NC State University

Department of Electrical and Computer Engineering

ECE 463/563: Fall 2021 (Rotenberg)

Project #1: Cache Design, Memory Hierarchy Design

by

SRI HARSHA TEJ PATNALA

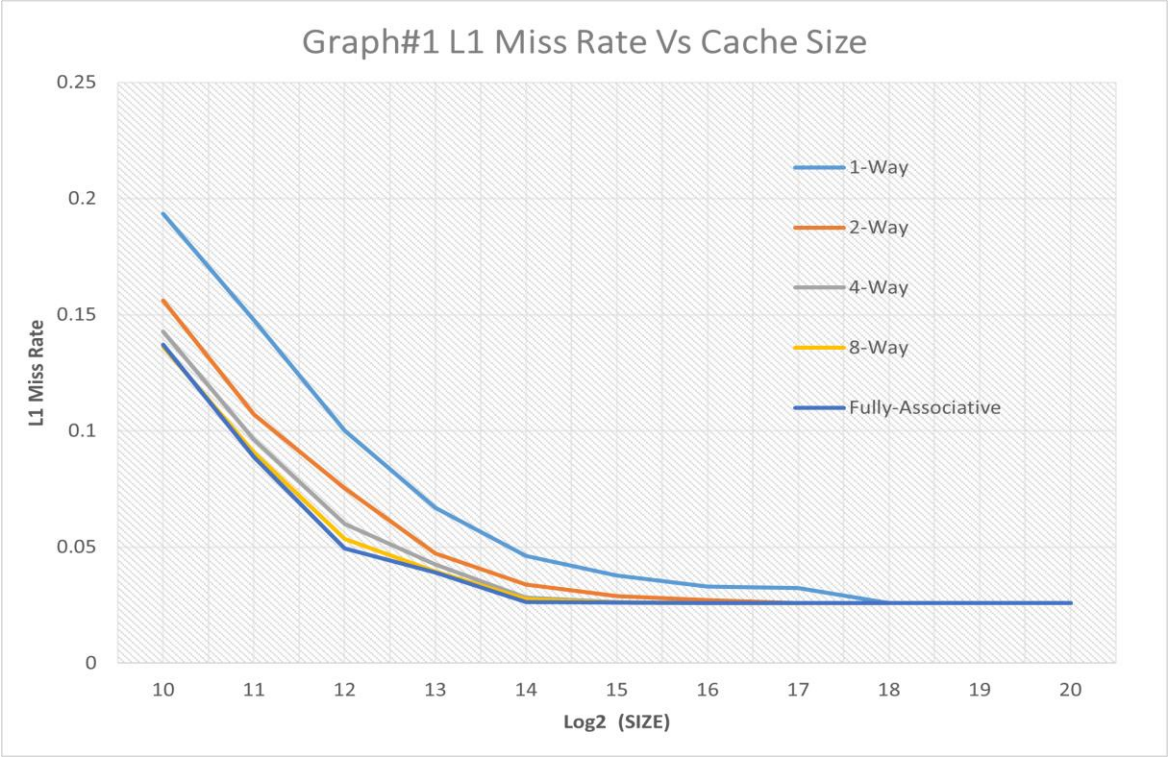
NCSU Honor Pledge: "I have neither given nor received unauthorized aid on this project."

Student's electronic signature: _____Sri Harsha Patnala_____
(Sign by typing your name)

Course number: _563_____
(463 or 563?)

L1 Cache Exploration: SIZE and ASSOC

Graph#1



1. From the simulation results and graph above, we can see that for a given associativity, increasing the cache size resulted in decreasing miss rate. This is expected as by increasing the cache size, the capacity misses could be greatly reduced.

   Further, increasing the associativity has shown a significant drop in the miss rate initially, but is not much for associativity greater than 4. This shows that increasing the associativity reduces the number of conflict misses in the cache.

   Although, increasing the associativity along with cache size has proved to be beneficial initially, it can be seen from the graph that regardless of increasing these two factors, the miss rate isn't reduced further and saturates for a larger caches and higher associativity.

2. Looking at the graph, we could see that the miss rate saturates despite increasing the cache size and associativity. This could mean the compulsory miss rate of the trace.

   From the graph, the compulsory miss rate is around 0.025.
3. To estimate the conflict miss rate for a given associativity, we need to consider a fully associative cache with a fairly large size and look at the miss rate of that to estimate the conflict miss rate. This is because, we know that for a fully associative cache, there are

enough ways for a block to go, which kind of eliminates conflict misses and the misses could be seen as compulsory + capacity misses.

For this discussion, I've assumed that a cache size of 32KB is large enough to eliminate capacity misses based on the graph and simulation results.

For this size, the miss rate is 0.0262

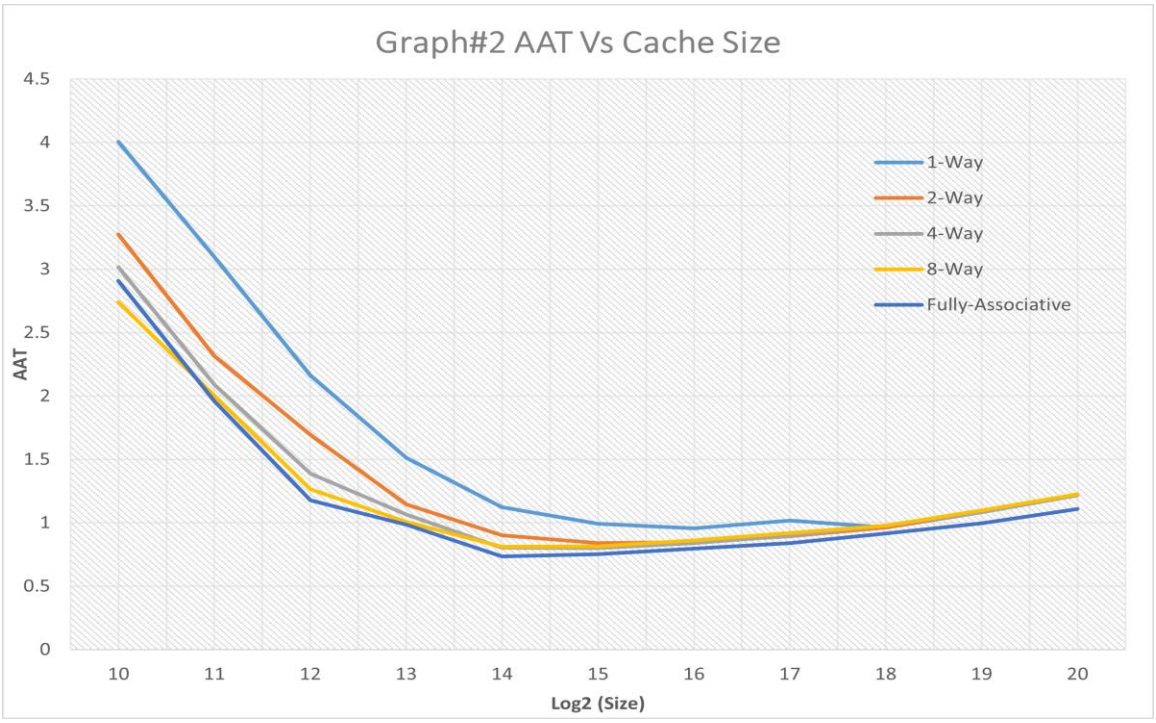
1-Way Associative (Direct-Mapped)
Miss rate = 0.0377
So, Approx. Conflict miss rate = 0.0377 – 0.0262 = 0.0115

2-Way Associative
Miss rate = 0.0288
Approx. conflict miss rate = 0.0288 – 0.0262 = 0.0026

4-Way Associative
Miss rate = 0.0264
Approx. conflict miss rate = 0.0264 – 0.0262 = 0.0002

8-Way Associative
Miss rate = 0.0262
Approx. conflict miss rate = 0

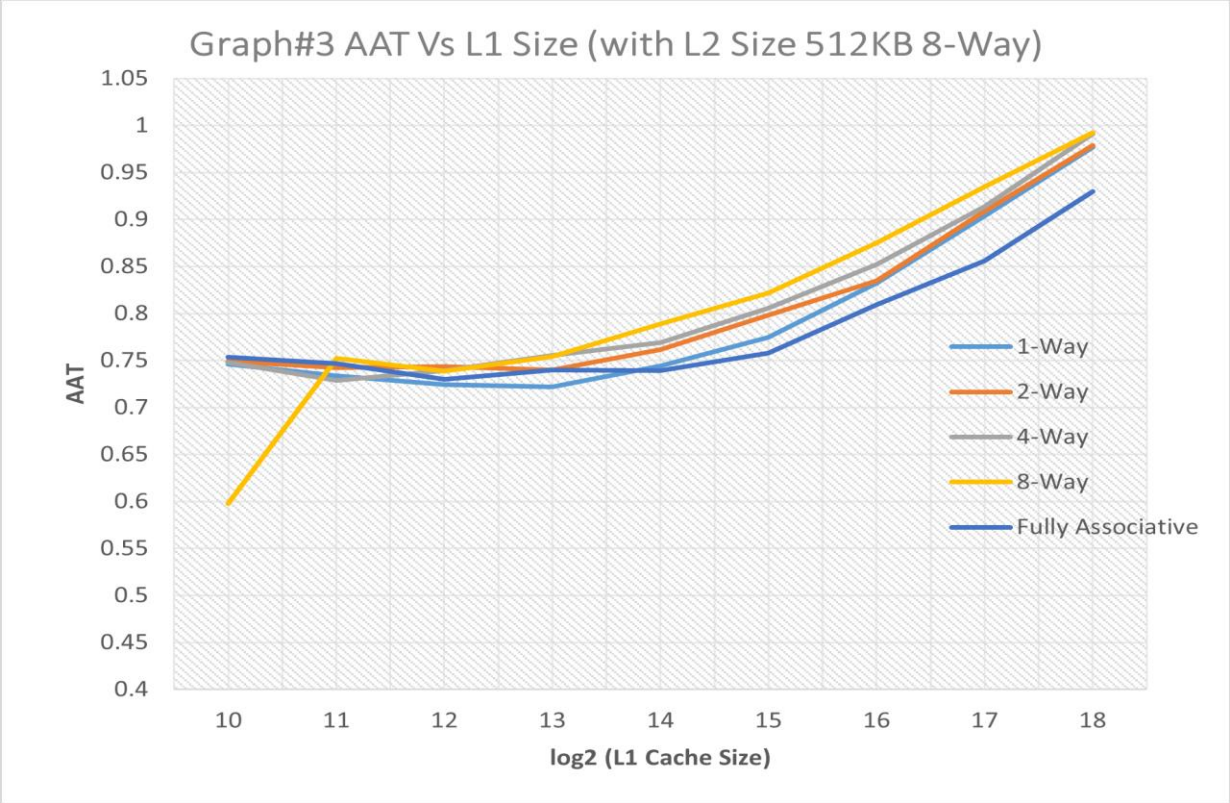
Graph#2



1. From the graph above, we can see that for an L1 cache size of 16KB, the values of AAT are quite minimal.
   So, for a cache configuration of 16KB size, 32B block size and 8-way or fully associative yields the best AAT of 0.734238 ns.

   Also, we could observe that by further increasing the size, the AAT increases. This is because as the size increases, the number of sets in the cache increase, which means that it takes time to access a block in cache, thereby increasing the hit time.

Graph#3



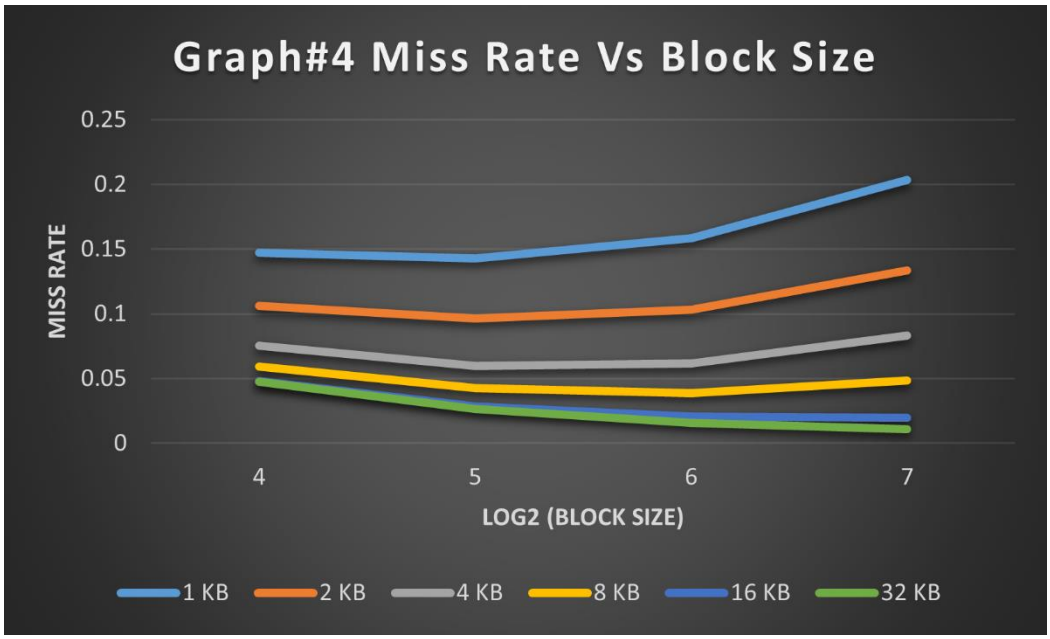
1. From the graph above, the cache with configuration 4 KB and 4-way associativity results in AAT quite close to the optimal AAT of graph#2.

2. Looking at the graph, we can see that for a L1 cache size of 8KB and 1-way associativity along with L2 size of 512KB and 8-way associativity, yields the optimal (best) AAT of 0.721586 ns.

   The optimal AAT for this graph is 1.26% lower than the AAT for graph#2.

L1 Cache Exploration: SIZE and BLOCKSIZE

Graph#4



1. Based on the graph above, as the block size increases, for smaller caches the miss rate is increases. But for larger caches, increasing the block size has diminished returns in terms of miss rate. As the cache size increases the graph tends to favor a larger block size.

   So clearly, smaller caches prefer smaller block sizes. Further as the block size increases the mis rate deteriorates, as cache pollution becomes a concern.
   However, for larger caches, they tend to prefer larger block sizes.

   This is because for smaller caches, as we go on increasing the block size, the number of blocks that could fit in the cache decreases. So, this could increase the capacity misses and thereby degrading the miss rate.

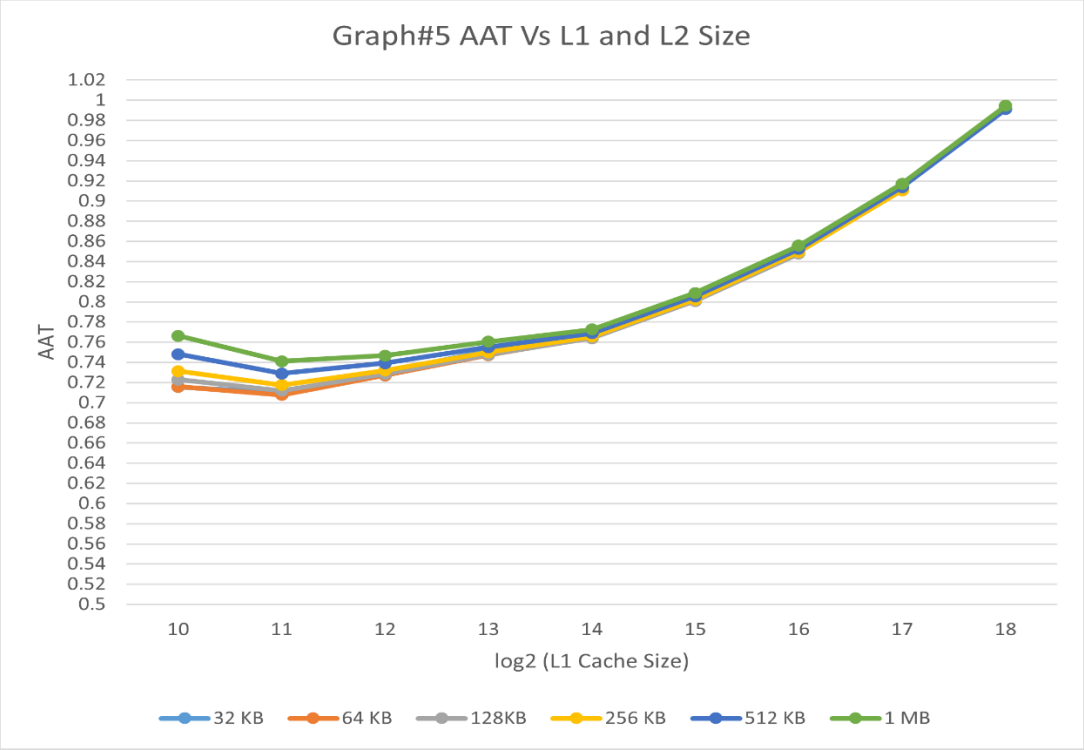
   For larger caches, increasing the block size has reduced the miss rate. This is because even though we are increasing the block size, the cache is still large enough to have a greater number of blocks than a smaller cache. So, it could decrease the capacity misses and thereby improve the miss rate.

   For smaller caches, there's a tradeoff between exploiting spatial locality and cache pollution. Smaller caches allow for lesser number of blocks. So, referencing consecutive bytes could prove effective if we increase block size. But if the block size is very large then for references where the bytes aren't sequential degrades the performance as cache pollution comes into picture.

For larger caches however, this trend could slightly change as larger size allows for greater number of blocks. So, even if the block size is large enough, we know that there are enough different blocks that could fit in the cache. So, whether the bytes are referenced consecutively or not, there are sufficient blocks and a fair block size to allocate in the cache.

L1 + L2 co-exploration

Graph#5



1. Based on the simulation results and the graph above, the following memory hierarchy yields the best (optimal) AAT:

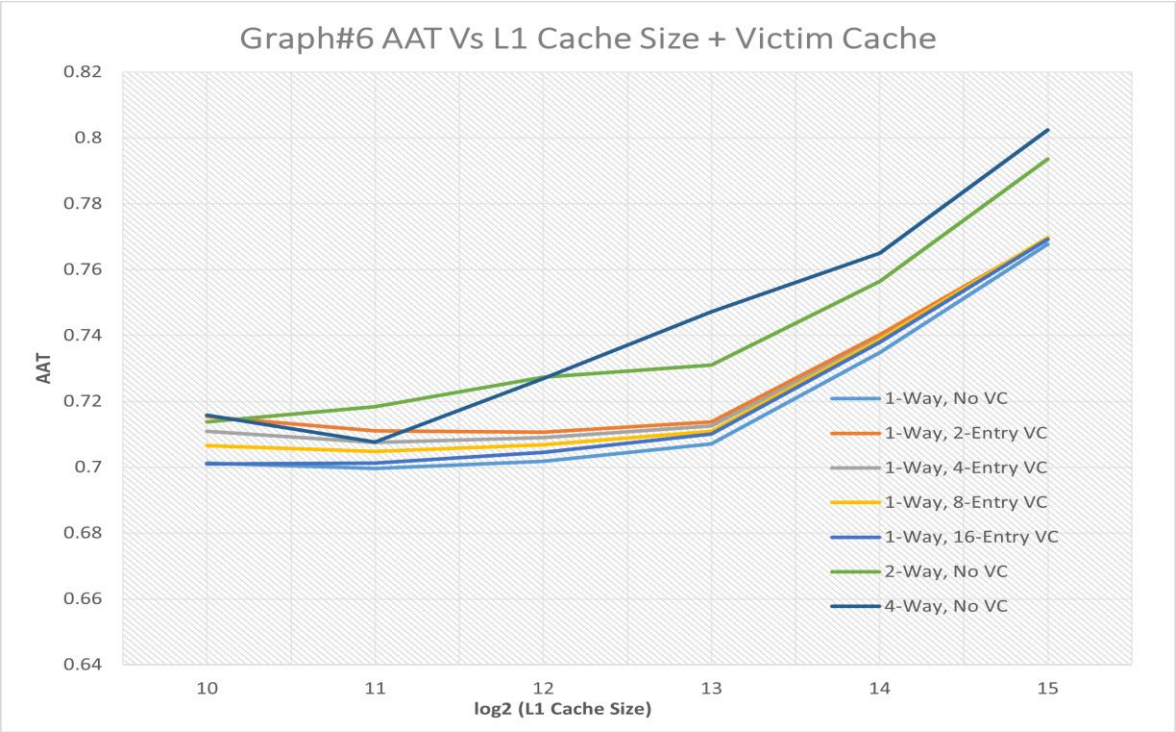
   L1 SIZE = 2 KB, ASSOC=4, BLOCK SIZE = 32B
   L2 SIZE = 64KB, ASSOC=8, BLOCK SIZE = 32B

2. The optimal AAT is 0.70765 ns. The memory hierarchy configuration which has the least area with AAT within 5% of the optimal AAT is:

   L1 SIZE = 1KB and L2 SIZE = 32KB

Victim Cache Study

Graph#6



1. For direct-mapped cache, it is evident that adding a victim cache has reduced the AAT. Initially as the number of victim blocks are increased for the same cache size, the AAT is almost the same. But for larger cache size, increasing the victim blocks led to increase in AAT as for larger caches the hit time increases. Eventually the values saturate for a large enough cache size.

   From the graph, having the additional victim cache has shown better performance than a cache with 2-way associativity, in terms of AAT.
   Expect for a cache size of 1KB with 2 victim block entries, the overall performance is better with victim blocks than a 2-way associative cache. As the number of victim blocks are increased, the performance is further improved compared to a 2-way associative cache of the same size.

2. From the graph, the following memory hierarchy configuration yields the best AAT:

   L1 SIZE= 2KB, L1 ASSOC=1, L2 SIZE = 64KB, Number of Victim Blocks = 0

3. From the graph, the following memory hierarchy configuration has the smallest area with AAT around 5% of the best AAT

   L1 SIZE = 1KB, L1 ASSOC=2, L2 SIZE=64KB, Number of Victim Blocks=0.