NC State University

Department of Electrical and Computer Engineering

ECE 463/563 (Prof. Rotenberg)

Project #1: Cache Design, Memory Hierarchy Design

REPORT TEMPLATE (Version 2.0)

by

Jordan Carr

NCSU Honor Pledge: "I have neither given nor received unauthorized aid on this project."

Student's electronic signature: Jordan Carr

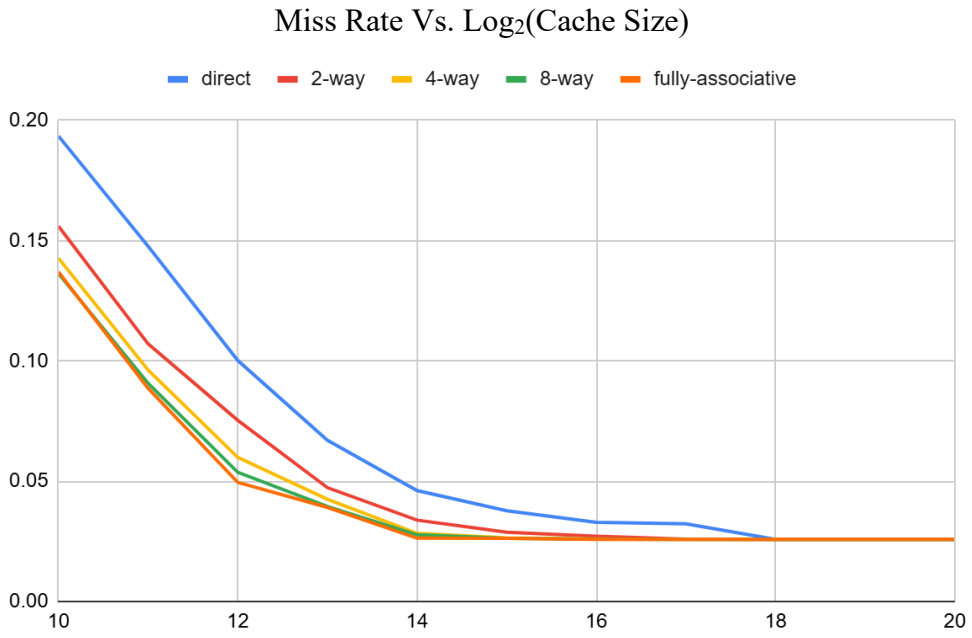
Course number: 563

1. L1 cache exploration: SIZE and ASSOC

GRAPH #1 (***total number of simulations: 55***)

For this experiment:

- Benchmark trace: gcc_trace.txt
- L1 cache: SIZE is varied, ASSOC is varied, BLOCKSIZE = 32.
- L2 cache: None.
- Prefetching: None.



Answer the following questions:

1. For a given associativity, how does increasing cache size affect miss rate?

Increasing Cache size for a given associativity decreases the miss rate.

2. For a given cache size, how does increasing associativity affect miss rate?

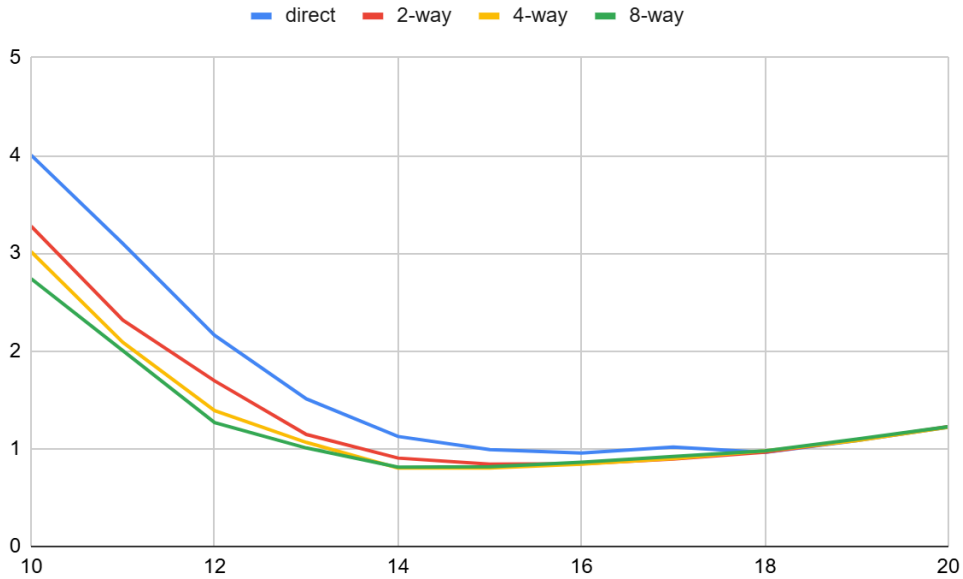
Increasing associativity for a given cache size decreases the miss rate

3. Estimate the *compulsory miss rate* from the graph and briefly explain how you arrived at this estimate.

compulsory miss rate = .0258 How I arrived at this estimate: All fully associative caches have the same miss rate, and fully associative caches eliminate conflict misses.

GRAPH #2 (***no additional simulations with respect to GRAPH #1***)

AAT Vs. Log2(Cache Size)

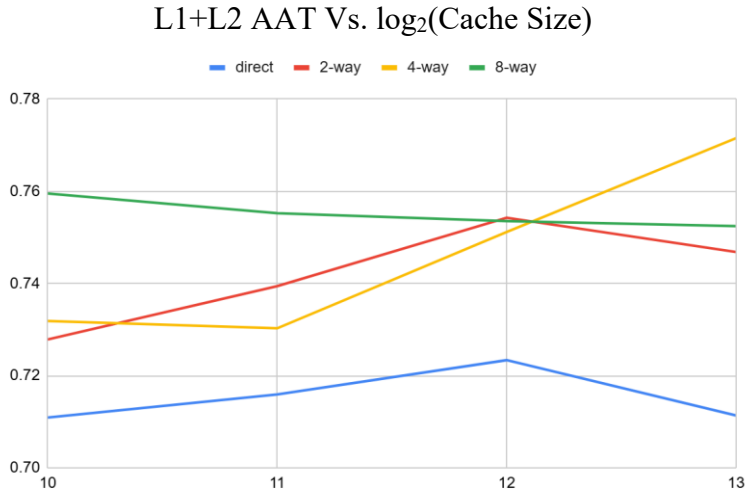


Answer the following question:

1. For a memory hierarchy with only an L1 cache and BLOCKSIZE = 32, which configuration yields the best (*i.e.*, lowest) AAT and what is that AAT?

Direct-mapped or set-associative cache configuration that yields the lowest AAT:
32KB, 4-way set associative
AAT for this configuration: .80189

GRAPH #3 (***total number of simulations: 16***)



Answer the following questions:

1. With the L2 cache added to the system, which L1 cache configuration yields the best (*i.e.*, lowest) AAT and what is that AAT?

L1 configuration that yields the lowest AAT with 16KB 8-way L2 added: 1KB, direct mapped AAT for this configuration: .7109234795

2. How does the lowest AAT with L2 cache (GRAPH #3) compare with the lowest AAT without L2 cache (GRAPH #2)?

The lowest AAT with L2 cache is less than the lowest AAT without L2 cache.

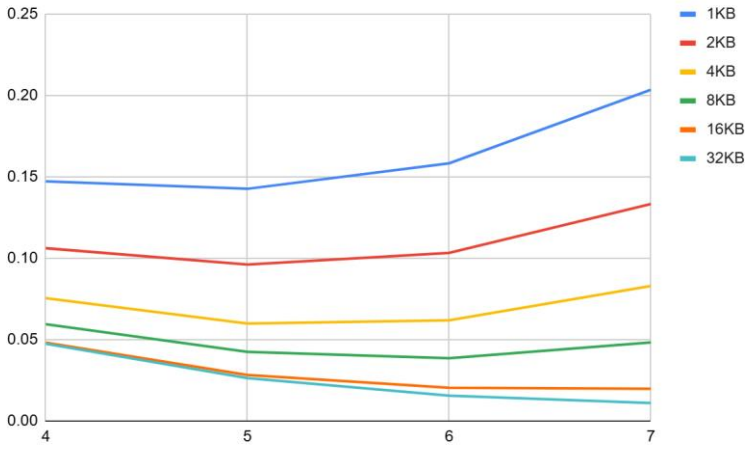
2. L1 cache exploration: SIZE and BLOCKSIZE

GRAPH #4 (***total number of simulations: 24***)

For this experiment:

- Benchmark trace: gcc_trace.txt
- L1 cache: SIZE is varied, BLOCKSIZE is varied, ASSOC = 4.
- L2 cache: None.
- Prefetching: None

L1 Miss Rate Vs. Log_2(BLOCKSIZE)



Answer the following questions:

1. Do smaller caches prefer smaller or larger block sizes?

Smaller caches prefer Smaller block sizes. For example, the smallest cache considered in Graph #4 (1KB) achieves its lowest rate with a block size of 32 B.

2. Do larger caches prefer smaller or larger block sizes?

Larger caches prefer Larger block sizes. For example, the largest cache considered in Graph #4 (32KB) achieves its lowest miss rate with a block size of 128 B.

3. As block size is increased from 16 to 128, is the tension between *exploiting more spatial locality* and *cache pollution* evident in the graph? Explain.

Yes, the tension between *exploiting more spatial locality* and *cache pollution* is evident in the graph.

For example, consider the smallest (1KB) cache in Graph #4. Increasing block size from 16 B to 32 B is helpful (reduces miss rate) due to exploiting more spatial locality. But then increasing block size further, from 32 B to 64 B, is not helpful (increases miss rate) due to cache pollution having greater effect.

3. L1 + L2 co-exploration

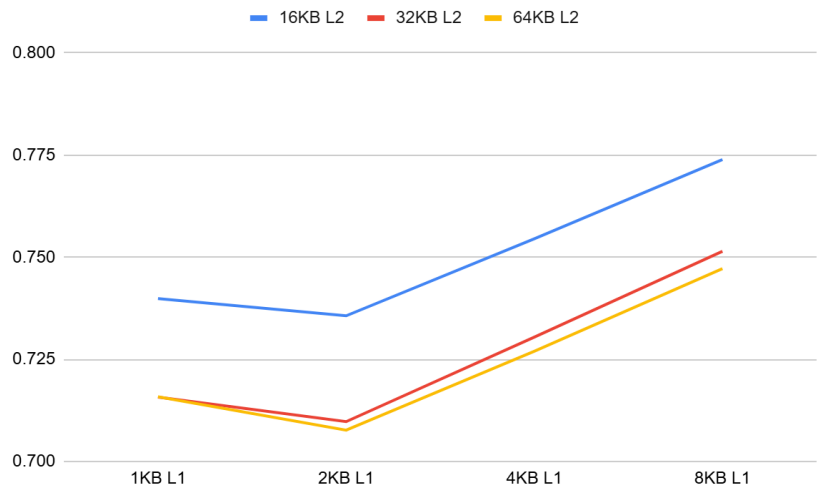
GRAPH #5 (***total number of simulations: 12***)

For this experiment:

- Benchmark trace: gcc_trace.txt
- L1 cache: SIZE is varied, BLOCKSIZE = 32, ASSOC = 4.
- L2 cache: SIZE is varied, BLOCKSIZE = 32, ASSOC = 8.
- Prefetching: None.

Plot AAT on the y-axis versus $\log_2$(L1 SIZE) on the x-axis, for four different L1 cache sizes: L1 SIZE = 1KB, 2KB, 4KB, 8KB. (That is, $\log_2$(L1 SIZE) = 10, 11, 12, 13.) The graph should contain three separate curves (*i.e.*, lines connecting points), one for each of the following L2 cache sizes: 16KB, 32KB, 64KB. All points for the 16KB L2 cache should be connected with a line, all points for the 32KB L2 cache should be connected with a line, *etc*.

L1 + L2 AAT Vs. L1 Cache Size



Answer the following question:

1. Which memory hierarchy configuration in Graph #5 yields the best (*i.e.*, lowest) AAT and what is that AAT?

Configuration that yields the lowest AAT: 2KB L1 with 64KB L2 Lowest AAT: 0.7076578326

4. Stream buffers study (ECE 563 students only)

TABLE #1 (***total number of simulations: 5***)

For this experiment:

- Microbenchmark: stream_trace.txt
- L1 cache: SIZE = 1KB, ASSOC = 1, BLOCKSIZE = 16.
- L2 cache: None.
- PREF_N (number of stream buffers): 0 (pref. disabled), 1, 2, 3, 4
- PREF_M (number of blocks in each stream buffer): 4

The trace "stream_trace.txt" was generated from the loads and stores in the loop of interest of the following microbenchmark:

```
#define SIZE 1000

uint32_t a[SIZE];
uint32_t b[SIZE];
uint32_t c[SIZE];

int main(int argc, char *argv[]) {
…
   // LOOP OF INTEREST
   for (int i = 0; i < SIZE; i++)
      c[i] = a[i] + b[i];   // per iteration: 2 loads (a[i], b[i]) and 1 store (c[i] = …)
…
}
```

Fill in the following table and answer the following questions:

PREF_N, PREF_M	L1 miss rate
0,0 (pref. disabled)	0.2500
1,4	0.2500
2,4	0.2500
3,4	0.0010
4,4	0.0010

1. For this streaming microbenchmark, with prefetching disabled, do L1 cache size and/or associativity affect the L1 miss rate (feel free to simulate L1 configurations besides the one used for the table)? Why or why not?

With prefetching disabled, L1 cache size and/or associativity does not affect L1 miss rate (for this streaming microbenchmark).

The reason: Due to the nature of the nature of the microbenchmark, data is only used once before it is evicted, so there little to no reuse of data.

2. For this streaming microbenchmark, what is the L1 miss rate with prefetching disabled? Why is it that value, *i.e.*, what is causing it to be that value? Hint: each element of arrays a, b, and c, is 4 bytes (uint32_t).

The L1 miss rate with prefetching disabled is .2500, because 4 bytes is $1/4^{th}$ the block size. In other words, only one in every 4 accesses will miss because the 16 Byte cache size will cover the next 3 elements in array when it is stored in cache.

3. For this streaming microbenchmark, with prefetching disabled, what would the L1 miss rate be if you doubled the block size from 16B to 32B? (hypothesize what it will be and then check your hypothesis with a simulation)

The L1 miss rate with prefetching disabled and a block size of 32B is .1260 because when cache stores a new block it will cover 8 of the 4Byte elements, meaning it will miss every 8 elements.

4. With prefetching enabled, what is the minimum number of stream buffers required to have any effect on L1 miss rate? What is the effect on L1 miss rate when this many stream buffers are used: specifically, is it a modest effect or huge effect? Why are this many stream buffers required? Why is using fewer stream buffers futile? Why is using more stream buffers wasteful?

Minimum number of stream buffers needed to have any effect on L1 miss rate: 3

With this many stream buffers, the effect on L1 miss rate is huge Specifically, the L1 miss rate is nearly 0. We only miss on the first elements of each array (hence a total of 3 misses).

This many stream buffers are required because we need a buffer for each array.

Using fewer stream buffers is futile because one or more arrays cannot be prefetched, resulting in misses in L1

Using more stream buffers is wasteful because 3 is enough to cover all arrays, any extra just becomes taken real estate