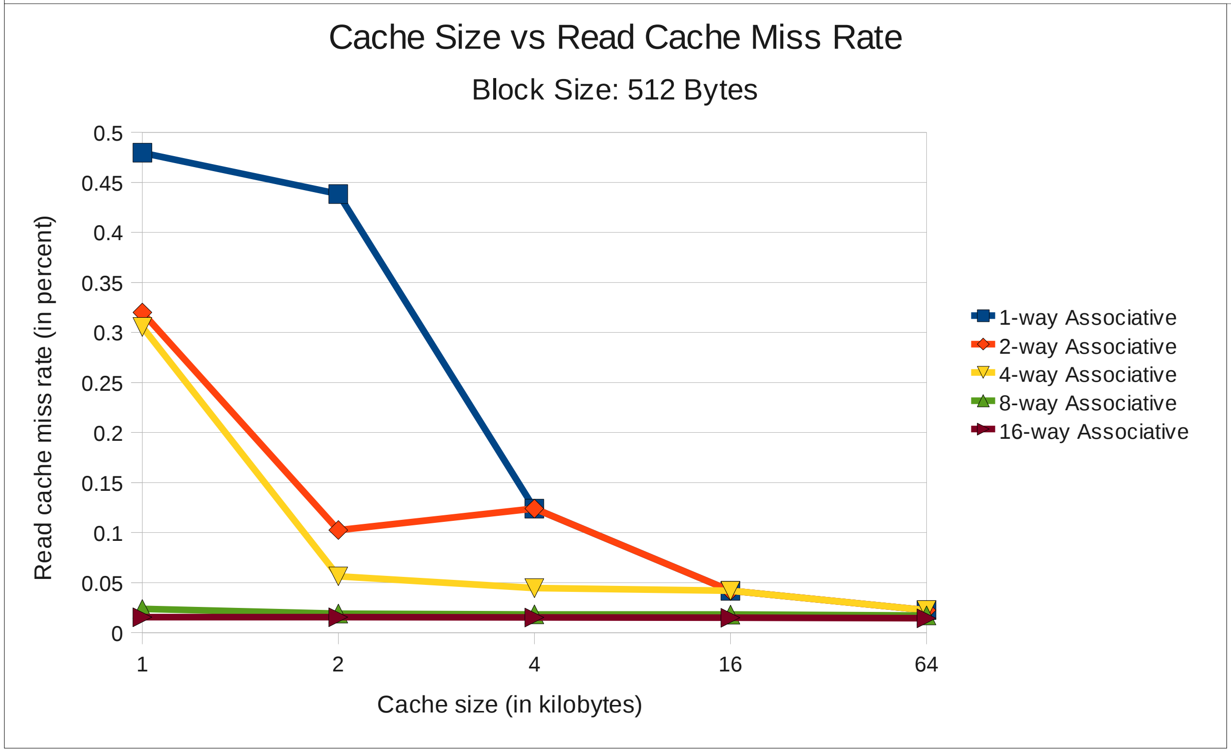




Ian Buitrago
Miguel Diaz
11-30-2011

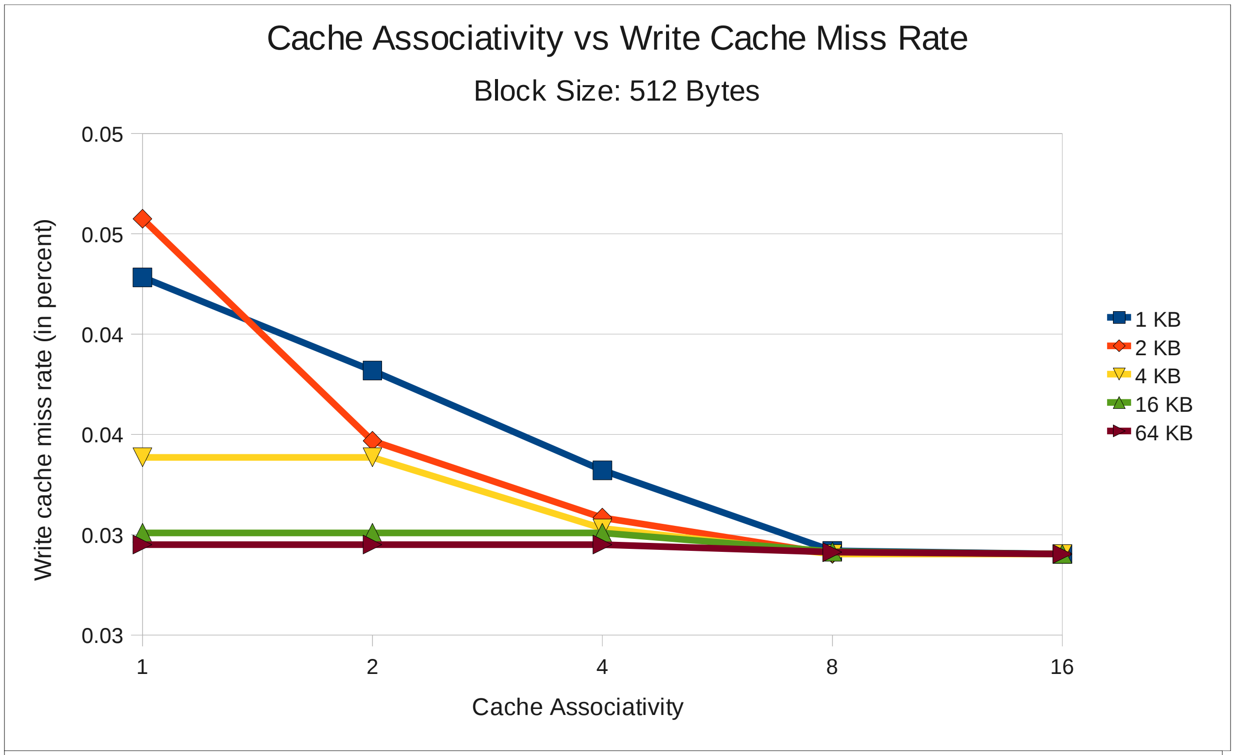
Block Size 512
Reads Cache Miss Rate

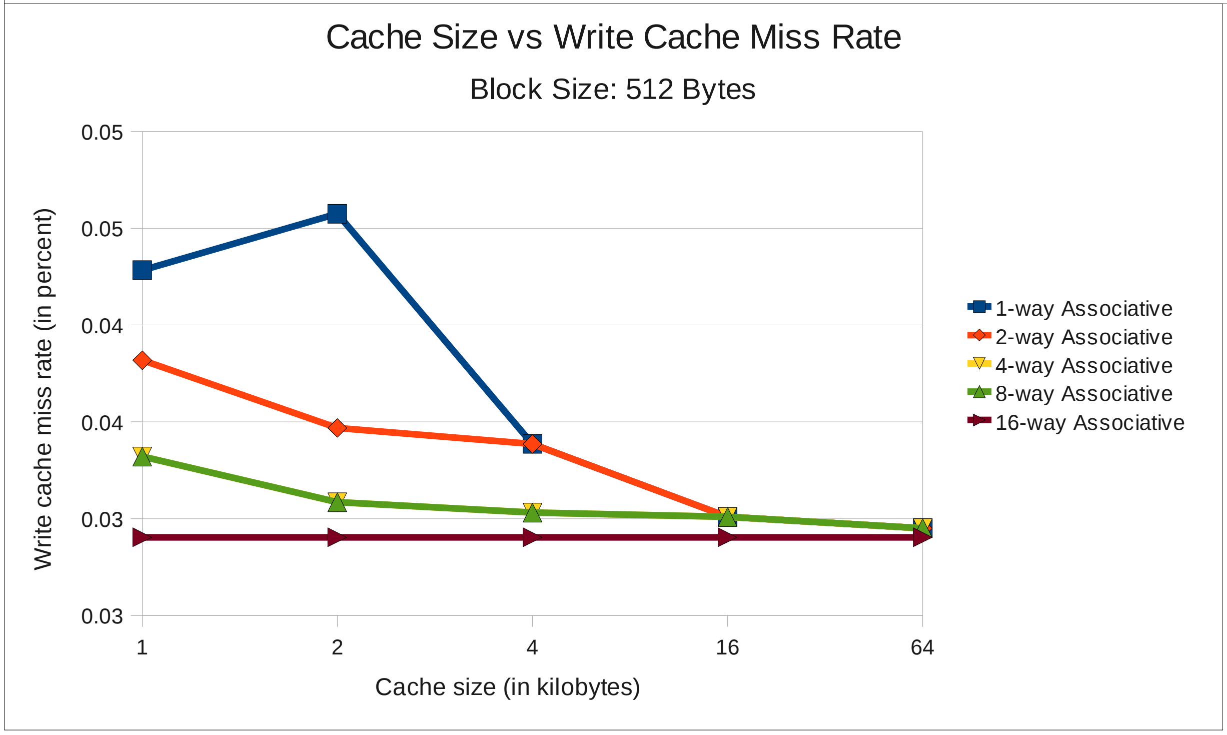




Ian Buitrago
Miguel Diaz
11-30-2011

Block Size 512
Writes Cache Miss Rate


Cache Associativity vs Write Cache Miss Rate
Block Size: 512 Bytes
Write cache miss rate (in percent)
Cache Associativity
1 KB
2 KB
4 KB
16 KB
64 KB


Cache Size vs Write Cache Miss Rate
Block Size: 512 Bytes
Write cache miss rate (in percent)
Cache size (in kilobytes)
1-way Associative
2-way Associative
4-way Associative
8-way Associative
16-way Associative