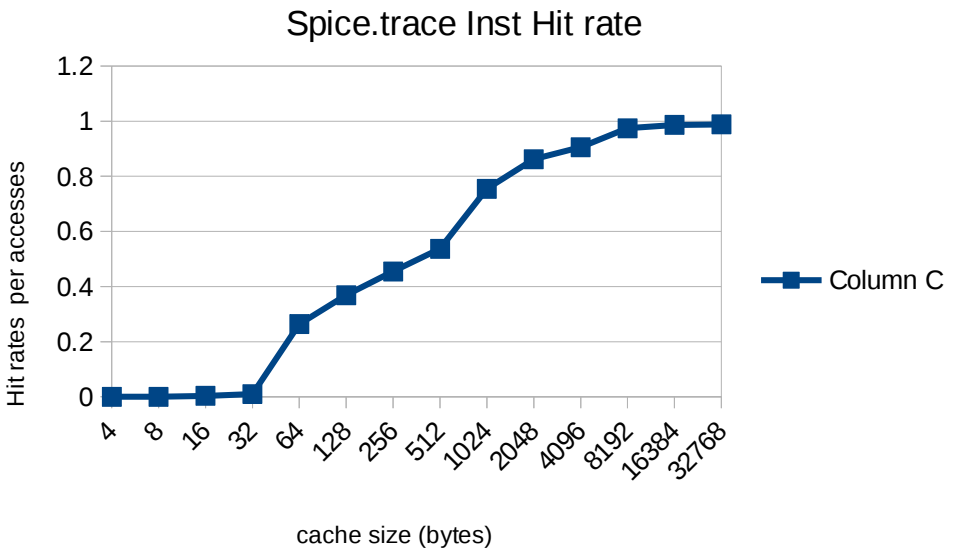
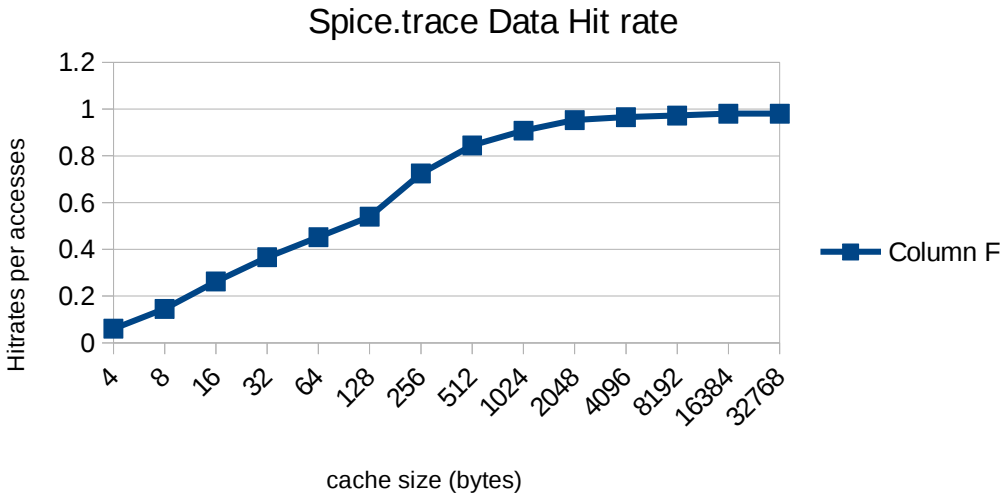
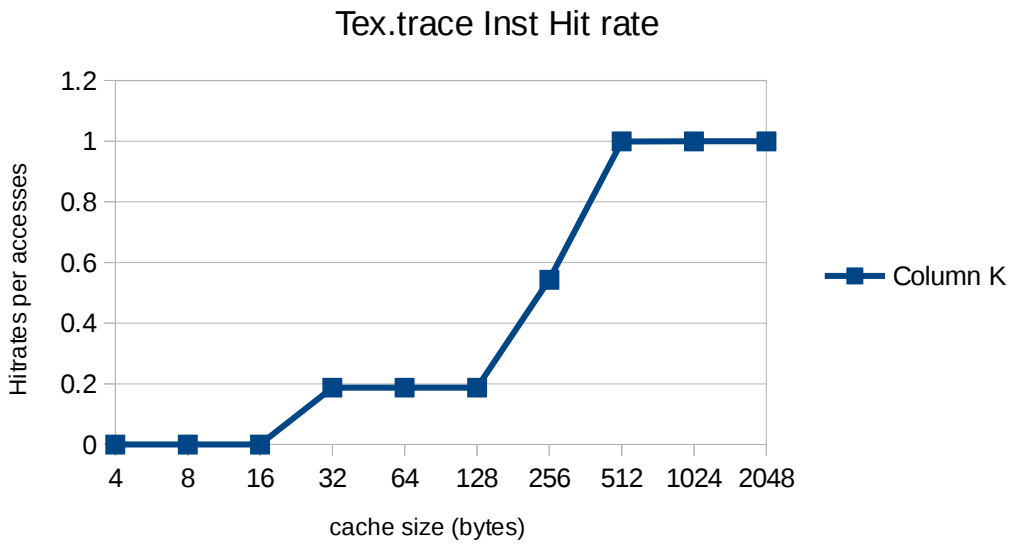
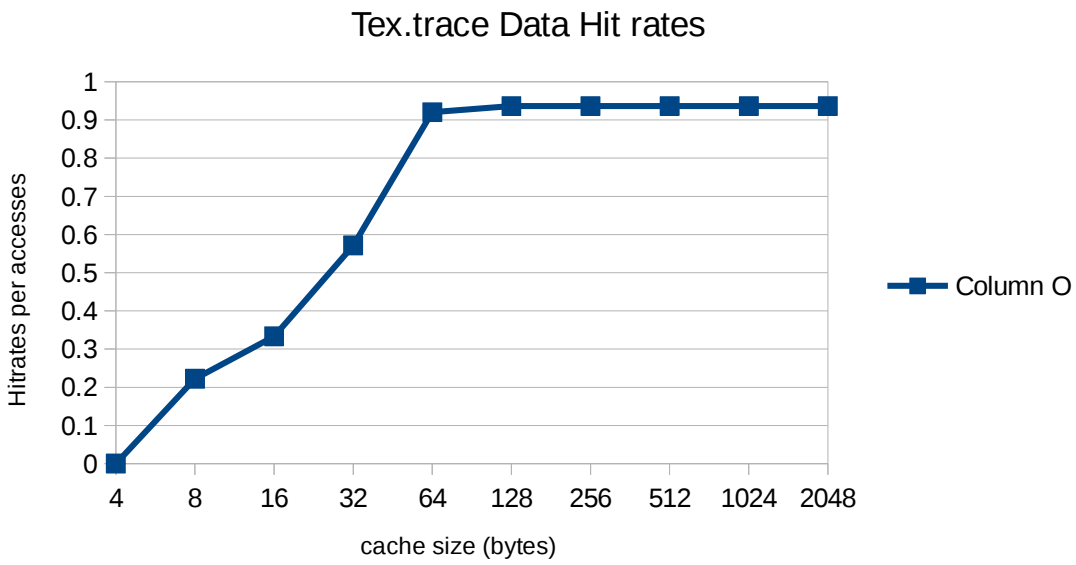
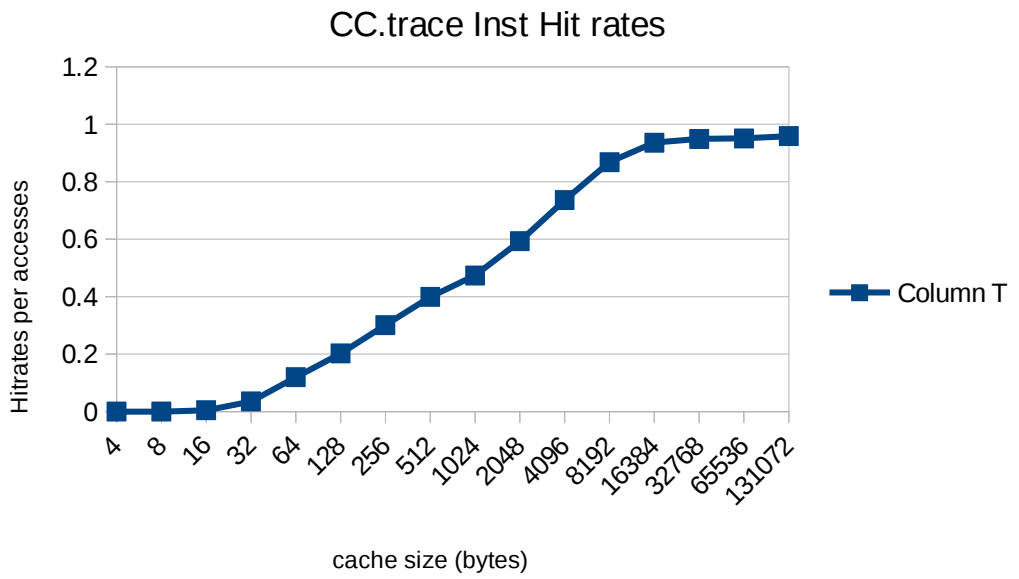
2.1 **Working Set Characterization**

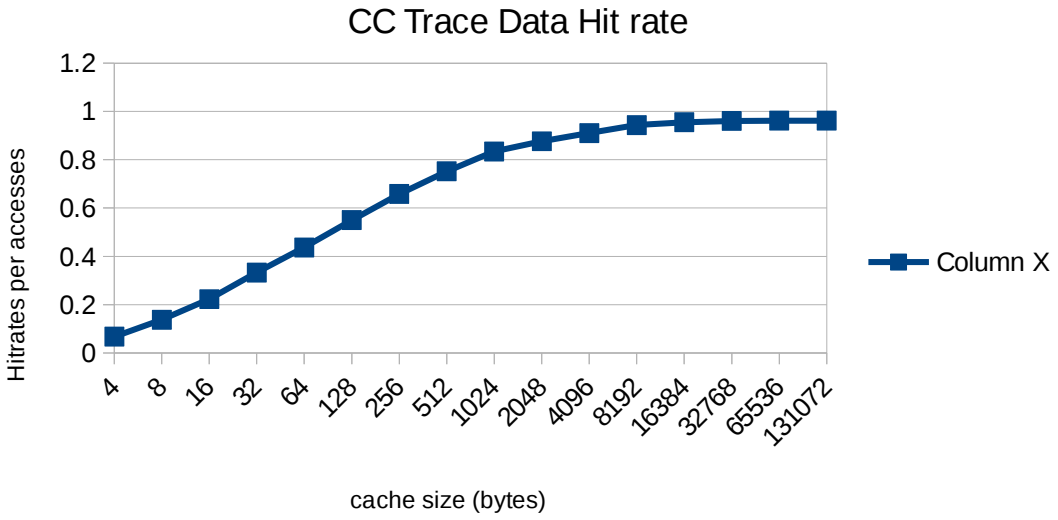





Tex.trace Inst Hit rate
Hitrates per accesses
1.2
1
0.8
0.6
0.4
0.2
0
4
8
16
32
64
128
256
512
1024
2048
cache size (bytes)
Column K


Tex.trace Data Hit rates
Hitrates per accesses
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
4
8
16
32
64
128
256
512
1024
2048
cache size (bytes)
Column O





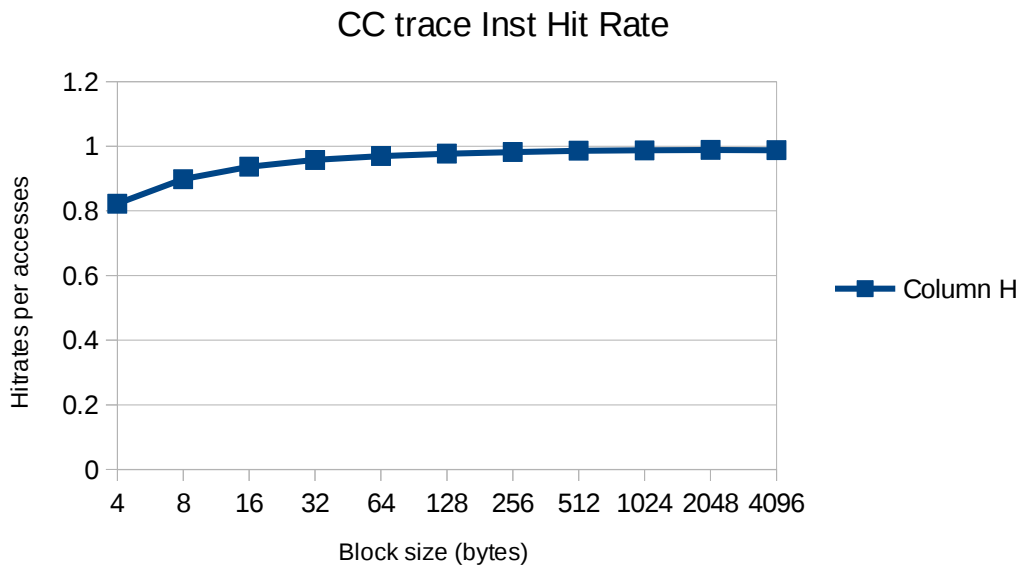
1. This experiment is slowly increasing the number of cache block and making the cache fully associative in order to see how many unique instruction and data reference are in the trace. The plot can be use to approximate the best cache size needed for the trace instruction and data reference. We can see from the plot that after the cache size increase above a certain threshold the number of hit does not increase as much .

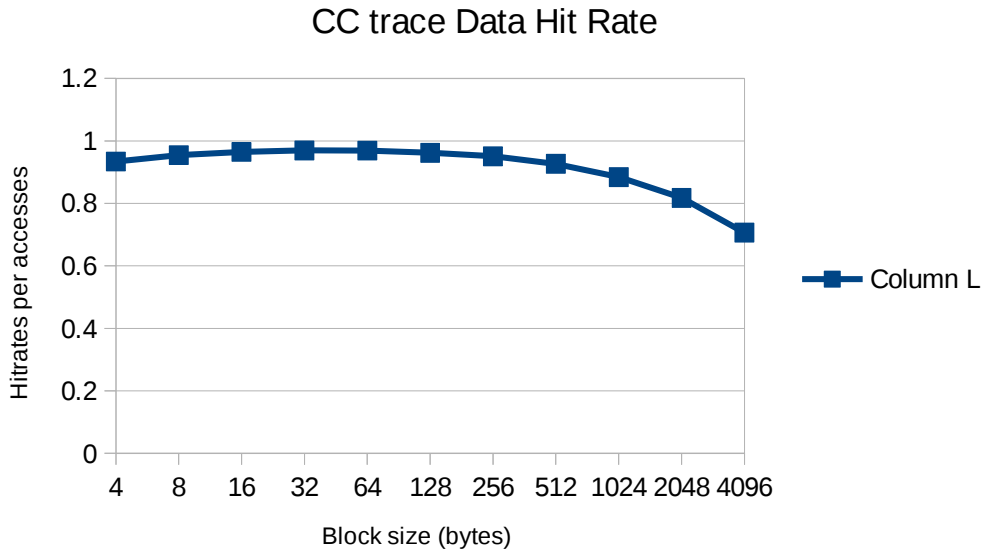
2. The total working size of instruction and data is equal to the maximum number of cache hit we can get with an infinite fully associative cache.

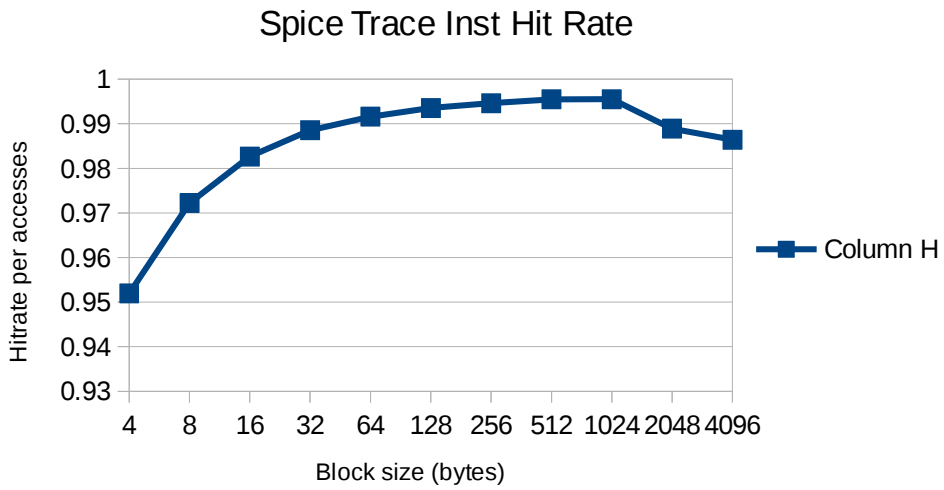
	Inst working size (approximate)	Data working size (approximate)
spice.trace	8964 instr	4225 data
cc.trace	31195 inst	9273 data
tex.trace	160 inst	9522 data

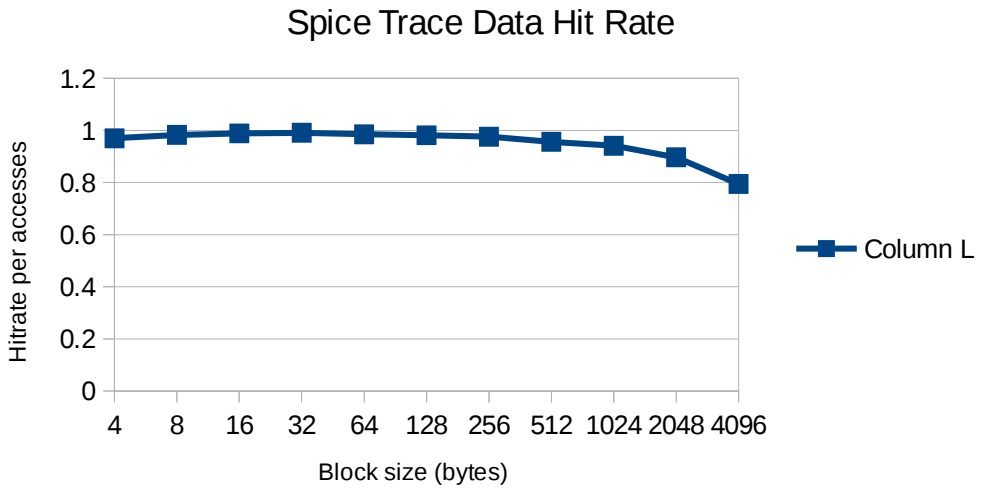
2.2

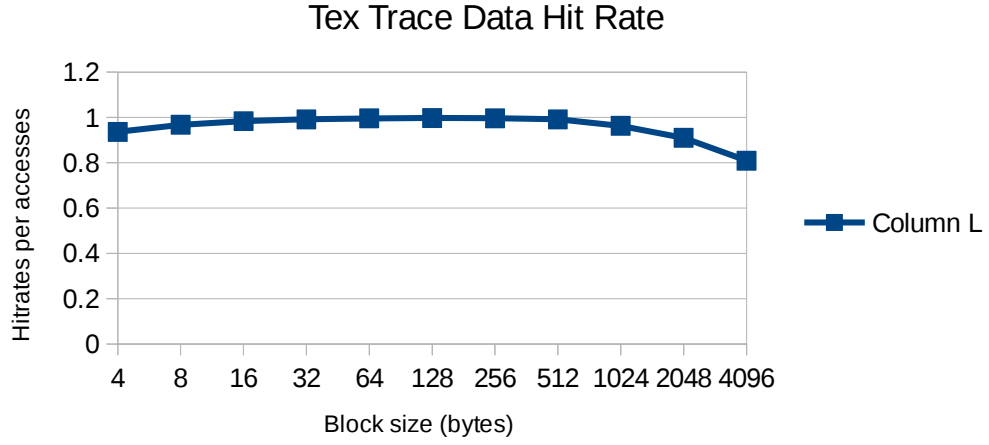
Impact of Block Size

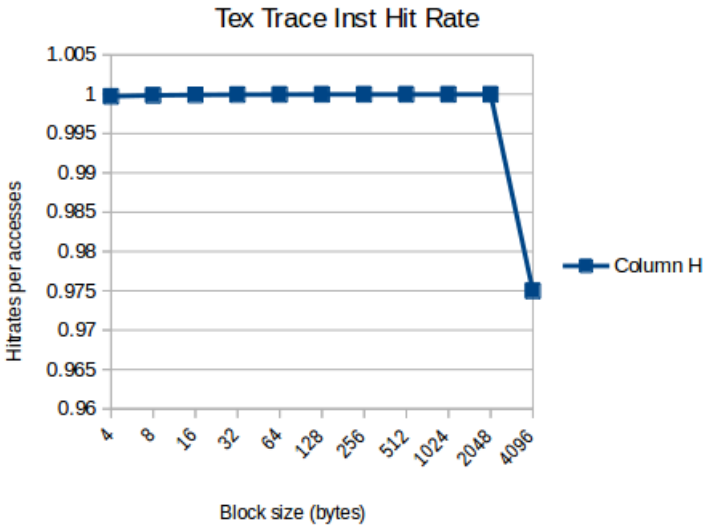





Spice Trace Inst Hit Rate
Hitrate per accesses
1
0.99
0.98
0.97
0.96
0.95
0.94
0.93
4
8
16
32
64
128
256
512
1024
2048
4096
Block size (bytes)
Column H


Spice Trace Data Hit Rate
Hitrate per accesses
1.2
1
0.8
0.6
0.4
0.2
0
4
8
16
32
64
128
256
512
1024
2048
4096
Block size (bytes)
Column L


Tex Trace Data Hit Rate
Hitrates per accesses
1.2
1
0.8
0.6
0.4
0.2
0
4
8
16
32
64
128
256
512
1024
2048
4096
Block size (bytes)
Column L

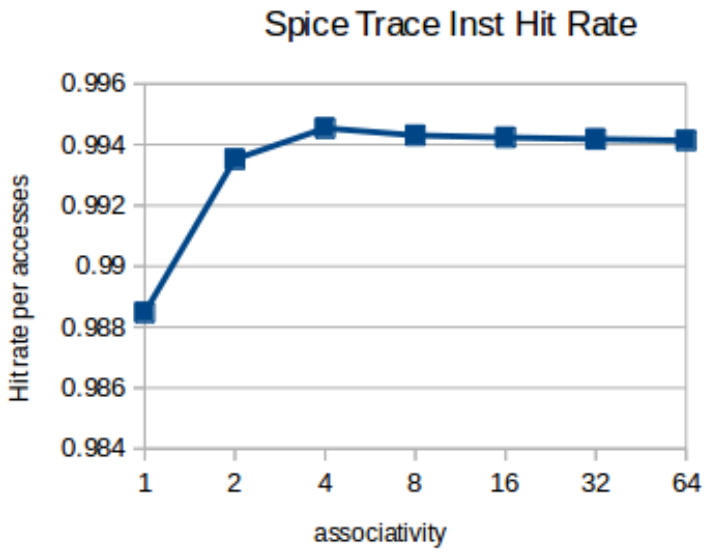


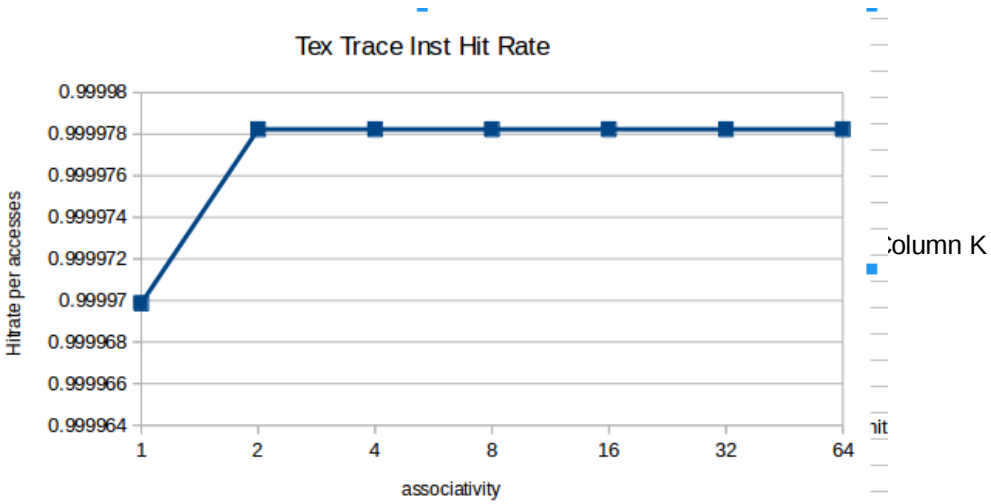
1. The graph has this shape because the bigger the cache block, the more instruction or data we can fetch from memory to the cache when we have a misses. When execute an instruction within a block, there is a strong probability that we will be executing an instruction within the same block. Thus is we fetch a large block we will have a hit on the next instruction or data access. But the larger the block get, the less set we have in the cache. On a miss we have to copies a lot more to memory and if the program has a lot of jump , we will have more cache miss

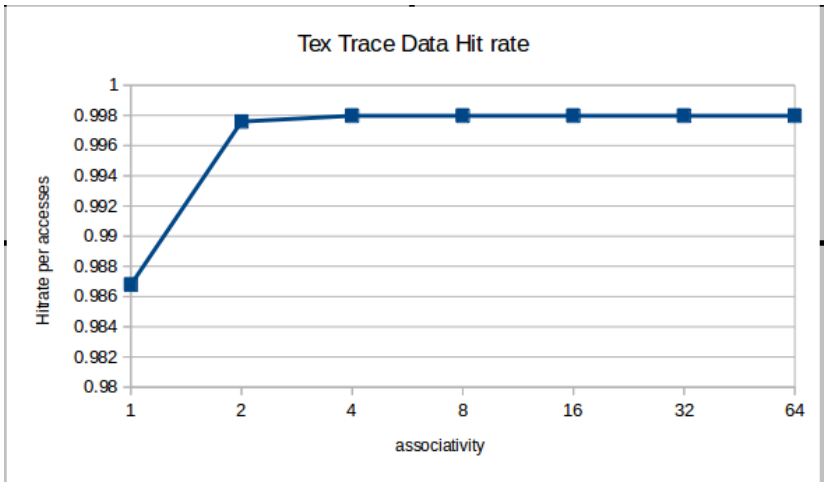
2. The optimal cache block size

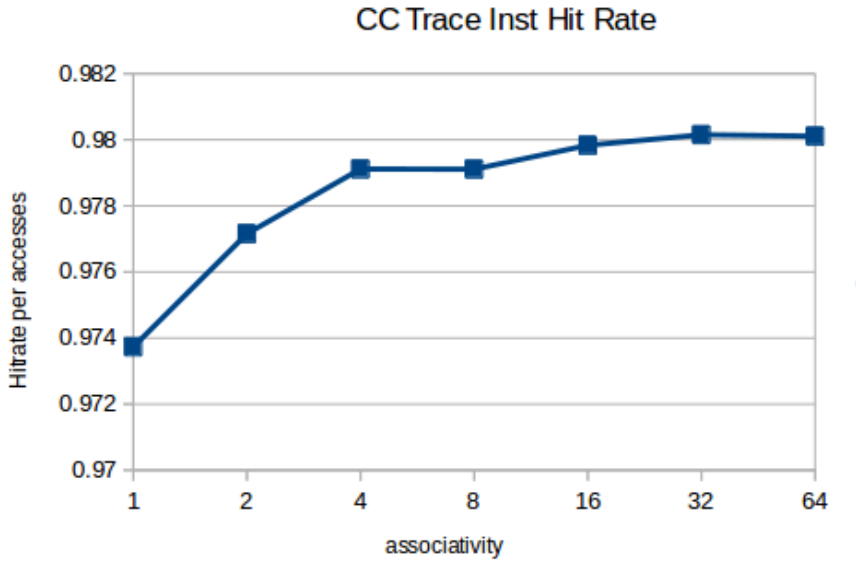
	Data cache block size	Inst cache block size
spice.trace	32	512
cc.trace	32	2048
tex.trace	128	4

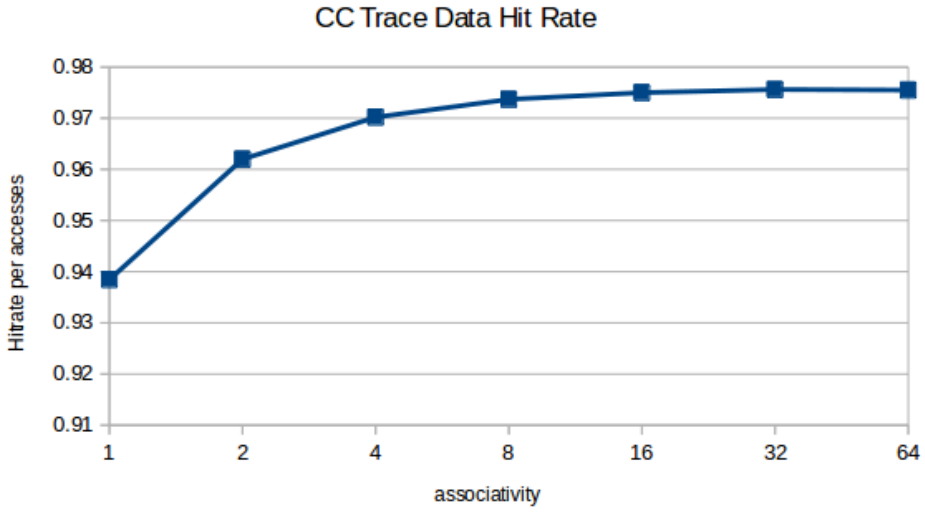
2.3
Impact of Associativity


Spice Trace Inst Hit Rate
0.996
0.994
0.992
0.99
0.988
0.986
0.984
Hit rate per accesses
1
2
4
8
16
32
64
associativity


Tex Trace Inst Hit Rate
0.99998
0.999978
0.999976
0.999974
0.999972
0.99997
0.999968
0.999966
0.999964
Hitrate per accesses
1
2
4
8
16
32
64
associativity
olumn K


Tex Trace Data Hit rate
Hitrate per accesses
1
0.998
0.996
0.994
0.992
0.99
0.988
0.986
0.984
0.982
0.98
1
2
4
8
16
32
64
associativity


CC Trace Inst Hit Rate
Hitrate per accesses
0.982
0.98
0.978
0.976
0.974
0.972
0.97
1
2
4
8
16
32
64
associativity


CC Trace Data Hit Rate
Hitrate per accesses
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
1
2
4
8
16
32
64
associativity

1. We can see from the above figure that the higher the associativity the higher the hit rate. This happens because each set has more blocks, so there's less chance of a conflict between two addresses.

2. There are no difference between the instruction and data figure. This tell us that both instruction and data can take full advantage of the higher association of the cache.

2.4
Memory Bandwidth
Spice.trace

Cache size	associativity	Block size	Write policies	Memory traffic per accesses
8192	2	64	WB and WNA	0.16
8192	2	64	WT and WNA	0.21
8192	4	64	WB and WNA	0.11
8192	4	64	WT and WNA	0.17
8192	2	128	WB and WNA	0.31
8192	2	128	WT and WNA	0.34
16384	4	128	WB and WNA	0.07
16384	4	128	WT and WNA	0.13
16384	2	128	WB and WNA	0.10
16384	2	128	WT and WNA	0.16

1. From the simulation ran , we can see from the table above that write back always have the smaller memory traffic. This is because in write back , we only access memory when we are evicting a dirty block or we are fetching a block from memory. In write through we access memory each time we are writing a block and when we are fetching a block from memory. In a program, writing a block happens more time than evicting a block.

2. Yes,. When the cache block is bigger and the programs isn't doing a lot of write then answer above will flip. For example if we have a block size of 128 byte and we are only writing a word , in this case a write through will have less memory traffic.

memory Bandwidth
Spice.trace

Cache size	associativity	Block size	Write policies	Memory traffic per accesses
8192	2	64	WB and WA	0.17
8192	2	64	WB and WNA	0.16
8192	4	64	WB and WA	0.117
8192	4	64	WB and WNA	0.116
8192	2	128	WB and WA	0.19
8192	2	128	WB and WNA	0.18
16384	4	128	WB and WA	0.0735
16384	4	128	WB and WNA	0.0739
16384	2	128	WB and WA	0.11
16384	2	128	WB and WNA	0.10

1. Based on the result of the table above , the write no allocate cache configuration has the smaller memory traffic. This may be because the above program isn't doing a lot of write .

2. If the program was doing more write to the same block , the answer above will flip. For example if the program is iterating two time through an array that fit in a block, the write no allocate cache configuration will write two word per block to memory where as the write allocate cache configuration will only write a word per block to memory when the block is evicted from the cache.