TDTS08: Lab Report

Lab 1: Cache Memories

Name	**PIN**	**Email**
Alexander Yngve	930320-6651	aleyn573@student.liu.se
Pål Kastman	851212-7575	palka285@student.liu.se

Contents

1 Introduction

The purpose of this lab was to understand the functionality of cache memories, and to get an insight into various trade-offs related to the design of systems with cache memories.

We analyze caches in sizes ranging from 128kB to 4096kB and with associativity from 1 to 8.

2 Cache basics

2.1 Problem 1

How is a 8-bit memory adress divided into tag, line number and byte number?

$$s = \log_2(\text{no. of blocks}) = \log_2(64) = 6$$
$$r = \log_2(\text{no. of lines in cache}) = \log_2(8) = 3$$
$$w = \log_2(\text{block size}) = \log_2(4) = 2$$

$$\text{size of tag} = (s - r) = 3\text{bits}$$
$$\text{size of line number} = r = 3\text{bits}$$
$$\text{size of byte number} = w = 2\text{bits}$$

2.2 Problem 2

Into what line would bytes with each of the following addresses be stored?

0001 1011 → line 6

0011 0100 → line 5

1101 0000 → line 4

1010 1010 → line 2

2.3 Problem 3

Suppose the byte address 1010 0001 is stored in the cache. What are the addresses of the other bytes stored in the cache?

1010 0000
1010 0010
1010 0011

2.4 Problem 4

How many total bytes of memory can be stored in the cache?

32 bytes

2.5 Problem 5

Why is the tag also stored in the cache?

To associate the cacheline with an address and to calculate hit or miss.

3 Locality of data

Here we ran tests on two different architectures and with two different configurations. cache1 has 1024 cachelines with each of them being 16 bytes long, its associativity being 1 which means it's directly mapped into the main memory.

cache2 also has 1024 cachelines with each of them being 8 bytes long, but its associativity is instead 2. This means that every memory cell in the main memory can store data into two different places in the cache, this means that cache1 and cache2 are still the same size.

Both caches are using the least recently used algorithm for deciding what data to store and what to overwrite.

The test files are doing the same thing but in different order, the test1 file is going through the array A two times, the test2 file though is going through the list one time and first adding the value in position *A[i]* and then the value in position *A[i + CACHE_SIZE]*. The test2 is going to cause a cache miss every time for cache1 since we are going outside the cache size every other time but for cache2 this won't be a problem since it has associativity 2 and therefore can save to two different places in the cache. For the test1 on the other hand, cache1 is going to perform better because we don't have to reload the cache as often, this can be seen in table 1.

This means that higher associativity is worse for higher spatial locality of data.

Table 1 – Miss rates

Miss rates	test1	test2
cache1	0.0091	0.1577
cache2	0.0177	0.0235

4 Evaluation of cache configurations

Here we analyze how the cache memory performs when it only handles data, instructions and for both data and instructions.

4.1 Description

For the three test cases we tested with caches of size ranging from 128kB up to 4096kB. For each size in memory we also tested with associativity ranging from 1 to 8.

4.2 Solution

We found that when only storing data in the cache, the smallest memory with the lowest associativity performed worst, and for every step in increasing the memory size and the associativity the performance also increased, this can be seen in figure 1.

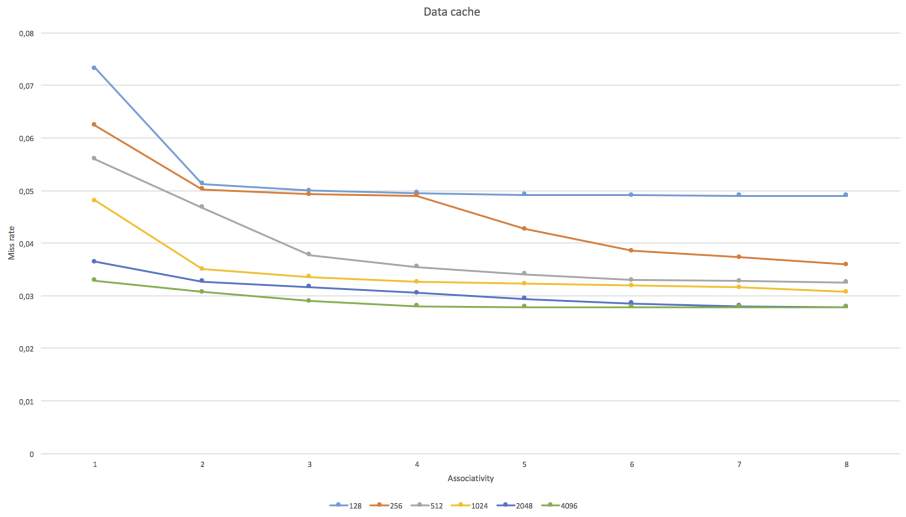

Figure 1 – Only data stored in cache.

When storing only instructions in the cache, the same behaviour as when storing only data can be observed, this can be seen in figure 2.

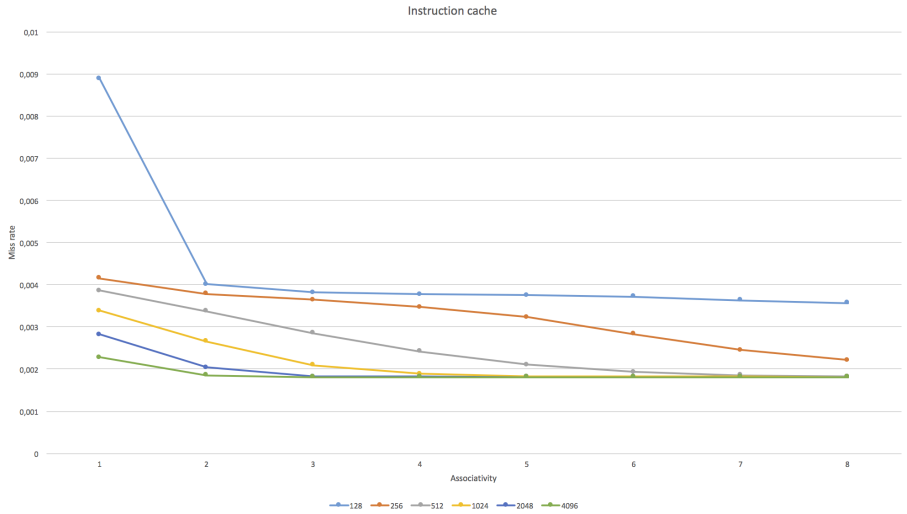

Figure 2 – Only instructions stored in cache.

And finally when storing both data and instructions the biggest memory was again at the top in performance, with the gaps in performance decreasing with increasing size and associativity as can be seen in figure 3.

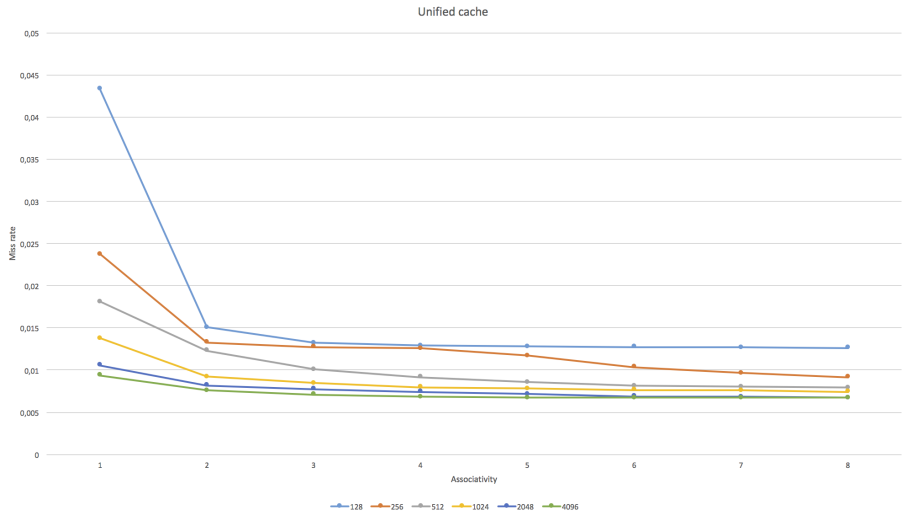

Figure 3 – Both data and instructions is stored in cache.

We also noticed that the gap in performance tightened when the sizes and associativity of the caches became greater.

5 Conclusion