Project: Cache Organization and Performance Evaluation

Author: Abdur Razzak

A. Cache Simulator

Please find the cache.c in the submission.

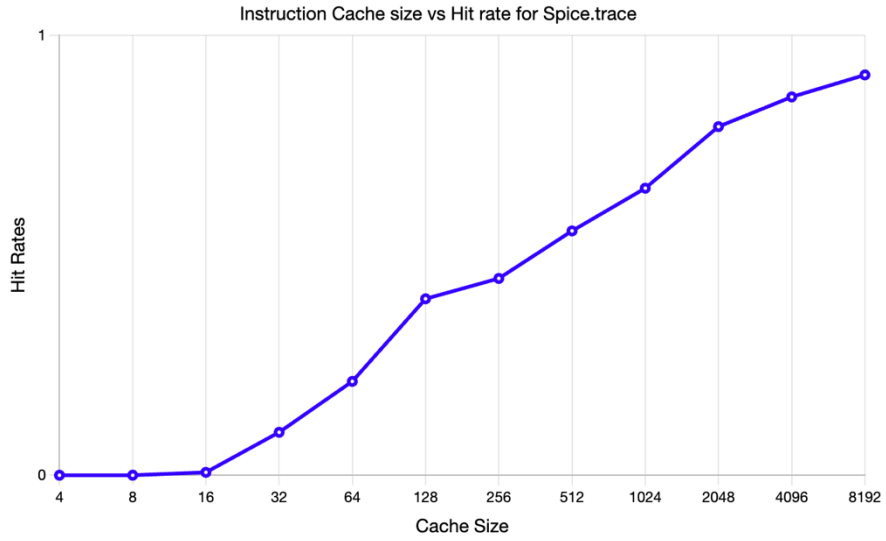
B. Performance Evaluation

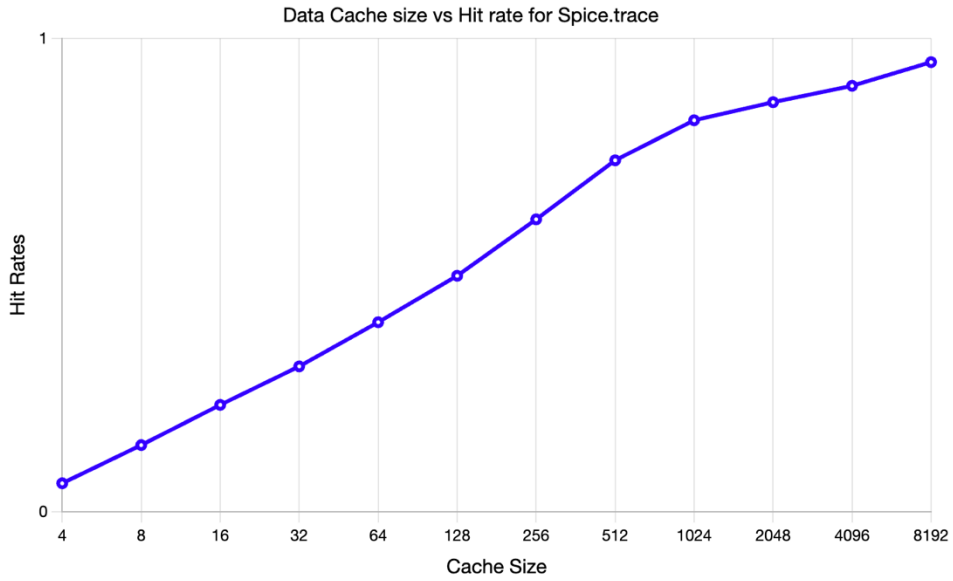
Working Set Characterization:

1. **Spice Trace Test:**
   Setup: Block size 4, associativity 1, data and instruction size varies from 4B to 8KB

   Sample Command Run: ./sim -bs 4 -is 32 -ds 32 -a 1 -wb -wa ../traces/spice10.trace

Instruction: Spice.trace		Data: Spice.trace	
Cache Size	**Hit Rates**	**Cache size**	**Hit Rate**
4	0	4	0.0604
8	0.0001	8	0.1411
16	0.0066	16	0.226
32	0.0977	32	0.3073
64	0.2134	64	0.4006
128	0.4011	128	0.4987
256	0.4474	256	0.6179
512	0.5554	512	0.7425
1024	0.6525	1024	0.8272
2048	0.7925	2048	0.8654
4096	0.86	4096	0.9
8192	0.91	8192	0.95

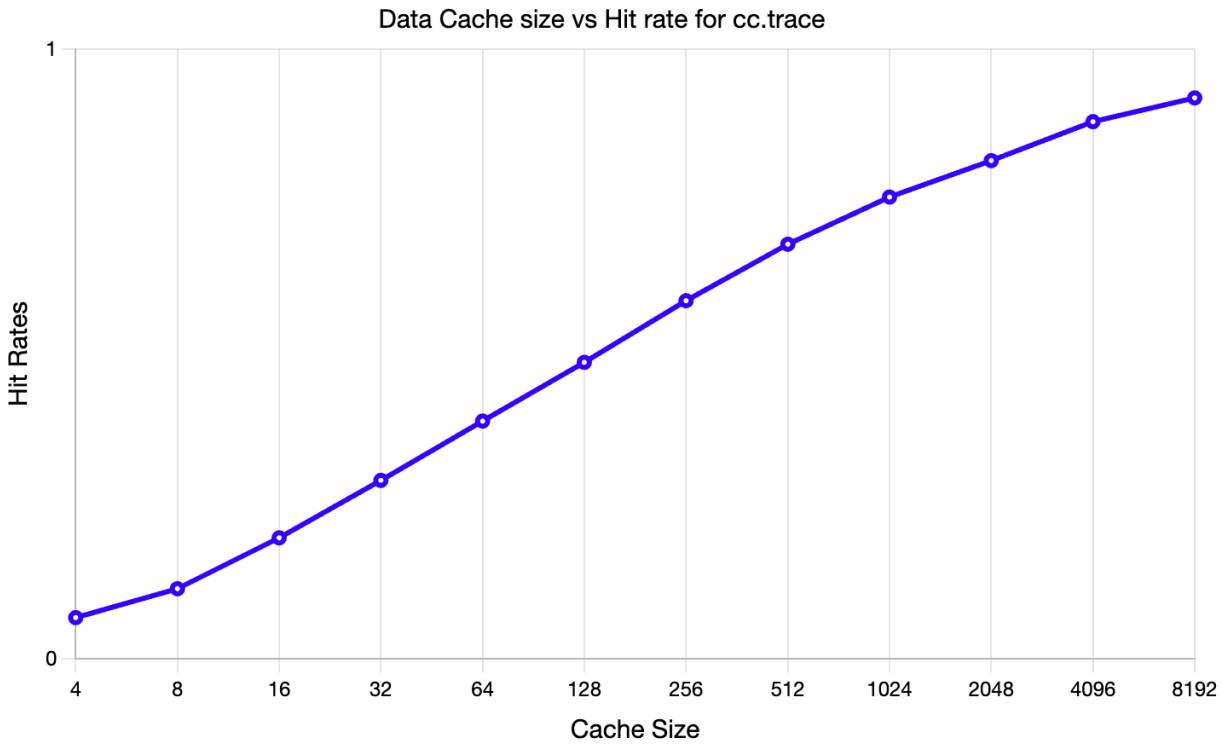


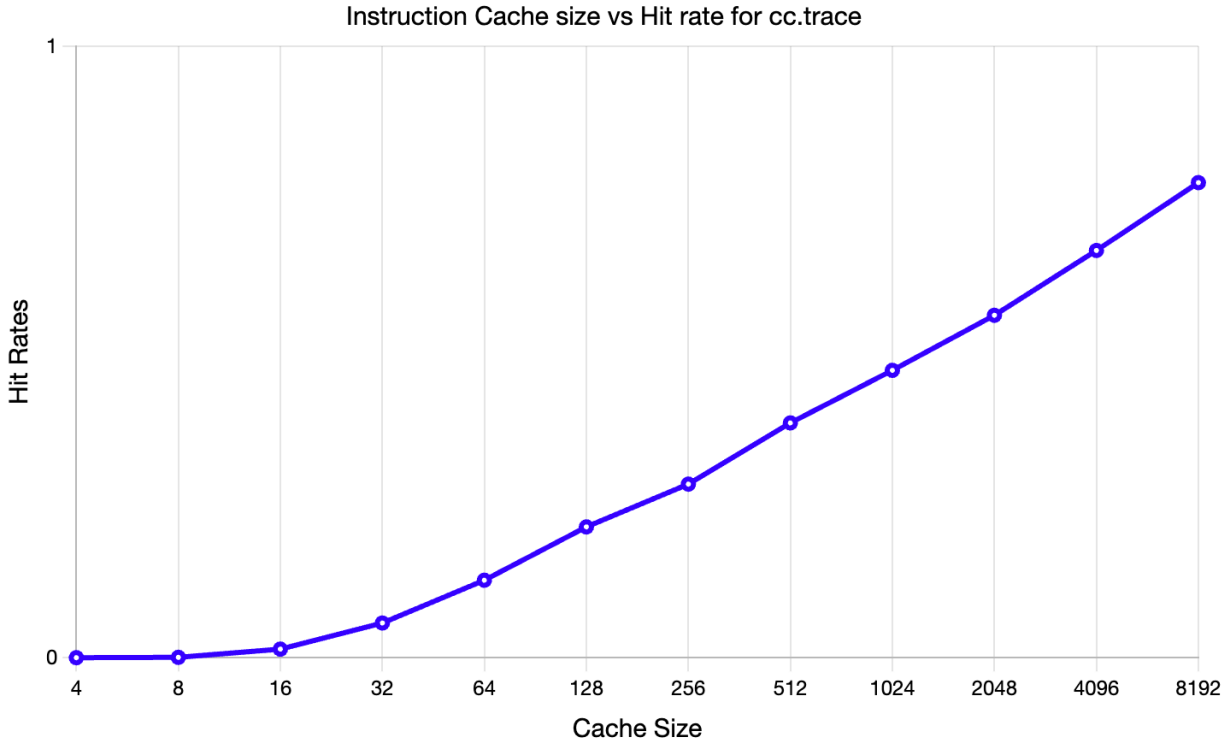


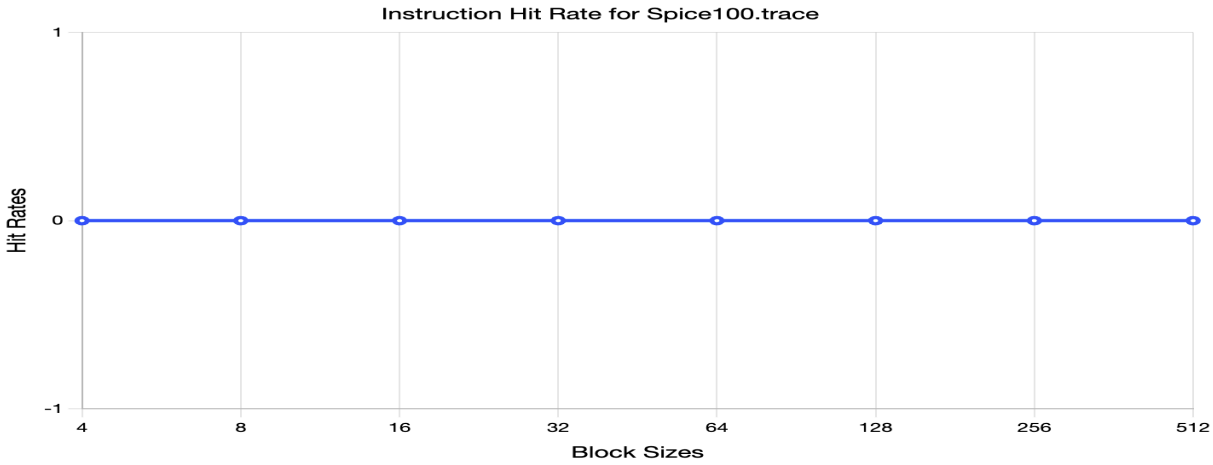
2. **Cc.trace Test: Block size from 4, associativity 1, data and instruction size varies from 4B to 8KB**

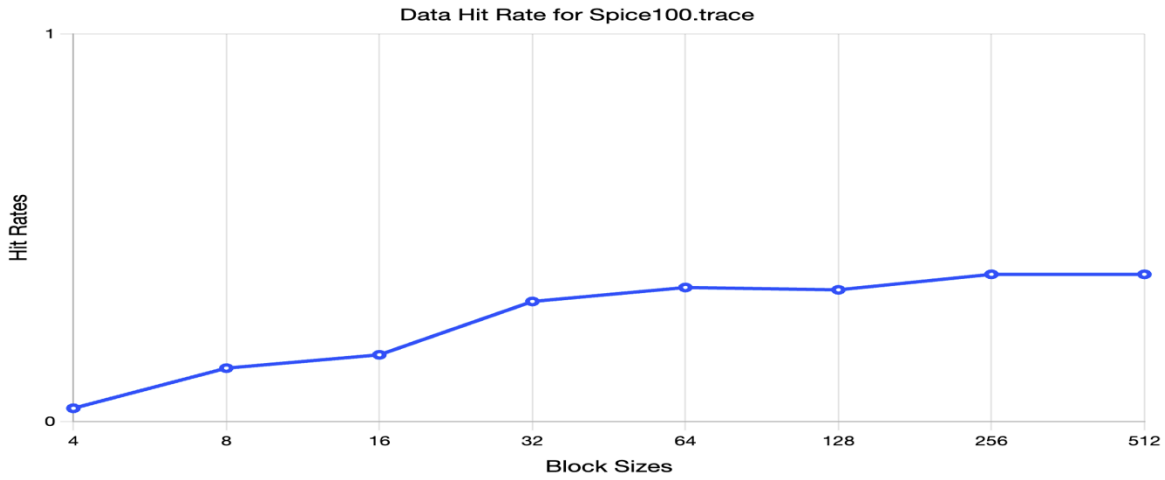
   Sample Command: ./sim -bs 4 -is 8192 -ds 8192 -a 1 -wb -wa ../traces/cc.trace

Instruction: cc.trace		Data: cc.trace	
Cache Size	**Hit Rates**	**Cache size**	**Hit Rate**
4	0	4	0.0676
8	0.0006	8	0.115
16	0.014	16	0.1985
32	0.0567	32	0.2927
64	0.1267	64	0.39
128	0.2139	128	0.4863
256	0.2851	256	0.5874
512	0.384	512	0.68
1024	0.47	1024	0.7574
2048	0.56	2048	0.8171
4096	0.666	4096	0.881
8192	0.777	8192	0.92


Data Cache size vs Hit rate for cc.trace
Hit Rates
1
0
4
8
16
32
64
128
256
512
1024
2048
4096
8192
Cache Size


Instruction Cache size vs Hit rate for cc.trace
Hit Rates
1
0
4
8
16
32
64
128
256
512
1024
2048
4096
8192
Cache Size

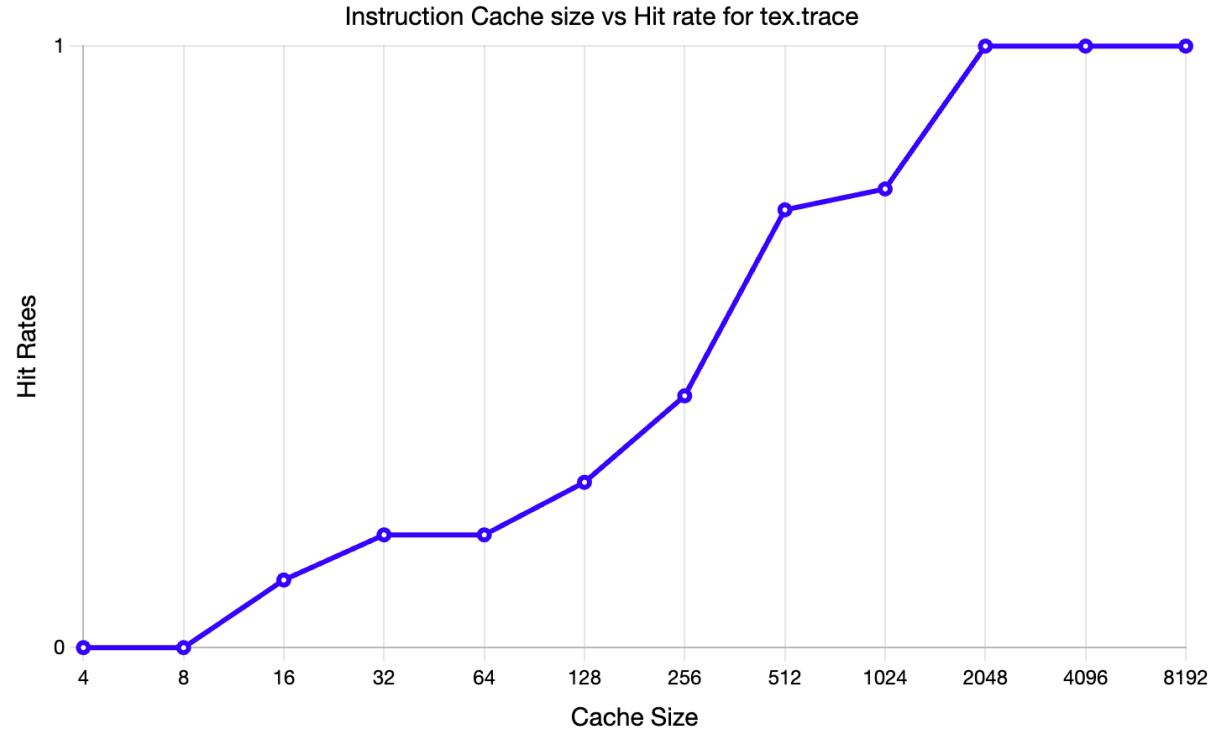


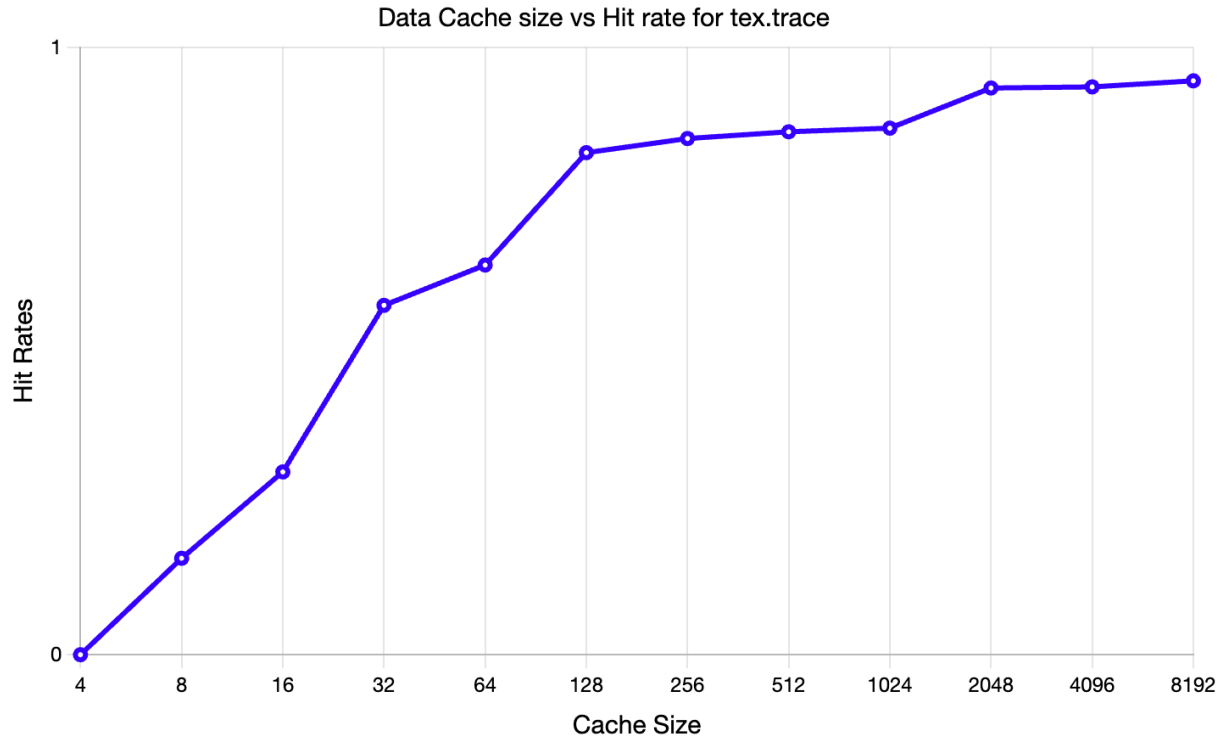


3. Tex Test: Block size from 4, associativity 1, data and ins size varies from 4B to 8KB

Sample Instruction: ./sim -bs 4 -is 8196 -ds 8196 -a 1 -wb -wa ../traces/tex.trace

Instruction: tex.trace		Data: tex.trace	
Cache Size	**Hit Rates**	**Cache size**	**Hit Rate**
4	0	4	0.0001
8	0	8	0.1588
16	0.1124	16	0.301
32	0.1874	32	0.5753
64	0.1874	64	0.6417
128	0.2748	128	0.8267
256	0.4185	256	0.85
512	0.7276	512	0.8611
1024	0.7623	1024	0.8671
2048	0.9997	2048	0.9332
4096	0.9997	4096	0.935
8192	0.9997	8192	0.945


Instruction Cache size vs Hit rate for tex.trace
Hit Rates
1
0
4
8
16
32
64
128
256
512
1024
2048
4096
8192
Cache Size


Data Cache size vs Hit rate for tex.trace
Hit Rates
1
0
4
8
16
32
64
128
256
512
1024
2048
4096
8192
Cache Size

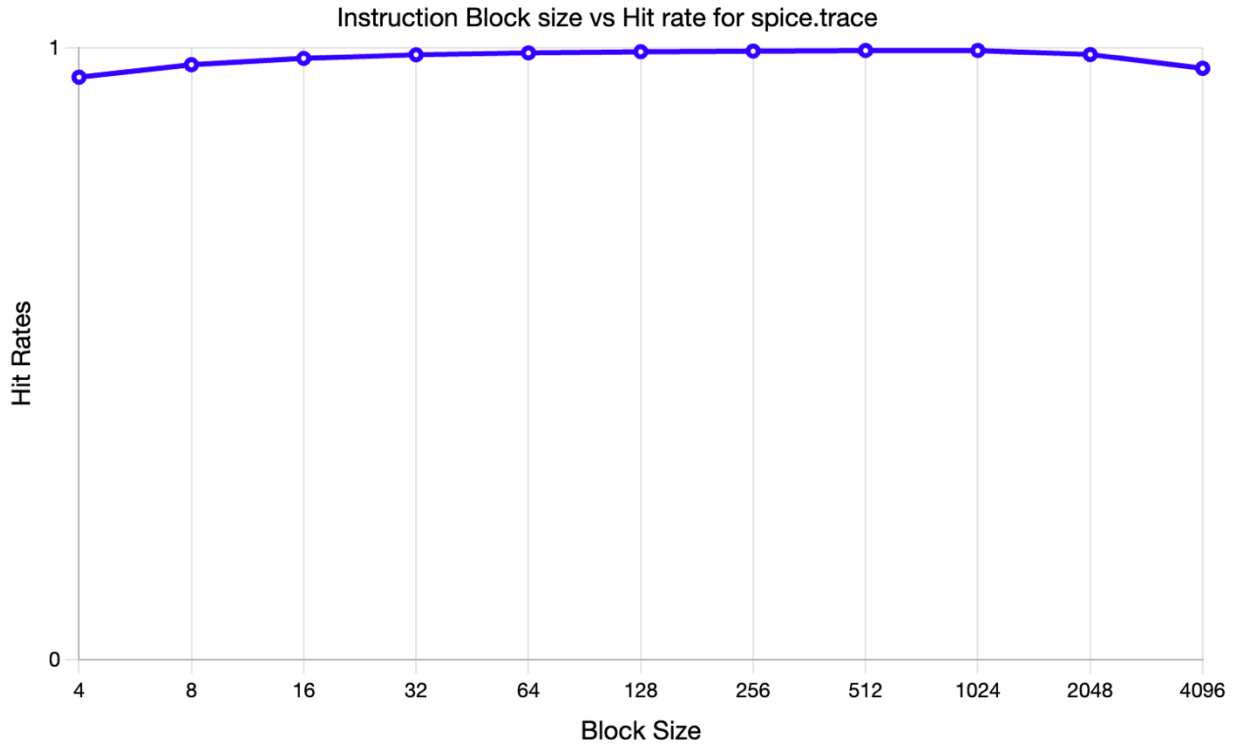
Answer to the Questions:
1. This experiment is increasing the cache sizes keeping the instructions same and testing the hit rates. The intension is to check whether the hit rates increase over the increase of cache size. The logical view says the hit rates should increases as the cache size increase, less conflict misses. However, compulsory misses are always there. And it reflects on the graphs.

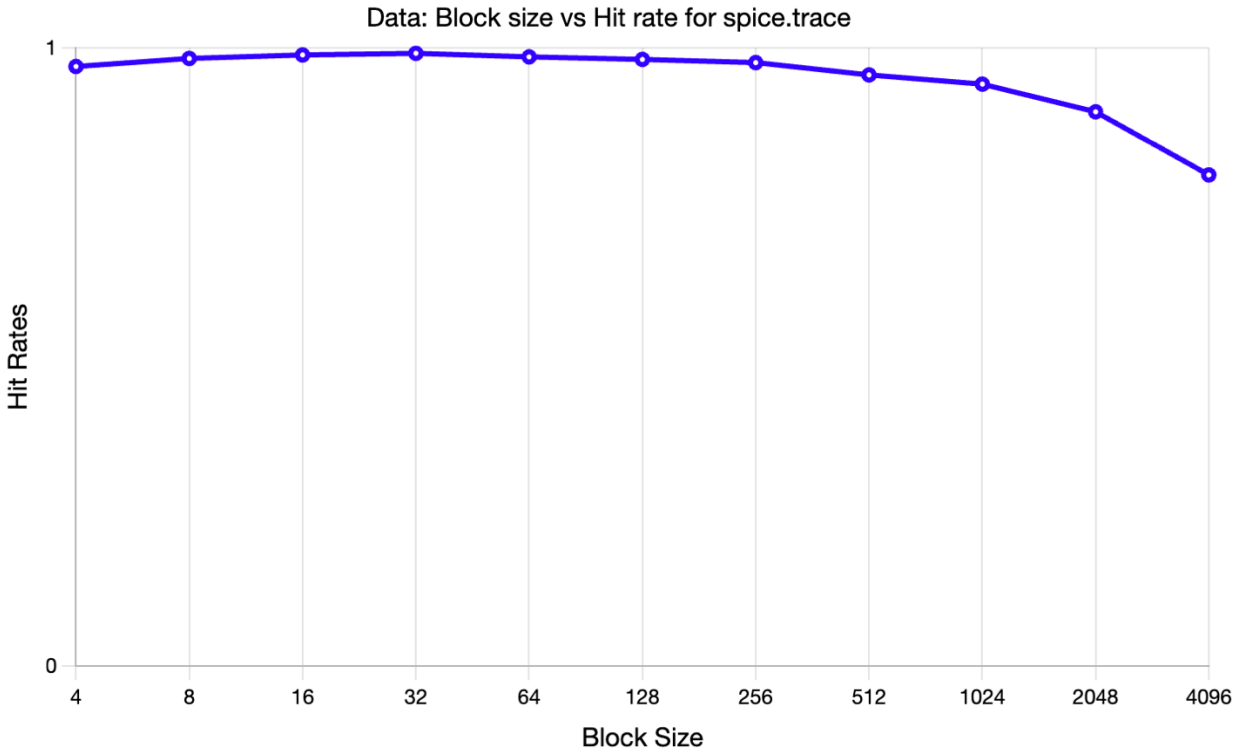
2. spice.trace: 782764 Instructions, 217237 Data accesses
cc.trace: 757341 Instructions, 242661 Data accesses
tex.trace: 597309 Instructions, 235168 Data accesses

Impact of Block Size: Setting: I and D cache 8196, associativity 2, wb, wa, bs [4B to 4KB]

1. spice.trace Test:
   Sample Instruction: ./sim -bs 4 -is 8196 -ds 8196 -a 2 -wb -wa ../traces/spice.trace

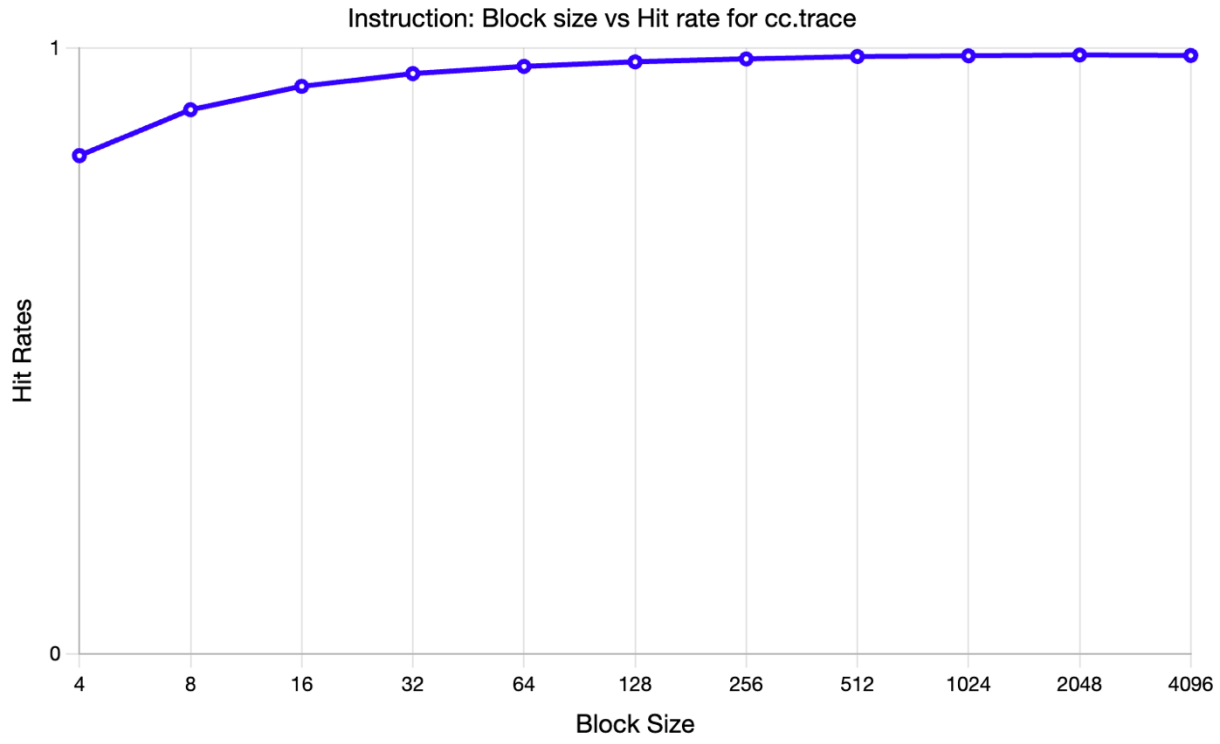
Instruction: spice.trace		Data: spice.trace	
Block Size	**Hit Rates**	**Block size**	**Hit Rate**
4	0.9519	4	0.97
8	0.9722	8	0.983
16	0.9826	16	0.9886
32	0.9885	32	0.9912
64	0.9916	64	0.9855
128	0.9935	128	0.9814
256	0.9946	256	0.9763
512	0.9955	512	0.9564
1024	0.9955	1024	0.9415
2048	0.9889	2048	0.8968
4096	0.9664	4096	0.7947

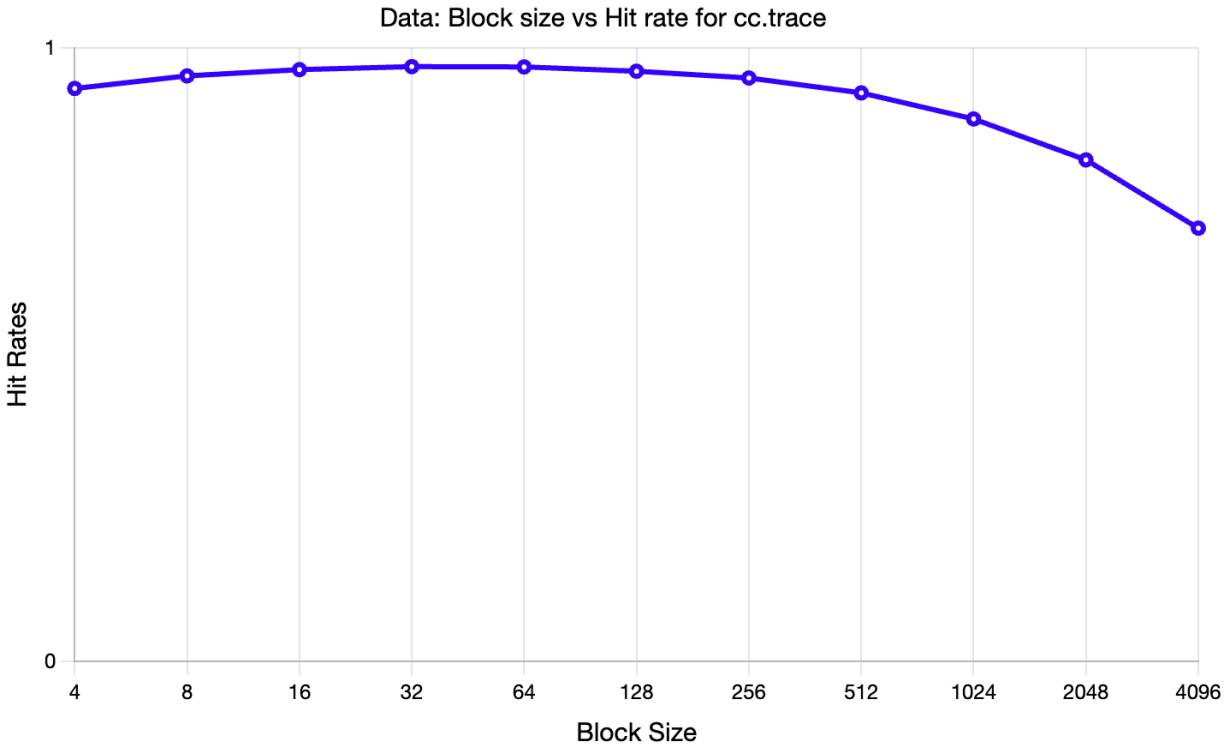

Instruction Block size vs Hit rate for spice.trace
Hit Rates
1
0
4
8
16
32
64
128
256
512
1024
2048
4096
Block Size


Data: Block size vs Hit rate for spice.trace
Hit Rates
1
0
4
8
16
32
64
128
256
512
1024
2048
4096
Block Size

2. cc.trace Test:
   ./sim -bs 4 -is 8196 -ds 8196 -a 2 -wb -wa ../traces/cc.trace

Instruction: cc.trace		Data: cc.trace	
Block Size	**Hit Rates**	**Block size**	**Hit Rate**
4	0.8225	4	0.9339
8	0.8982	8	0.9544
16	0.9368	16	0.9646
32	0.9578	32	0.9695
64	0.9696	64	0.969
128	0.9772	128	0.962
256	0.982	256	0.951
512	0.986	512	0.9267
1024	0.9872	1024	0.8842
2048	0.9885	2048	0.8175
4096	0.9877	4096	0.7062





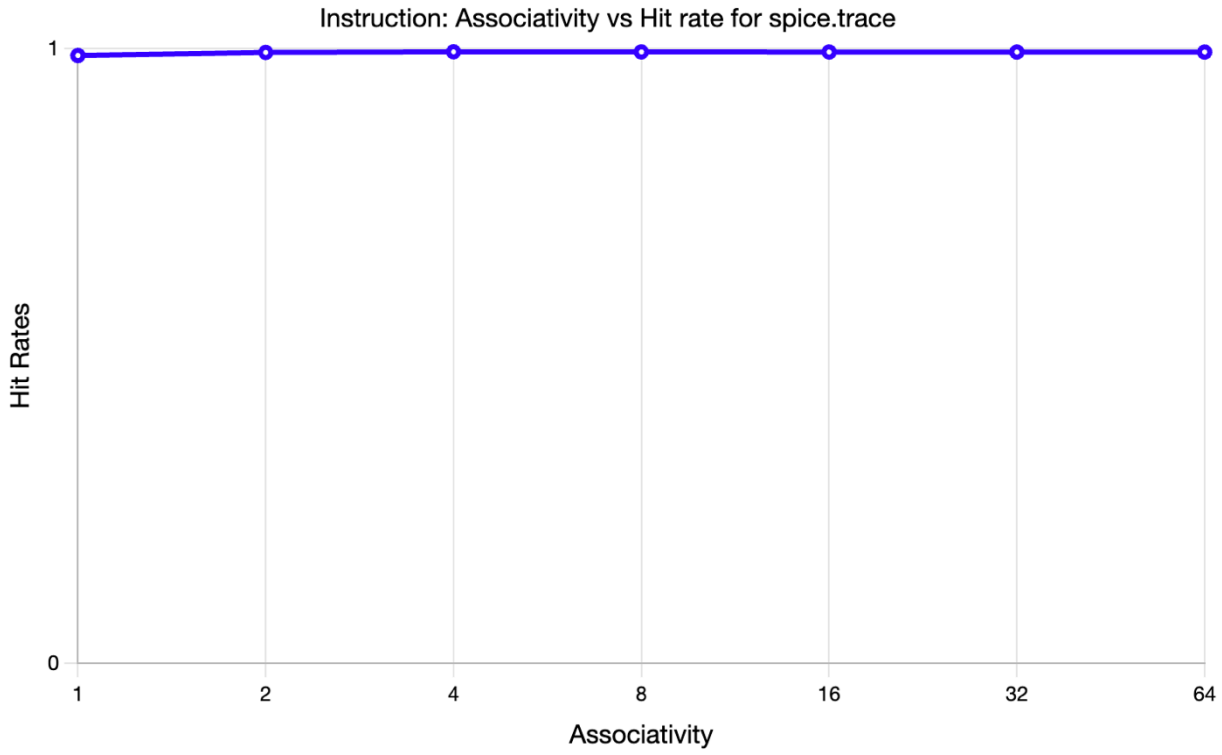
Answer to the Questions:

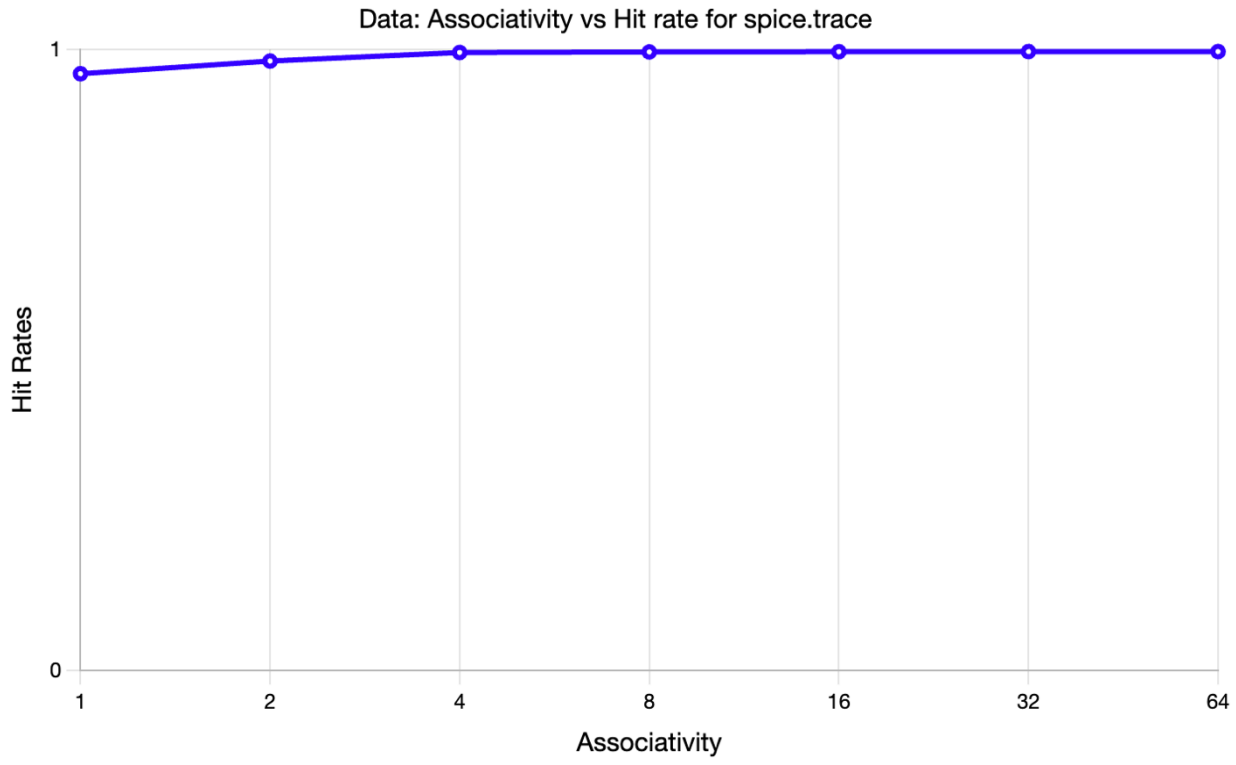
1. The block size increase causes to give room for temporal or spatial locality data. However, when spatial locality brings some nearby data that is not being used later time, then it will cause more cache misses. Therefore, it could curve a little down.
2. For both Instruction and Data, 64B is good in my opinion.
3. Yes, optimal block is different for Instruction and data references. It is because data references require to write back to the cache where instruction requires to read and execute, nothing going back to memory.

Impact of Associativity: Block size 128, Instruction and data cache 8KB, associativity will vary from 1 to 64.

1. Spice.trace Test:
   ./sim -bs 128 -is 8196 -ds 8196 -a 1 -wb -wa ../traces/spice.trace

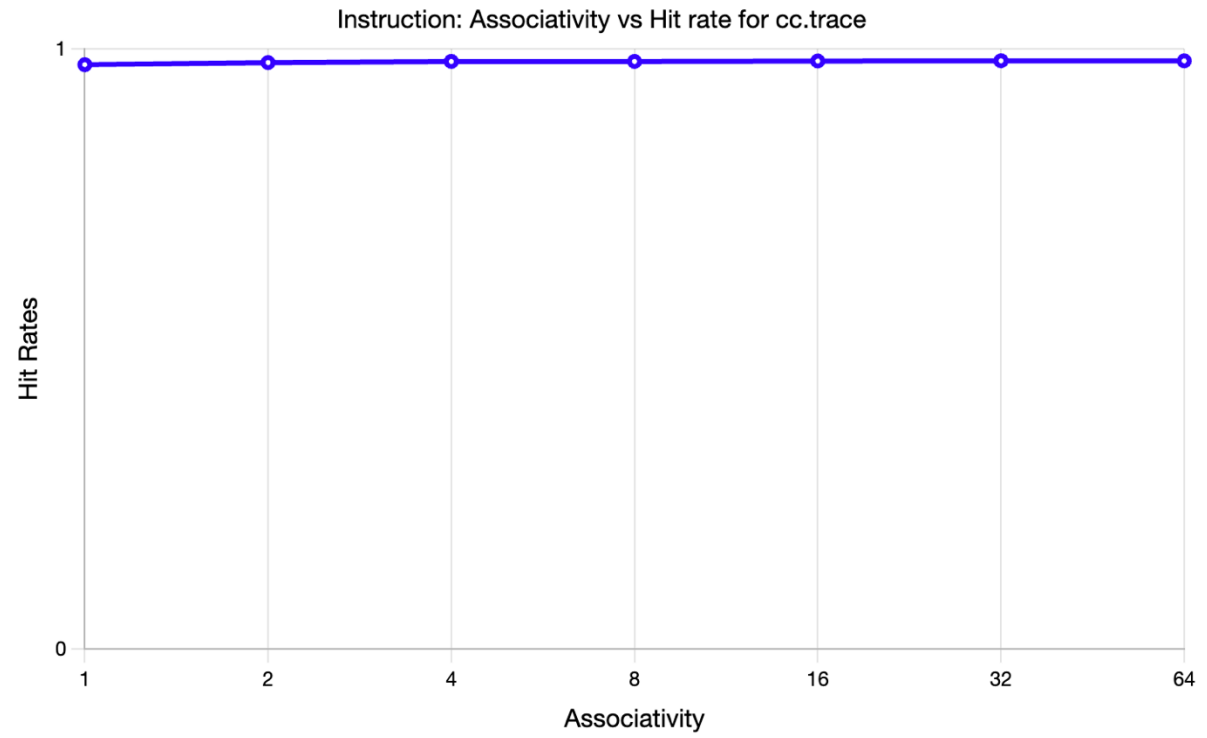
Instruction: spice.trace		Data: spice.trace	
Associativity	**Hit Rates**	**Associativity**	**Hit Rate**
1	0.9885	1	0.961
2	0.9935	2	0.9814
4	0.9945	4	0.9951
8	0.9943	8	0.996
16	0.9942	16	0.9964
32	0.9942	32	0.9966
64	0.9941	64	0.9966

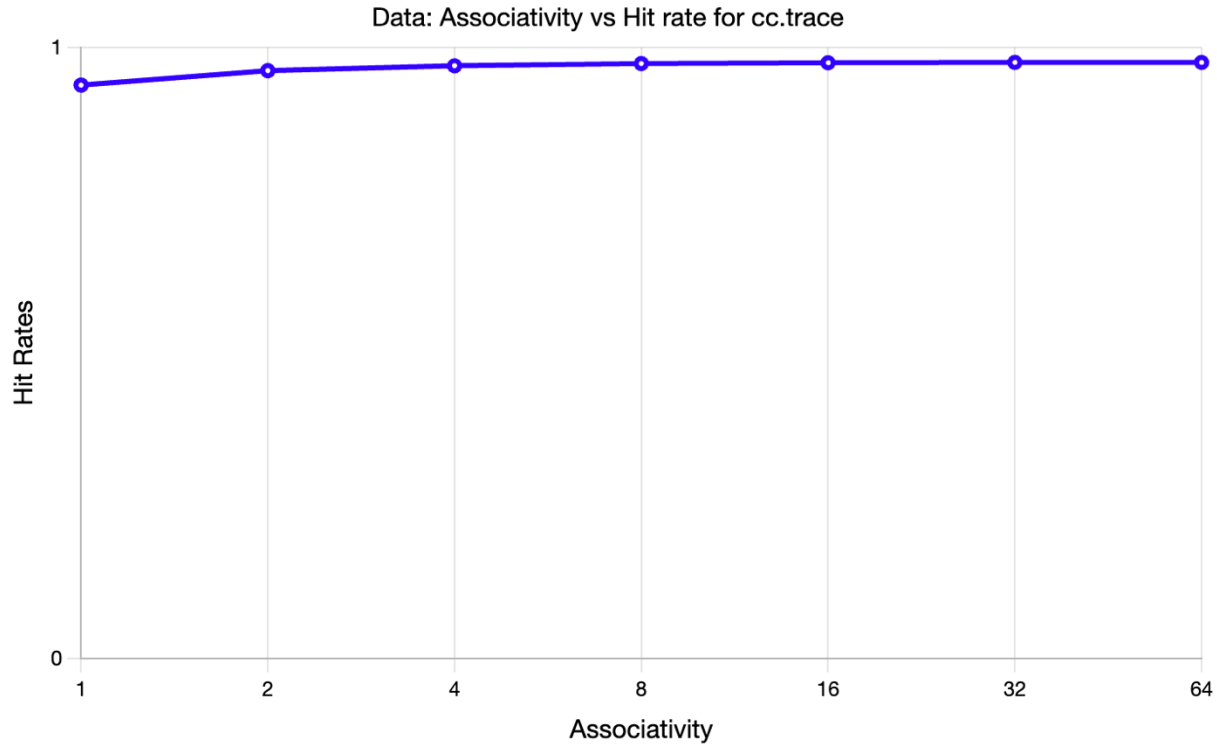




2. Cc.trace Test:
   ./sim -bs 128 -is 8196 -ds 8196 -a 64 -wb -wa ../traces/cc.trace

Instruction: cc.trace		Data: cc.trace	
Associativity	**Hit Rates**	**Associativity**	**Hit Rate**
1	0.9737	1	0.9385
2	0.9772	2	0.962
4	0.9791	4	0.97
8	0.9791	8	0.9737
16	0.9798	16	0.975
32	0.9802	32	0.9756
64	0.9801	64	0.9755

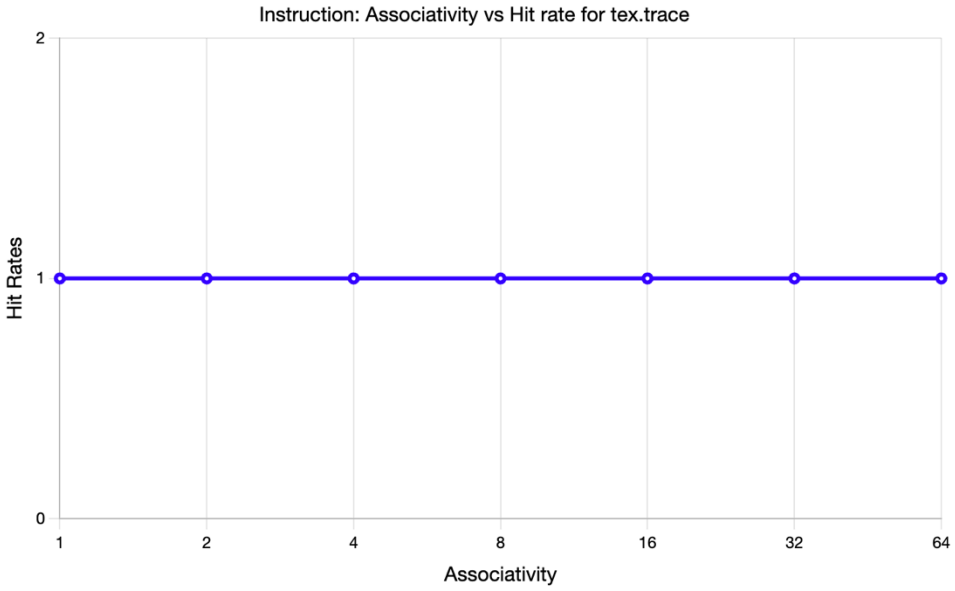

Instruction: Associativity vs Hit rate for cc.trace
Hit Rates
1
0
1
2
4
8
16
32
64
Associativity

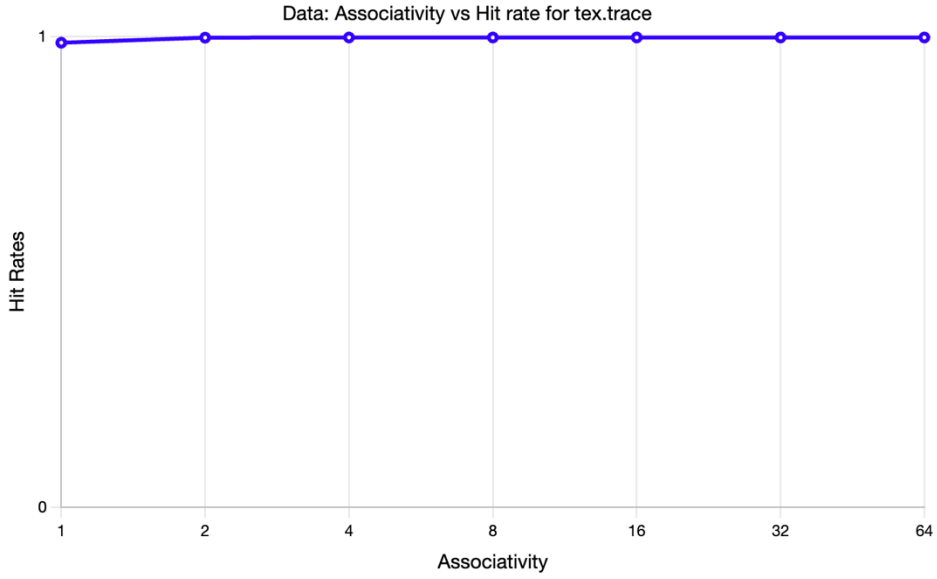

Data: Associativity vs Hit rate for cc.trace
Hit Rates
1
0
1
2
4
8
16
32
64
Associativity

3. Tex.trace Test:

./sim -bs 128 -is 8196 -ds 8196 -a 32 -wb -wa ../traces/tex.trace

Instruction: tex.trace		Data: tex.trace	
Associativity	**Hit Rates**	**Associativity**	**Hit Rate**
1	1	1	0.9868
2	1	2	0.9976
4	1	4	0.998
8	1	8	0.998
16	1	16	0.998
32	1	32	0.998
64	1	64	0.998





Answer to the questions:

1. Increasing Associativity increases more room in one set so that conflict misses will be reduced. Therefore, the graph shows like this way.
2. As instruction does not need to write back, increasing more and more associativity will not help after one time insertion into the cache. Hence it will be 1 for rest of the executions regardless increase of associativity.

Memory Bandwidth: (Only test with spice.trace)

	Settings	Demand Fetch	Copies Back
Write-Through	bc: 128, cache: 16384, assoc: 4	66304	66538
	bc: 64, cache: 16384, assoc: 2	59856	66538
	bc: 64, cache: 8192, assoc: 4	110400	66538
	bc: 128, cache: 8192, assoc: 4	171136	66538
Write-back	bc: 128, cache: 16384, assoc: 4	66304	7200
	bc: 64, cache: 16384, assoc: 2	59856	6160
	bc: 64, cache: 8192, assoc: 4	110400	7168
	bc: 128, cache: 8192, assoc: 4	171136	14144

Answer to the Questions:

1. Write-back is taking less memory traffic here. Wright-though causes more memory traffic because when a write operation use wt concept it must update immediately and in all the memory hierarchy.
2. When there is many consequent small and random write operations, then accumulating the dirty blocks in using write-back would not be efficient.

	Settings	Demand Fetch	Copies Back
write-allocate	bc: 128, cache: 16384, assoc: 4	66304	7200
	bc: 64, cache: 16384, assoc: 2	59856	6160
	bc: 64, cache: 8192, assoc: 4	110400	7168
	bc: 128, cache: 8192, assoc: 4	171136	14144
Write-no-allocate	bc: 128, cache: 16384, assoc: 4	66304	9668
	bc: 64, cache: 16384, assoc: 2	57008	8252
	bc: 64, cache: 8192, assoc: 4	107296	9219
	bc: 128, cache: 8192, assoc: 4	162240	13733

Answer to the Questions:

1. Write-allocate has lower memory traffic. Because it bring the entire block which helps to remove spatial locality and less cache misses that is responsible less memory traffic.
2. I think when there is large writes involve and cache size is small. Then brining the entire block in to the cache in write-allocate would not be helpful.