The State University of New York at Binghamton

Department of Computer Science

CS 520 – Spring 2019

Project #2: Cache Design, Memory Hierarchy Design

By

DEVINA SACHIN DHURI

Honor Pledge: I have neither given nor received unauthorized aid on this test or assignment.

Student's electronic signature: _Devina Sachin Dhuri______

8.1] L1 cache exploration: SIZE and ASSOC

GRAPH #1: Miss Rate (L1) vs Cache Size

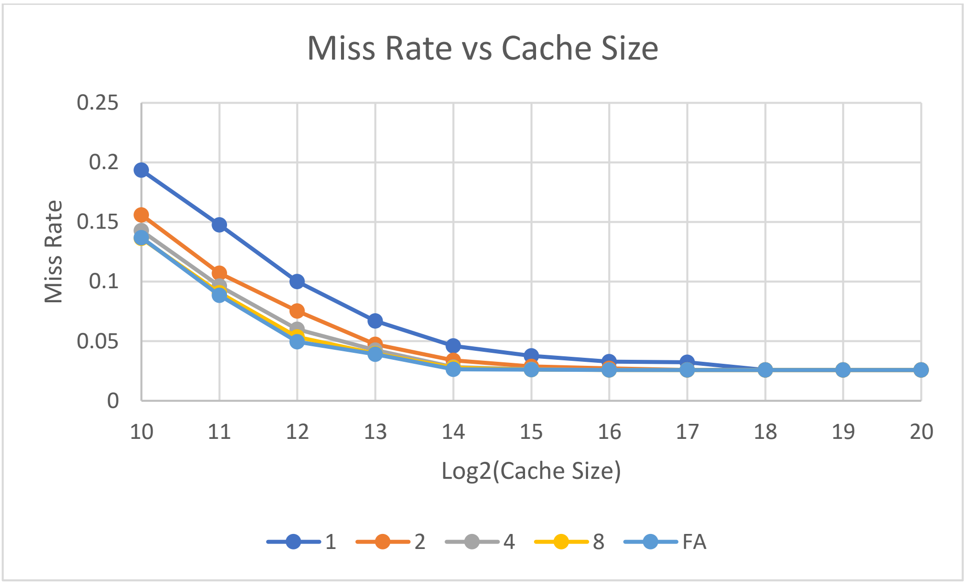
L1 cache: SIZE is varied, ASSOC is varied, BLOCKSIZE = 32.

L2 cache: None.

Replacement policy: LRU

Inclusion property: non-inclusive

Trace: GCC trace



1] As seen in the graph, for any given associativity, increase in the cache size reduces the miss rate *exponentially*. This happens because when the cache size increases a greater number of blocks can be accommodated in a greater number of sets. Thus, reduces the number of misses. But after a point (point 17 onwards from the graph), we face *diminishing returns* as more size doesn't always mean more performance. The larger the size, more is the distance of the data store and thus, it takes more time to drive and latch contents of a block. Thus, larger caches are slower to access and increase power consumption.

Also, given a cache size, if we increase the associativity, the miss rate generally reduces. However, increasing the associativity doesn't have much effect on the miss rate after a point of time. Fully Associative performance is better than direct mapped, but, higher associativity means you need to search more blocks thus is slower and also increases power consumption.

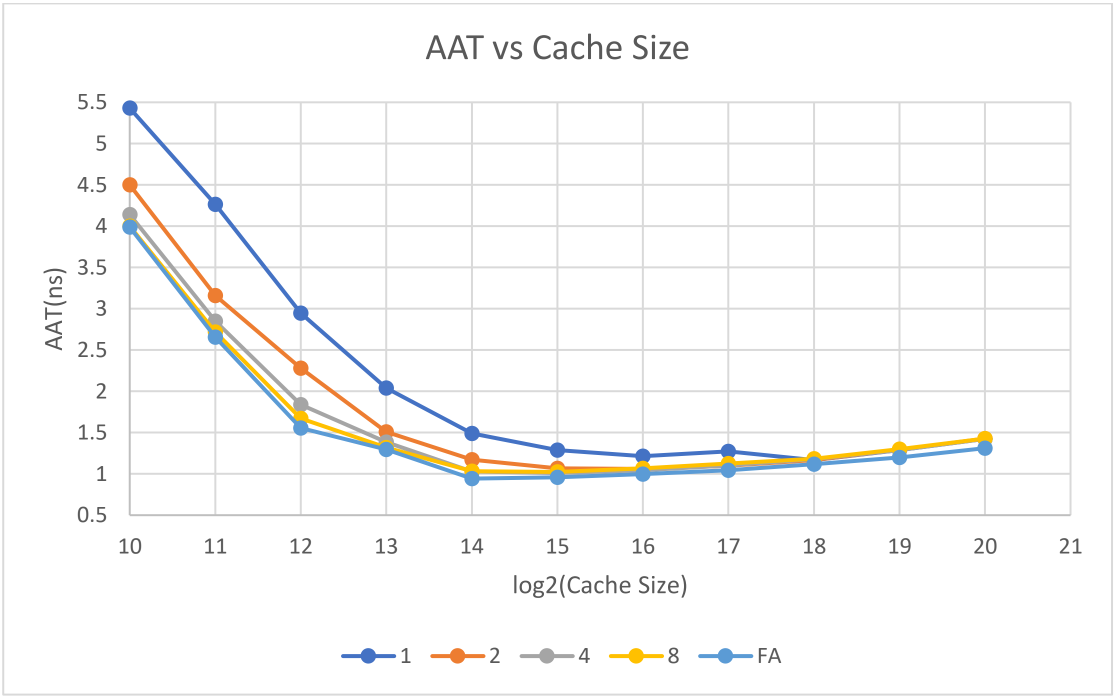
2] Compulsory misses can be estimated by looking at the fully associative cache at 1MB. Thus the compulsory miss rate is 0.0258

3] Conflict Miss rate can be estimated by subtracting the miss rate of fully associative cache from the miss rate of a given associativity for the same cache size.

Associativity:

	10	11	12	13	14	15	16	17	18	19	20
1	0.0565	0.0591	0.0506	0.02788	0.01975	0.01144	0.00709	0.00651	2E-05	2E-05	2E-05
2	0.0190	0.0185	0.0257	0.0082	0.0075	0.0025	0.0013	8E-05	2E-05	0	0
4	0.0057	0.0076	0.0103	0.0033	0.0019	0.00016	0.00012	0	0	0	0
8	0.000003	0.00209	0.00411	0.00042	0.0014	1E-05	6E-05	0	0	0	0

GRAPH #2: AAT (L1) vs Cache Size



As shown in the graph, for L1 cache with block size of 32 bytes, the lowest AAT value is obtained by a *16KB fully associative cache* (FA POINT 14).

8.2] Replacement Policy Study:

Graph #3: AAT vs log2(SIZE):

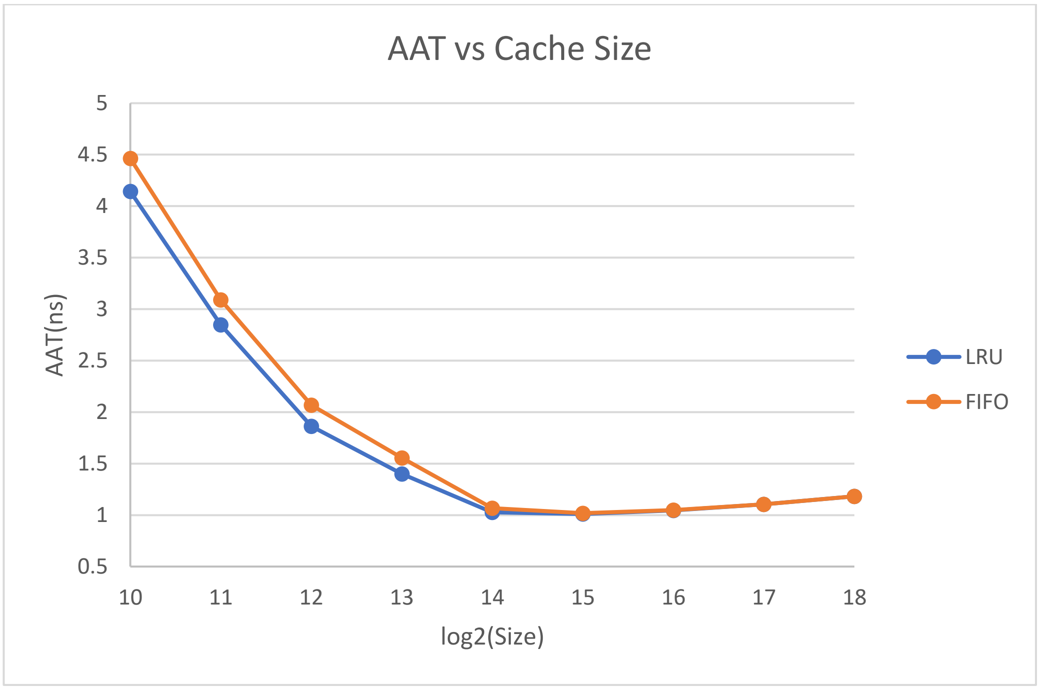
L1 cache: SIZE is varied, ASSOC = 4, BLOCKSIZE = 32.

L2 cache: None.

Replacement policy: varied

Inclusion property: non-inclusive

Trace: GCC trace



As the L2 Cache Size increases, the AAT value decreases up to a point. After a point it starts increasing again.

For the given specifications, as shown in the graph, LRU policy with L1 cache of size 16KB has the lowest AAT.

8.3] Inclusion Policy Study:

<u>Graph #4: AAT vs log2(SIZE):</u>

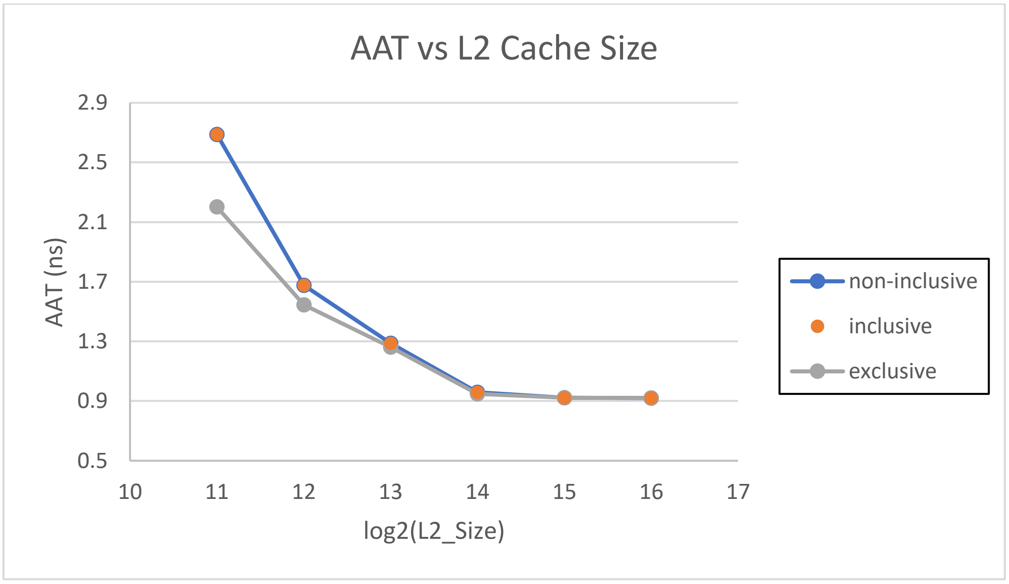
L1 cache: SIZE = 1KB, ASSOC = 4, BLOCKSIZE = 32.

L2 cache: SIZE = 2KB – 64KB, ASSOC = 8, BLOCKSIZE = 32.

Replacement policy: LRU

Inclusion property: varied

TRACE: GCC trace



As shown in the graph, the AAT decreases as the L2 cache size increases. This is because as the cache size decreases, the miss rate also decreases thus decreasing the AAT value.

The lowest AAT is received by Exclusive cache.