TDTS10 Computer Architecture

Lab Assignment 1: Cache Memories

Axel Strid (axest556)
Dennis Zakrisson (denza625)

November 28, 2024

Contents

1 Objective

The purpose of this lab is to understand the functionality of cache memories and to gain insight into various trade-offs related to the design of systems with cache memories.

2 Preparation

Before performing this lab, we reviewed the following:

- Lab 0: Using the Toolset
- Lecture notes about cache memories and related exercises
- Resources describing caching policies and tools to visualize cache structures

3 Assignments

3.1 Cache Basics

3.1.1 Locality of Data

1. **Analysis of Cache Addressing:**

   (a) **Byte, Line, and Tag Bits:**

   $$\text{(Off-set) Byte number} = \log_2(\text{block size}) = \log_2(8) = 3\,\text{bits} = \text{bit 0-2}$$

   $$\text{(Set) Line number} = \log_2(\text{number of lines}) = \log_2(32) = 5\,\text{bits} = \text{bit 3-7}$$

   $$\text{Tag number} = 16 - (\text{byte bits} + \text{line bits}) = 16 - 8 = 8\,\text{bits} = \text{bit 8-15}$$

   (b) **Address to Line Number Conversion:**

   i. Address 1: Line number = `00011` = 3
   ii. Address 2: Line number = `00110` = 6
   iii. Address 3: Line number = `00011` = 3
   iv. Address 4: Line number = `10101` = 21

   (c) **Closest Addresses:**

   - `0001 1010 0001 1110` $_2 = 6\ _{10}$
   - `0001 1010 0001 1000` $_2 = 0\ _{10}$

   (d) **Total Cache Memory:**

   $$32\ \text{lines} \times 8\ \text{bytes per line} = 256\,\text{bytes}$$

   (e) **Purpose of Tag Bits:** To determine whether the cached data corresponds to the requested main memory address.

2. **f) Address Access Analysis:**

(a) For array element `a[2]`:

$$\text{Byte number} = 2 \times 4 = 8$$

Line and set calculations:

$$\text{Line number} = \text{Byte number/Bytes per line} = 8/8 = 1$$

$$\text{Set number} = (\text{Line number}) \text{ mod } (\text{Number of sets}) = 1 mod 32 = 1$$

Result: `a[2]` resides in set 1.

(b) For array element `a[64]`:

$$\text{Byte number} = 64 \times 4 = 256$$

Line and set calculations:

$$\text{Line number} = 256/8 = 32, \quad \text{Set number} = 32 mod 32 = 0$$

Result: `a[64]` resides in set 0.

3.2 Locality of Data

1. **Analysis of Nested Loops**

```
test1.c
    for (i = 0; i < TAB_SIZE; i++) {
        for (j = 0; j < 5; j++) {
            sum += A[i];
        }
    }

test2.c
    for (i = 0; i < TAB_SIZE - CACHE_SIZE; i++) {
        for (j = 0; j < 5; j++) {
            sum += A[i];
            sum+= A[i+CACHE_SIZE];
        }
    }
```

Spatial and temporal locality of the data: The for-loop iterating over i always access a new index next to it, and never itself again. This is spatial locality of the data. The for-loop iterating over j will always go from 0,1,2,3,4 and then repeat itself again, meaning it is temporal locality of data.

- Cache1: Block size = 16 bytes, associativity = 1
- Cache2: Block size = 8 bytes, associativity = 2

2. **Miss Ratio Analysis:**

- **test1 with Cache1:**

  Block size 16 + Associativity 1 = 4 spots in each Cache Line

  First access is empty because Cache Memory is empty

  Load the Cache Line with A[0], A[1], A[2], A[3]

  Hit each access until i = 4, then it repeats

  Total of 1 miss and 19 hits

  Miss ratio = 1/20 = 5%

- **test1 with Cache2:**

  Miss ratio = 2/20 = 10%

- **test2 with Cache1:**

  Miss ratio = 100% (always misses due to conflicts)

- **test2 with Cache2:**

  Miss ratio = 1/10 = 10%

Best combination: `test1 with Cache1 (5% miss ratio)`
Worst combination: `test2 with Cache1 (100% miss ratio)`

3. **Simulated Results:**

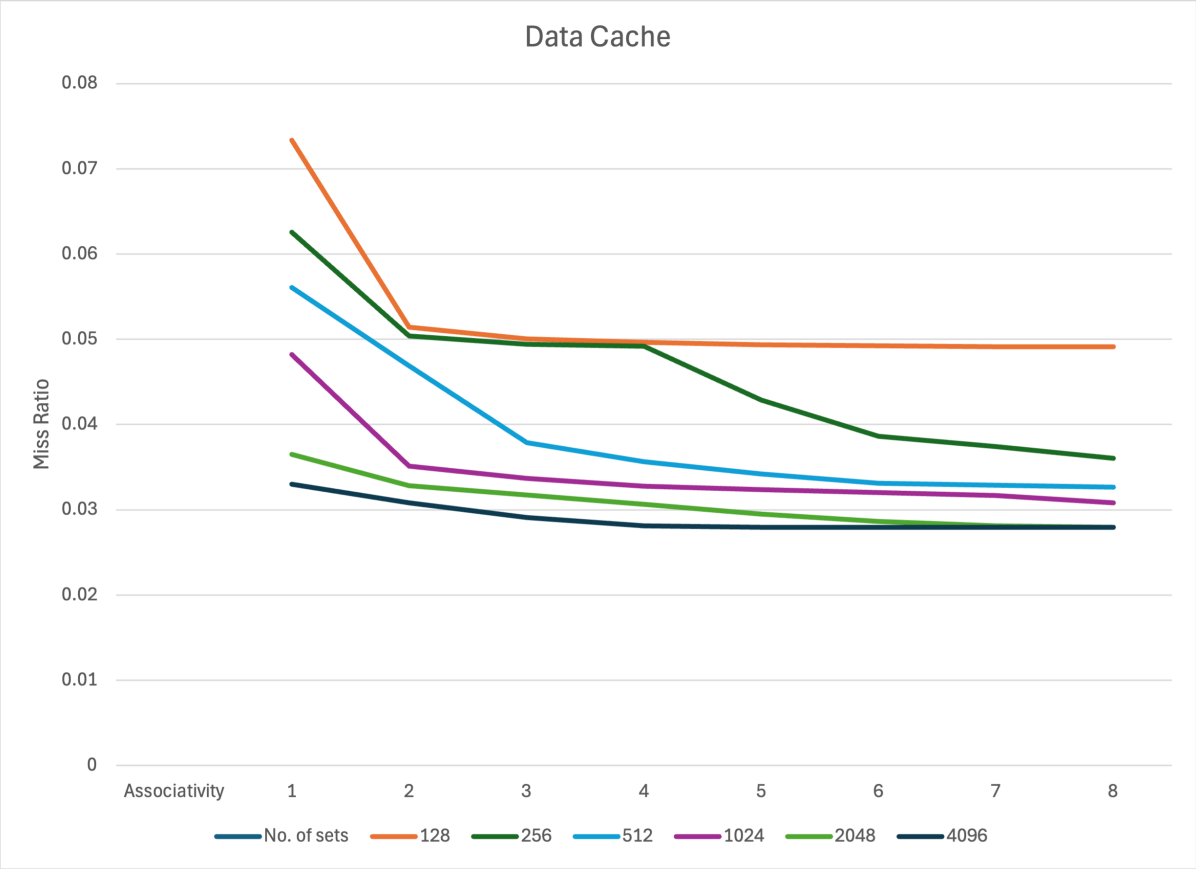
   For `test1` with `cache1`: miss ratio = 1%
   For `test1` with `cache2`: miss ratio = 1.8%
   For `test2` with `cache1`: miss ratio = 15%
   For `test2` with `cache2`: miss ratio = 2.4%

   As our calculations showed:

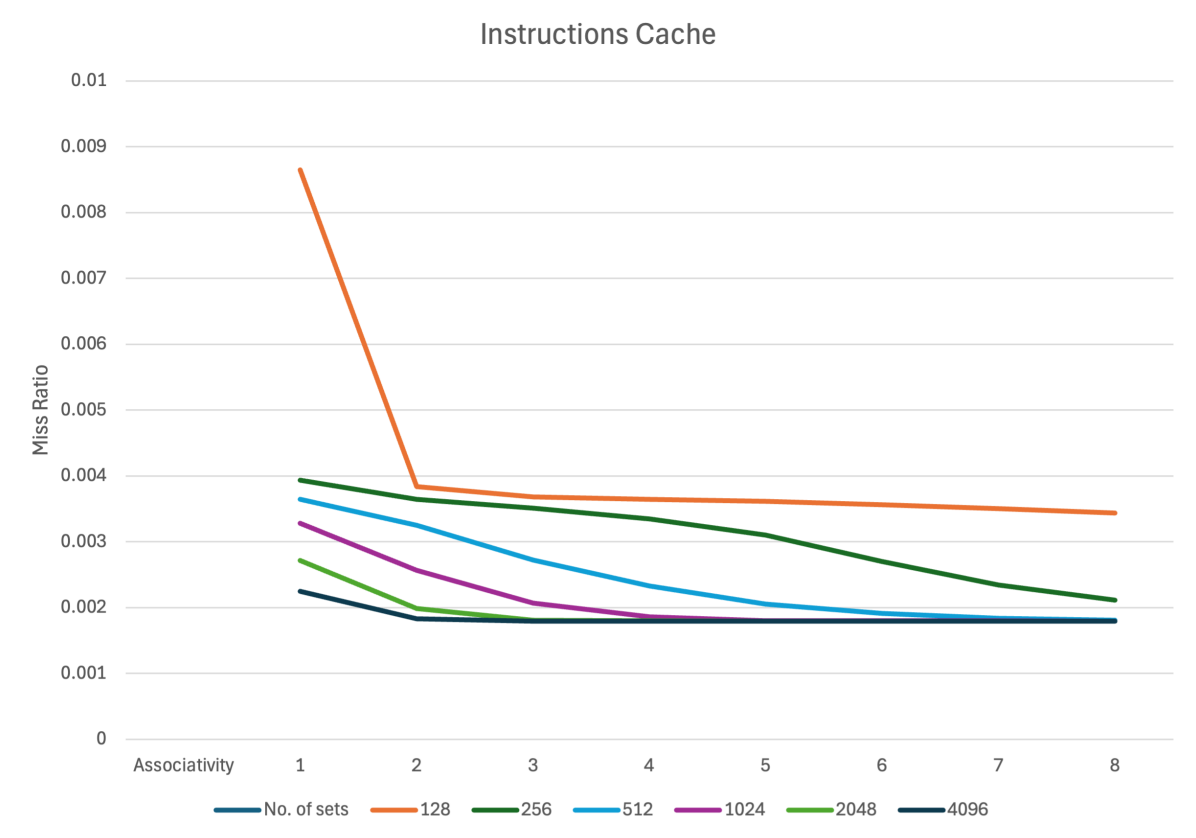
   - `test1` with `cache1` is the best.
   - `test2` with `cache1` is the worst.

   Conslusion: Higher associativity leads to a lower miss ratio.

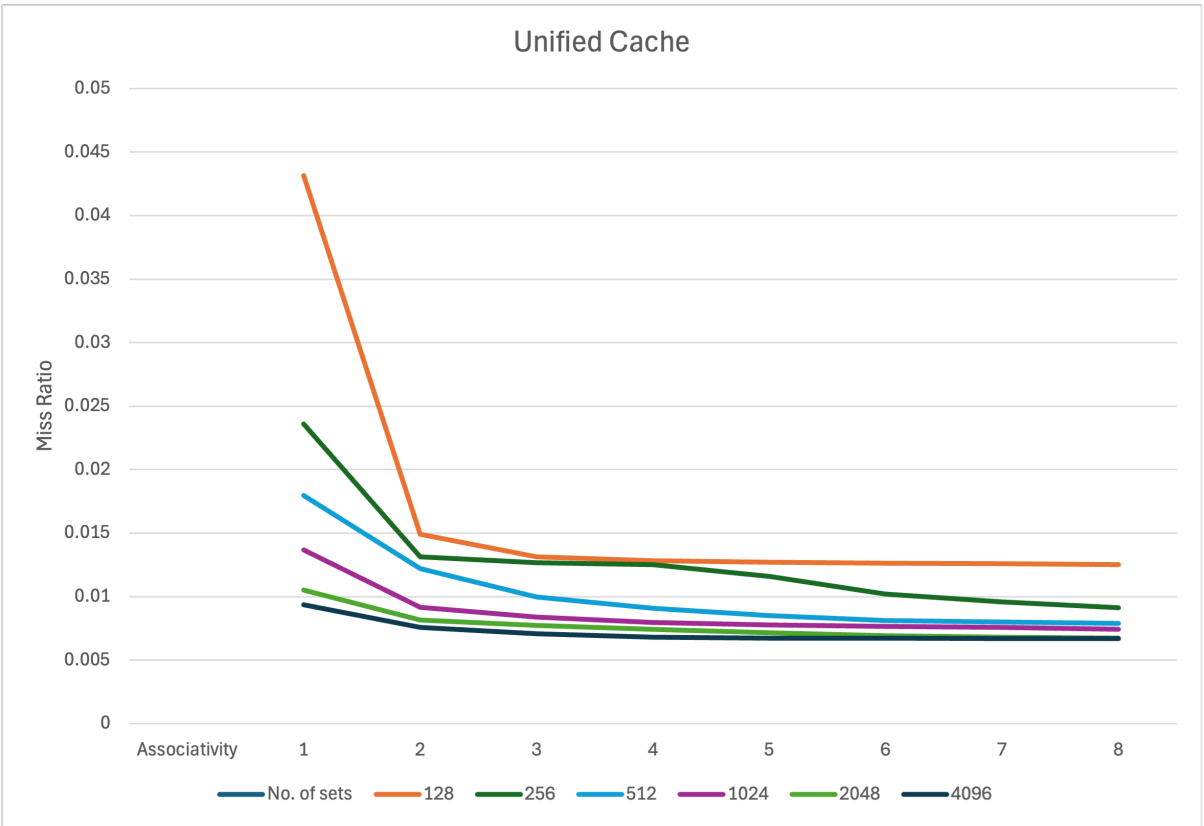
3.3 Evaluation of Cache Configurations



Miss ratio for the data cache based on the level of associativity and the number of sets.



Miss ratio for the instruction cache based on the level of associativity and the number of sets.



Miss ratio for the unified cache based on the level of associativity and the number of sets.

1. **Miss Ratio vs. Associativity**
   The graph shows that increasing associativity leads to a lower miss ratio. Similarly, increasing the number of sets also reduces the miss ratio.

2. **Cache System Design for go.ss**
   To determine the optimal cache system design for go.ss, calculations are based on a single representative data point from all three respective graphs. For this analysis, the configuration with 512 number of sets and an associativity level of 3 has been selected as the reference point.

   Miss Ratio:

   - Data Cache Miss Ratio: 0.0375
   - Instructions Cache Miss Ratio: 0.00275
   - Unified Cache Miss Ratio: 0.01

   Workload Distribution:

   - Instruction-heavy workload: 70% instructions, 30% data
   - Data-heavy workload: 30% instructions, 70% data
   - Balanced workload: 50% instructions, 50% data

Using the Total Miss Ratio function:

Total Miss Ratio = (Instruction Ratio × Instruction Miss Ratio) + (Data Ratio × Data Miss Ratio)

Instruction-heavy Workload:

- Data Cache: $(0.7 \times 1) + (0.3 \times 0.0375) = 0.71125$
- Instructions Cache: $(0.7 \times 0.00275) + (0.3 \times 1) = 0.301925$
- Unified Cache: $(0.7 \times 0.01) + (0.3 \times 0.01) = \mathbf{0.01}$

Data-heavy Workload:

- Data Cache: $(0.3 \times 1) + (0.7 \times 0.0375) = 0.32625$
- Instructions Cache: $(0.3 \times 0.00275) + (0.7 \times 1) = 0.701925$
- Unified Cache: $(0.3 \times 0.01) + (0.7 \times 0.01) = \mathbf{0.01}$

Balanced Workload:

- Data Cache: $(0.5 \times 1) + (0.5 \times 0.0375) = 0.51875$
- Instructions Cache: $(0.5 \times 0.00275) + (0.5 \times 1) = 0.501375$
- Unified Cache: $(0.5 \times 0.01) + (0.5 \times 0.01) = \mathbf{0.01}$

From the calculations above it is obvious that a separate instruction access and data access cache is the most optimal choice, since the unified cache has the lowest miss ratio for all workloads.

3. **Cost-Effective Cache Configuration**

Using the cost function:

$$\text{Cost Function} = (100000 \times \text{miss ratio}) + (0.01 \times \text{cache size in bytes})$$

Using the cache size function:

$$\text{Cache Size} = \text{Number of Sets} \times \text{Associativity} \times \text{Line size}$$

Note: The line size is 32 (found in the configuration file).

The optimal configuration balances the miss ratio and cache size to minimize costs within the 8000 SEK budget. Plugging the different combinations into excel gives us these results:

Instructions Cache

No. of sets	1	2	3	4	5	6	7	8
128	0.008648	0.003842	0.003686	0.003644	0.003618	0.003564	0.003505	0.003436
256	0.003939	0.003646	0.003509	0.003352	0.003104	0.002702	0.002346	0.002117
512	0.003646	0.003255	0.002726	0.002333	0.002055	0.001915	0.001836	0.001809
1024	0.003282	0.002568	0.002066	0.001862	0.001805	0.001802	0.0018	0.001799
2048	0.002714	0.001987	0.001811	0.001801	0.001799	0.001799	0.001799	0.001799
4096	0.002246	0.001832	0.001799	0.001799	0.001799	0.001799	0.001799	0.001799

Table representing miss ratio based on the level of associativity and the number of sets.

		cache line siz	32					
Cache size								
No. of sets	1	2	3	4	5	6	7	8
128	4096	8192	12288	16384	20480	24576	28672	32768
256	8192	16384	24576	32768	40960	49152	57344	65536
512	16384	32768	49152	65536	81920	98304	114688	131072
1024	32768	65536	98304	131072	163840	196608	229376	262144
2048	65536	131072	196608	262144	327680	393216	458752	524288
4096	131072	262144	393216	524288	655360	786432	917504	1048576

Table representing the cache size based on the number of sets and associativity levels.

Cost table								
No. of sets	1	2	3	4	5	6	7	8
128	905.76	466.12	491.48	528.24	566.6	602.16	637.22	671.28
256	475.82	528.44	596.66	662.88	720	761.72	808.04	867.06
512	528.44	653.18	764.12	888.66	1024.7	1174.54	1330.48	1491.62
1024	655.88	912.16	1189.64	1496.92	1818.9	2146.28	2473.76	2801.34
2048	926.76	1509.42	2147.18	2801.54	3456.7	4112.06	4767.42	5422.78
4096	1535.32	2804.64	4112.06	5422.78	6733.5	8044.22	9354.94	10665.66

Table calculating the cost values for various cache configurations, considering the miss ratio and cache size.

From the tables we see that we find a minimum price of 466.12 SEK with the combination of 128 number of sets and an associativity level of 2.

4 Conclusions

Through this lab, we analyzed the performance of different cache configurations and determined the trade-offs involved in cache design. The results highlighted the importance of associativity and block size in achieving an optimal miss ratio for various access patterns.