Georgia Institute of Technology

Advanced Computer Architecture [ECE 6100], Fall 2014

Project-I Cache Simulator

Project Report

By:

Harshit Jain

GT #ID: 903024992

Email: harshit@gatech.edu

Introduction:

Based upon the given requirements of a cache simulator I have developed the simulator encapsulating all desired requirements. I have chosen C++ for modeling the cache and all desired entities. The chosen platform for coding was eclipse on a windows platform, I have also used Github for version control.

Algorithm & Data Structures:

I have modelled the main cache using a different approach than **time-stamping**. My model uses a queue of doubly linked list as a set of blocks, and an array of such sets as my whole cache (I wanted to create my own hash). The sets are indexed with the index bits making the complexity as O(1) and the block on the begin of the queue is the MRU block in a set and at bottom is the LRU.

Victim cache is modeled similarly as main cache, it's just a special case of fully associative cache with number of sets as 1. The top of the FIFO is **begin** of the queue and the bottom is **end** of queue.

Prefetcher is implemented as per the desired requirements listed in the project proposal manual.

Initial Results:

My results are a 100% match to the statistics provided on Tsquare. Please check mylog, in the attached folder, or perform your own run to recheck results.

Simulations and Analysis:

The simulation analysis is done using a python script which runs multiple iterations altering C,B,S,V & K values and monitoring AAT for traces accordingly. The Analysis of all the traces is given below:

1. Astar.trace

Initial results:

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	5	3	4	2	16.853
2	10	5	0	4	2	34.13
3	15	5	3	0	0	22.52
4	15	5	3	0	2	16.855
5	15	5	3	4	0	22.52

Given that maximum cache size is of 48Kb. Hence max value of C:

C = $\log_2(48*1024)$

= Abs(15.54)

= 15

With C as a value equal to 15, we have a 32KB of cache. Rest 16KB will be used by tag store and victim cache and victim tags.

Max value of Tag store considering for a fully associative cache with B=1 and no victim cache: $(64-15+14+2)*2^{15-1}$

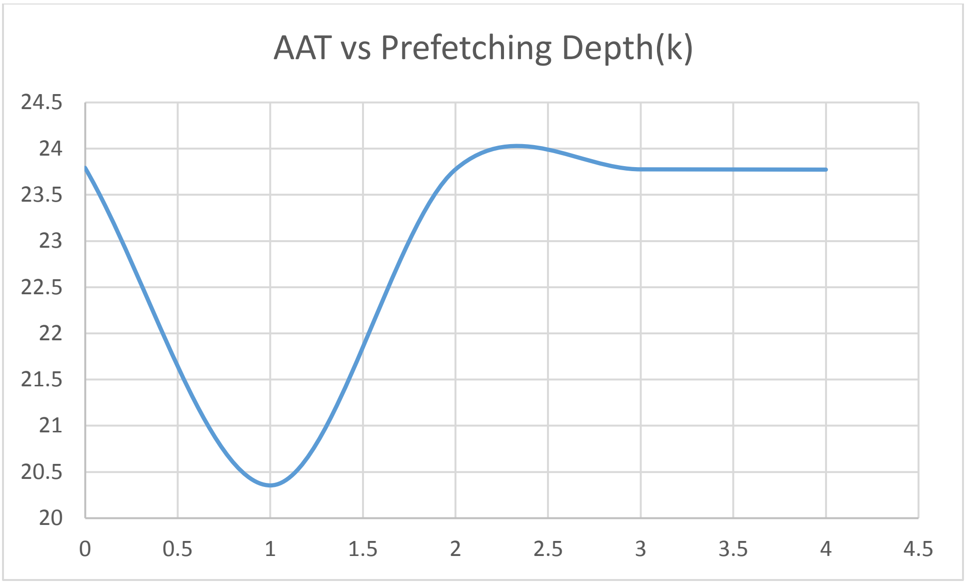
Hence we have a tag store of: $65*2^{14}$ bits or $65*2^{11}$ bytes or 130KB of tag storage. Which far exceeds our storage of 16KB. Thus we need to choose a value of B so that our Tag store is within 16KB boundary. Which is only 5(Tag store~8KB) and 6(~4KB).

Thus we have limited B only to values 5 and 6. Now let's consider a case for fully associative cache with block size of 5 and 6.

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	5	10	0	0	23.7919
2	15	6	9	0	0	14.413
3	15	5	3	4	2	16.853
4	15	5	3	0	2	16.855

Clearly evident from above that for trace astar.trace, a fully associative cache with B=5 and S=10 is having a lot of sequential accesses, as a cache with larger block size is showing better results. Even prefetching for case 1 will not yield better results than case 2, B=5 as it will cause more evictions and more pollutions, thus nullyfing its prefetching effect.

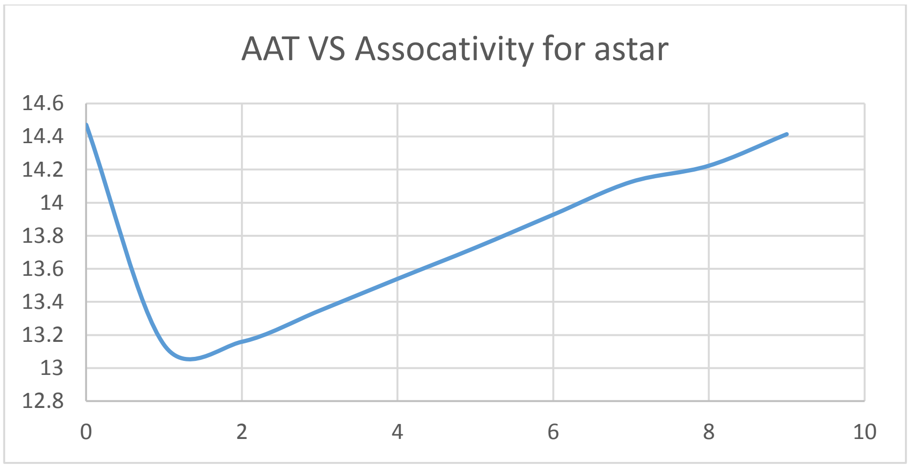
S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	5	10	0	0	23.7919
2	15	5	10	0	1	20.3528
3	15	5	10	0	2	23.7751
4	15	5	10	0	3	23.7755
5	15	5	10	0	4	23.7727



We see that at K=1 we have the prefetching knee.

It is clear from above that for astar we need to vary associativity at block size = 6.

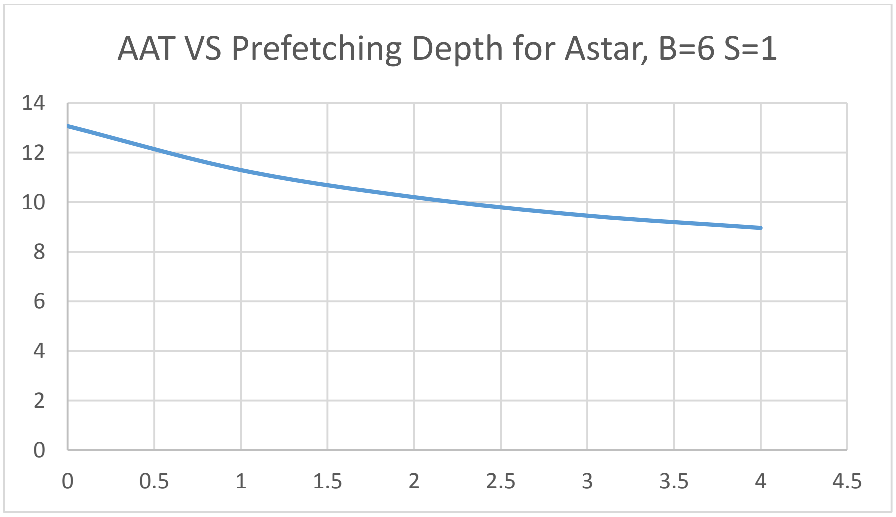
S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	6	9	0	0	14.4136
2	15	6	8	0	0	14.2232
3	15	6	7	0	0	14.1253
4	15	6	6	0	0	13.9273
5	15	6	5	0	0	13.7277
6	15	6	4	0	0	13.5397
7	15	6	3	0	0	13.3476
8	15	6	2	0	0	13.1592
9	15	6	1	0	0	13.1403
7	15	6	0	0	0	14.4706



Clearly the choice of associativity for astar is s=1, hence we have our c,b and s values as 15,6 & 1

Taking a case for v=4, and varying values of k we will get minimum AAT.

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	6	1	4	0	13.0685
2	15	6	1	4	1	11.2965
3	15	6	1	4	2	10.2045
4	15	6	1	4	3	9.45949
5	15	6	1	4	4	8.96574



Solution1) For astart.trace Cache values are:

C =15

B= 6

S= 1

V= 4

K= 4

AAT(ns) = 8.96ns

2. Bzip2.trace:

Let’s analyze initial results for bzip.trace:

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	5	3	4	2	3.022395
2	10	5	0	4	2	7.657889
3	15	5	3	0	0	3.212656
4	15	5	3	0	2	3.022395
5	15	5	3	4	0	3.212656

Similarly as in case of astar I analze case for fully associative cache for (B,S) equal to(6,9) and (5,10).

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	5	10	0	0	4.61266
2	15	6	9	0	0	4.1089

Taking (B,S) as (6,9) I decrease the associativity

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	6	9	0	0	4.1089
2	15	6	8	0	0	3.9089
3	15	6	7	0	0	3.7089
4	15	6	6	0	0	3.5089
5	15	6	5	0	0	3.3089

6	15	6	4	0	0	3.1089
7	15	6	3	0	0	2.9089
8	15	6	2	0	0	2.71331
9	15	6	1	0	0	2.61542
7	15	6	0	0	0	5.57897

Now taking prefetching into account:

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	6	1	4	0	2.56399
2	15	6	1	4	1	2.51221
3	15	6	1	4	2	2.47952
4	15	6	1	4	3	2.45821
5	15	6	1	4	4	2.43213

Solution2) For bzip2 the cache parameters are:

C= 15

B =6

S =1

V =4

K =4

AAT = 2.432ns

3. mcf.trace:

Let’s analyze initial results for mcf.trace:

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	5	3	4	2	6.248611
2	10	5	0	4	2	34.908410
3	15	5	3	0	0	6.249005
4	15	5	3	0	2	6.248611
5	15	5	3	4	0	6.249005

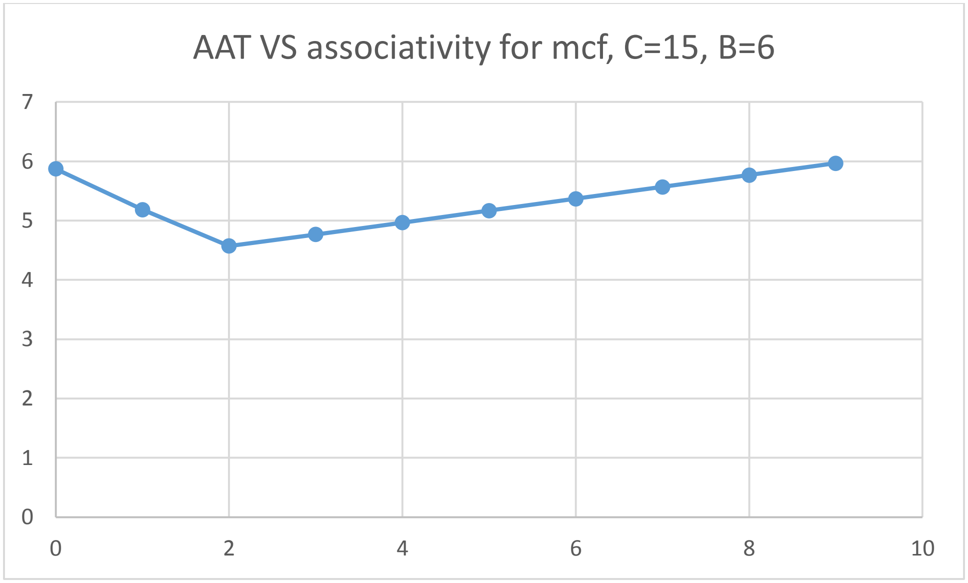
Similarly as in case of astar I analze case for fully associative cache for (B,S) equal to(6,9) and (5,10).

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	5	10	0	0	7.64901
2	15	6	9	0	0	5.96742

Taking (B,S) as (6,9) I decrease the associativity

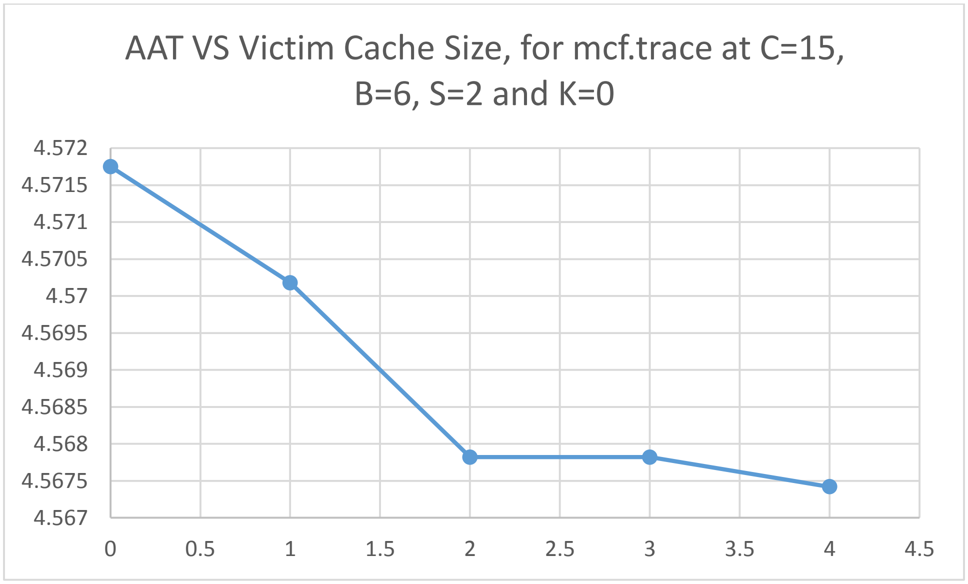
S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	6	9	0	0	5.96742
2	15	6	8	0	0	5.76742
3	15	6	7	0	0	5.56742
4	15	6	6	0	0	5.36742
5	15	6	5	0	0	5.16742
6	15	6	4	0	0	4.96742
7	15	6	3	0	0	4.76742
8	15	6	2	0	0	4.57175
9	15	6	1	0	0	5.18602
7	15	6	0	0	0	5.87355

This above data has a knee at s=2



For this trace we will look only into victim cache as for ubove run there was no prefetching in the trace. (Prefetched blocks were 0).

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	6	2	0	0	4.57175
2	15	6	2	1	0	4.57018
3	15	6	2	2	0	4.56782
4	15	6	2	3	0	4.56782
5	15	6	2	4	0	4.56742



Solution3) For mcf the cache parameters are:

C= 15

B =6

S =2

V =4

K =0

AAT = 4.56742ns

4. perlbench.trace:

Let’s analyze initial results for perlbench.trace:

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	5	3	4	2	10.502397
2	10	5	0	4	2	42.827998
3	15	5	3	0	0	11.272141
4	15	5	3	0	2	10.517768
5	15	5	3	4	0	11.264653

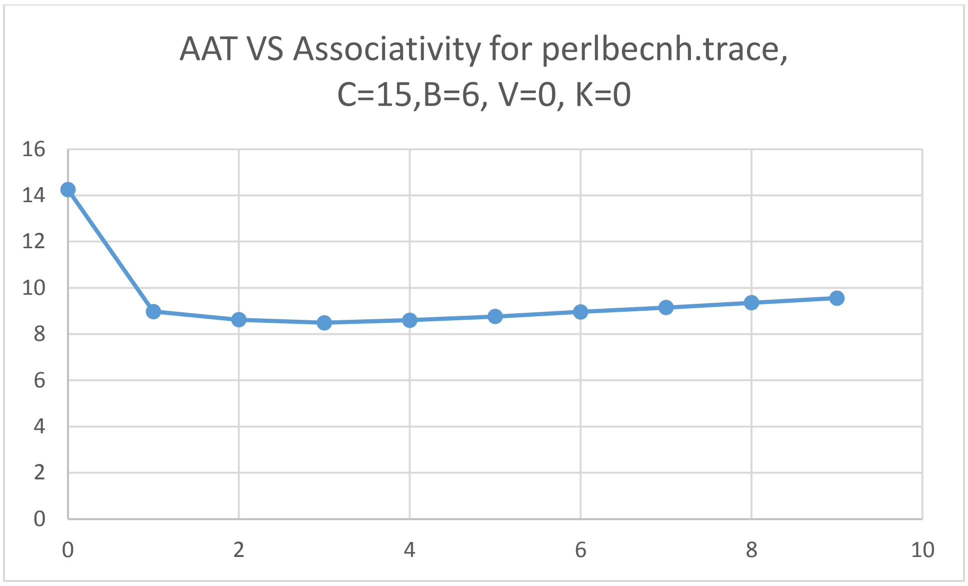
Similarly as in case of astar I analze case for fully associative cache for (B,S) equal to(6,9) and (5,10).

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	5	10	0	0	12.6828
2	15	6	9	0	0	9.55318

Taking (B,S) as (6,9) I decrease the associativity

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	6	9	0	0	9.55318
2	15	6	8	0	0	9.35673
3	15	6	7	0	0	9.14569
4	15	6	6	0	0	8.95791
5	15	6	5	0	0	8.76185
6	15	6	4	0	0	8.59929
7	15	6	3	0	0	8.48798
8	15	6	2	0	0	8.62299
9	15	6	1	0	0	8.97123
7	15	6	0	0	0	14.2481

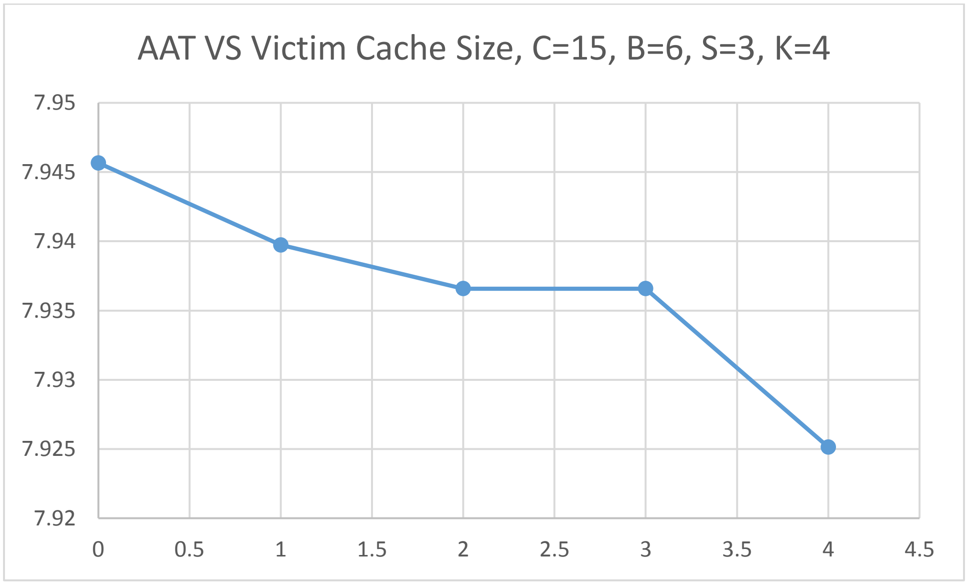
This above data has a knee at s=3



For this trace we will look only into victim cache and prefetcher:

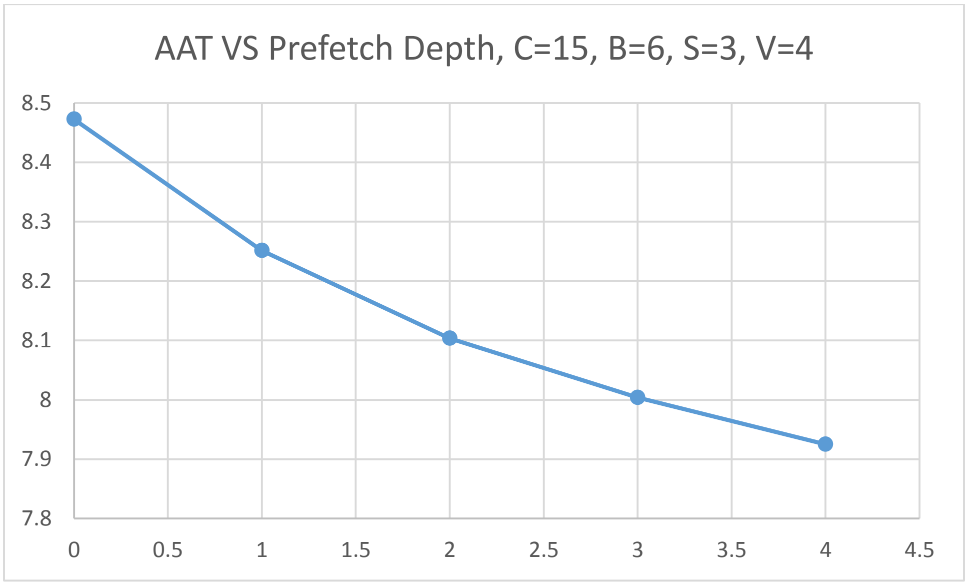
For k=4 varying V,

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	6	3	0	4	7.94565
2	15	6	3	1	4	7.93973
3	15	6	3	2	4	7.93658
4	15	6	3	3	4	7.93658
5	15	6	3	4	4	7.92515



For V=4, Varying K

S. No.	CacheSize (C)	Block Size (B)	Blocks Per Set (S)	Victim Cache(V)	Prefetcher depth(K)	AAT (ns)
1	15	6	3	4	0	8.4726
2	15	6	3	4	1	8.25149
3	15	6	3	4	2	8.10369
4	15	6	3	4	3	8.00398
5	15	6	3	4	4	7.92515



Solution3) For perlbench the cache parameters are:

C= 15

B =6

S =3

V =4

K =4

AAT = 7.92515ns