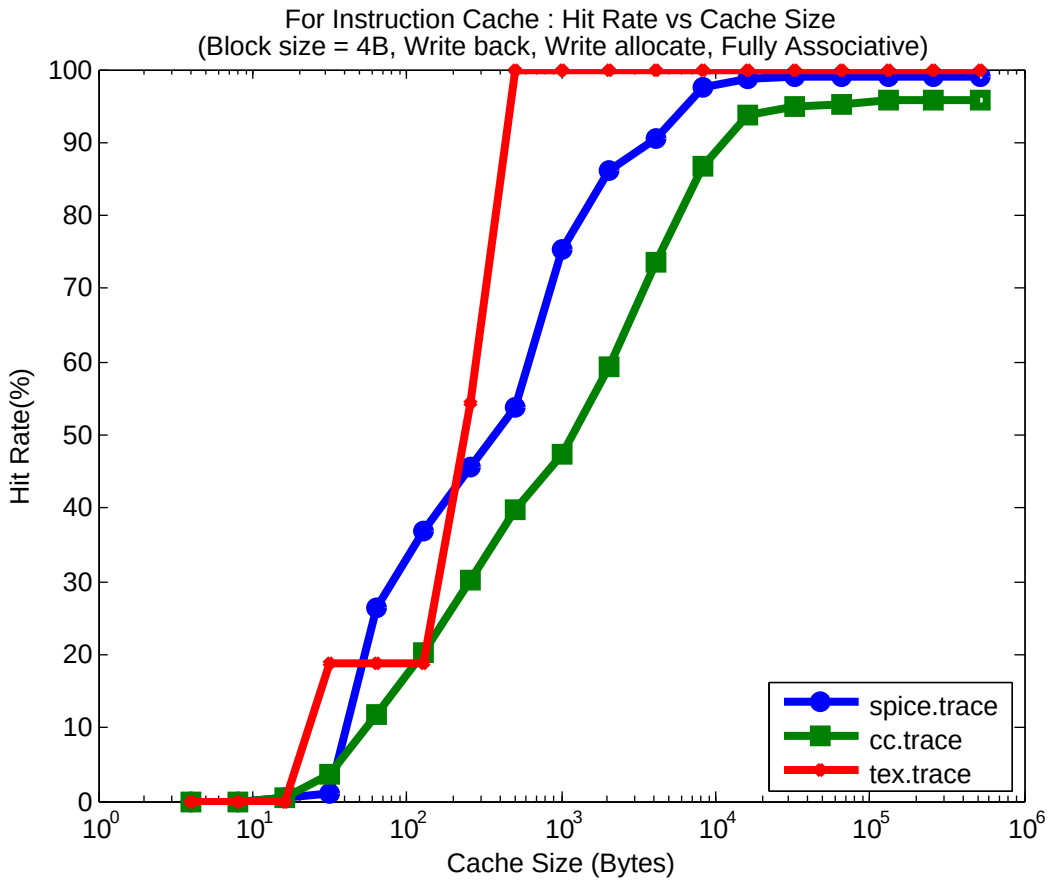
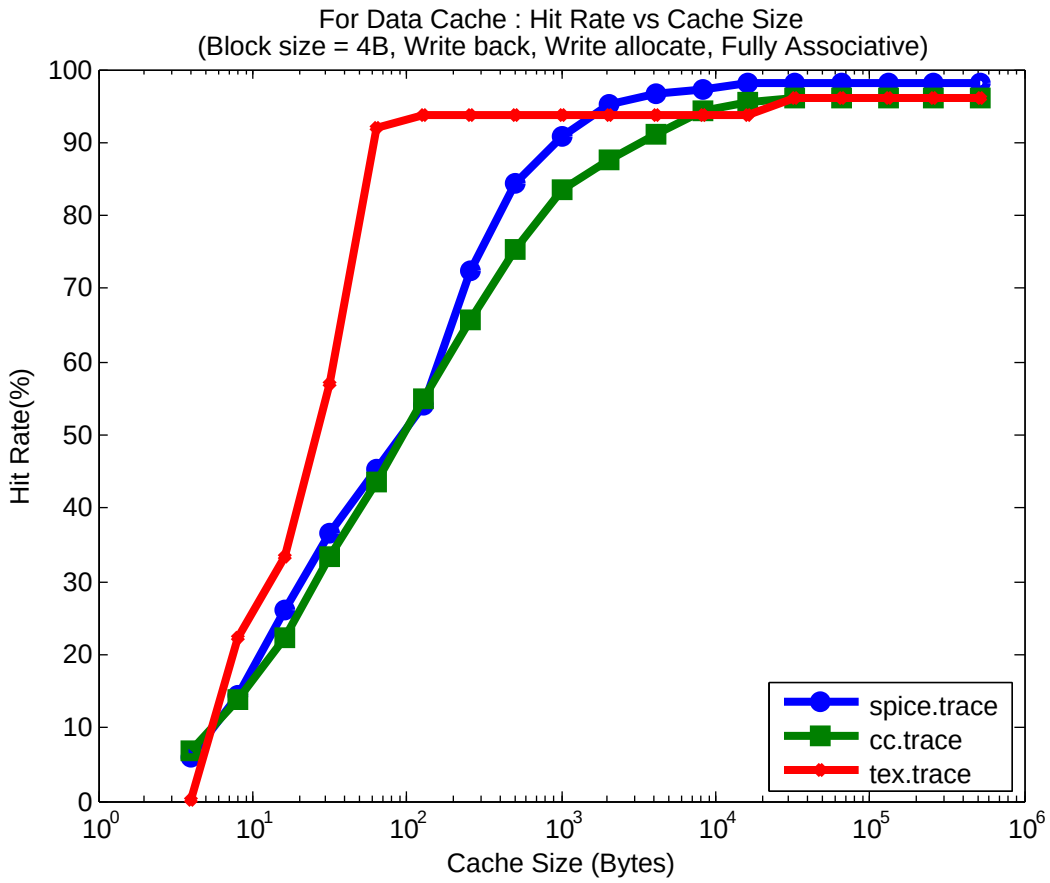
Problem 2.1

1. The experiment simulates a fully associative cache of varying size and tries to plot the hit rate vs cache size. Here, it is trying to see as to how the cache size (not its associativity) in itself impacts the misses and hits. I.e. it tries to compute capacity misses for a given cache size. Since the cache is always fully associative there are be no conflict misses. It also helps compute he working set for a given application. I.e. the size of cache beyond which the application would not have any benefit of a bigger cache.
2. The working Set from the Plot below is

	Spice.trace	cc.trace	Tex.trace
Instruction Cache	32KB-64KB	16KB-32KB	2KB-4KB
Data Cache	16KB-32KB	32KB-64KB	32KB-64KB

Working set is the size of the cache size for which there would be no non-compulsory misses. Only a range of the working set can be determined from the above experiment as he cache size is increased only in power of 2.


For Instruction Cache : Hit Rate vs Cache Size
(Block size = 4B, Write back, Write allocate, Fully Associative)
Hit Rate(%)
Cache Size (Bytes)
spice.trace
cc.trace
tex.trace

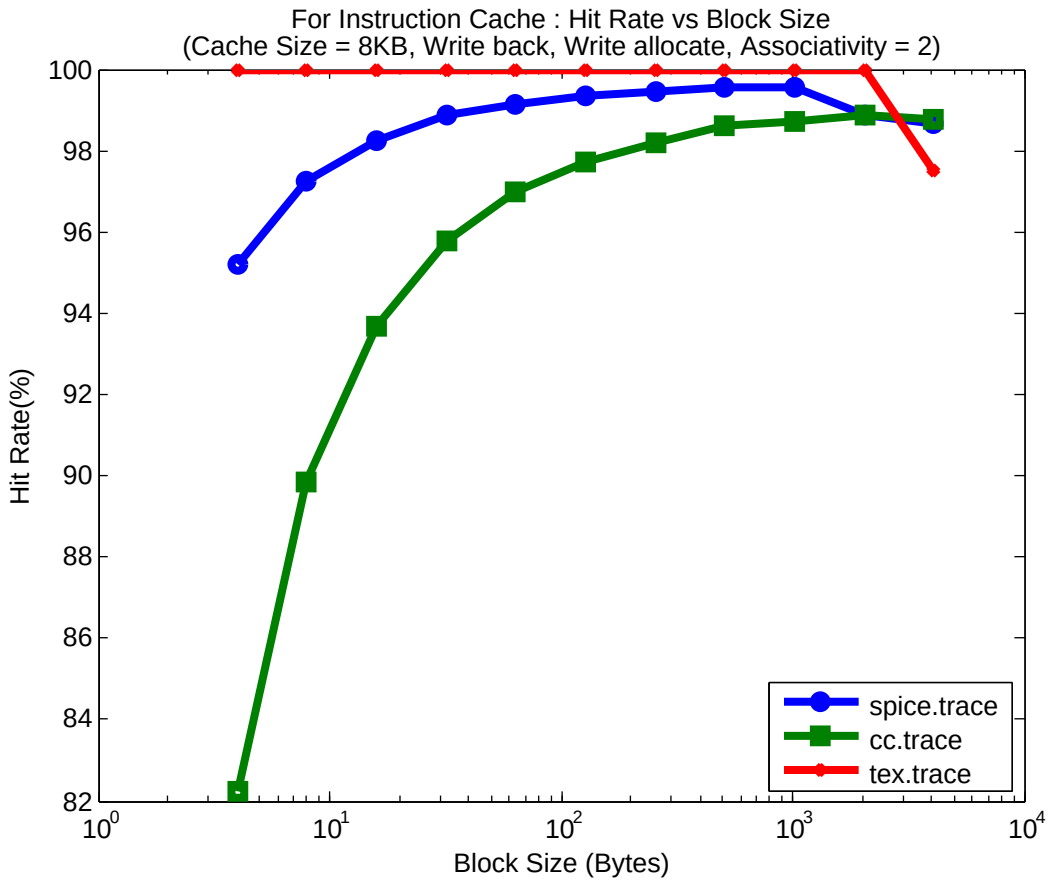

For Data Cache : Hit Rate vs Cache Size
(Block size = 4B, Write back, Write allocate, Fully Associative)
Hit Rate(%)
Cache Size (Bytes)
spice.trace
cc.trace
tex.trace

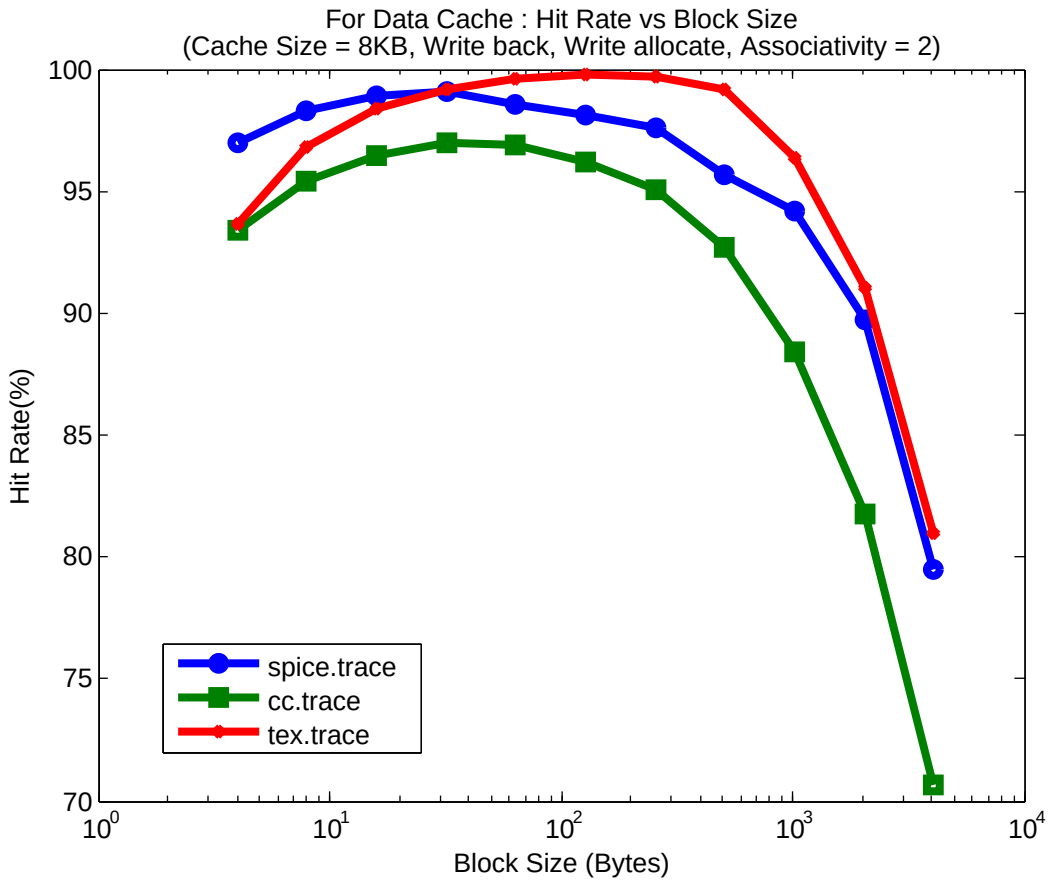
Problem 2.2

1. The hit rate vs block size plot as the shape of a oval as in figure because **initially with increase in block size the hit rate increases because of better utilization of spatial locality**. However, after a while the increase in block size reduces the number of blocks that can be kept in the cache and thereby causes eviction of useful blocks. **This detrimental effect dominates** and the hit rate decreases with increase in cache block size beyond a level. The peak of the curve gives a good indication of the pervasive spatial locality in the application.
2. **The optimal block size for each trace is as follows:**

	Spice.trace	cc.trace	Tex.trace
Instruction Cache	256KB	512KB	4B
Data Cache	32B	32B	128B

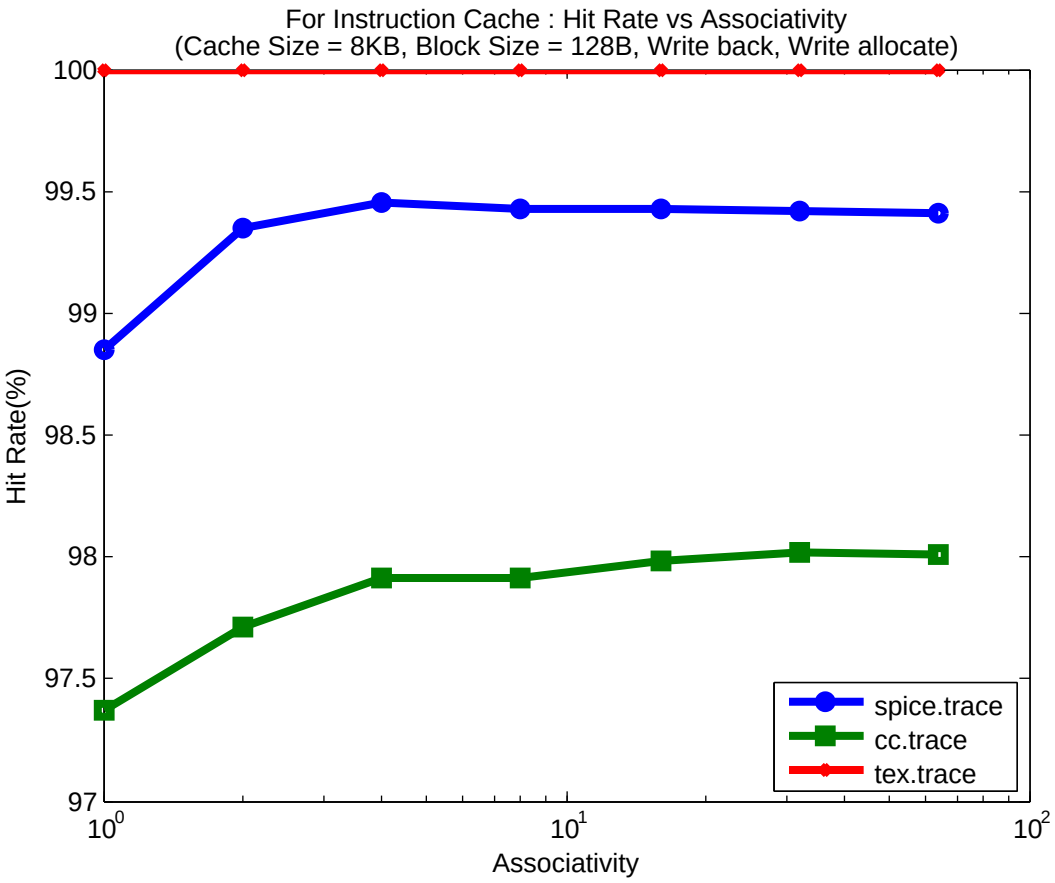
3. Yes the optimal block size for instruction and data references is different. From the plots, it shows that **instructions have much bigger spatial locality** and hence a larger cache block size is often beneficial for the instruction cache. Whereas, **data accesses exhibits much smaller spatial locality**.

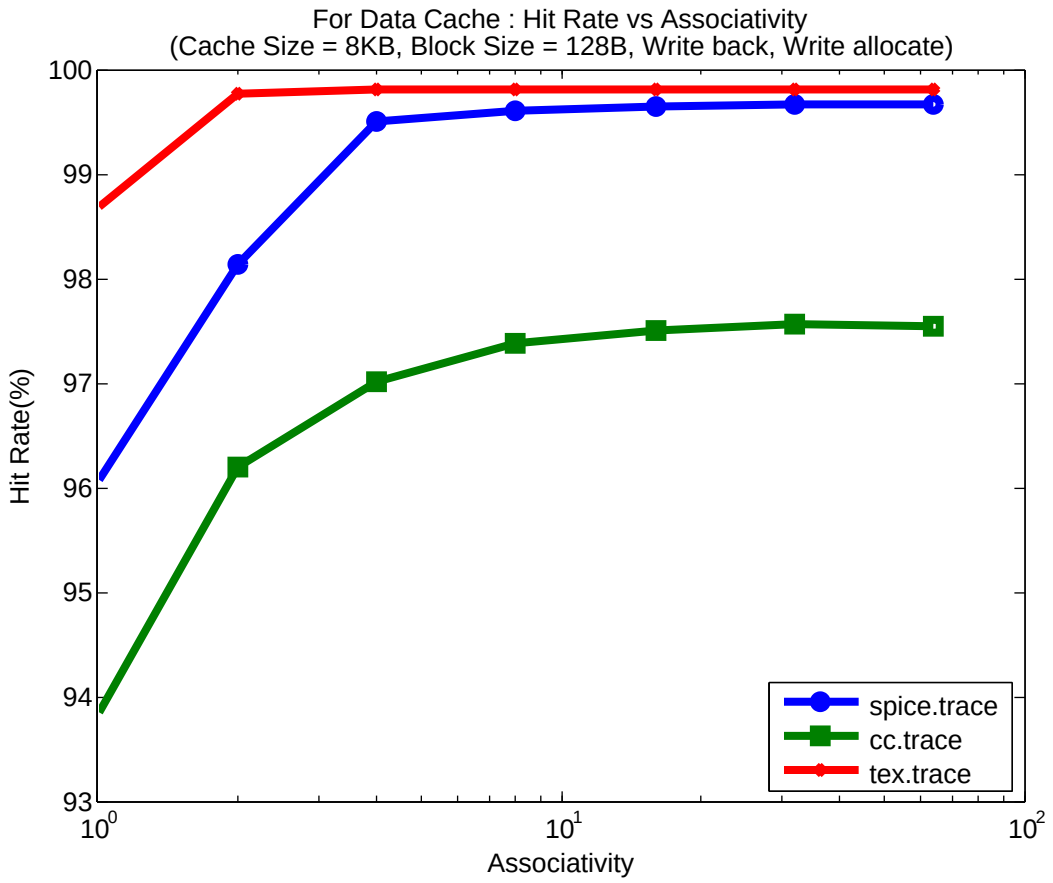

For Instruction Cache : Hit Rate vs Block Size
(Cache Size = 8KB, Write back, Write allocate, Associativity = 2)
Hit Rate(%)
100
98
96
94
92
90
88
86
84
82
10^0
10^1
10^2
10^3
10^4
Block Size (Bytes)
spice.trace
cc.trace
tex.trace


For Data Cache : Hit Rate vs Block Size
(Cache Size = 8KB, Write back, Write allocate, Associativity = 2)
Hit Rate(%)
Block Size (Bytes)
100
95
90
85
80
75
70
10^0
10^1
10^2
10^3
10^4
spice.trace
cc.trace
tex.trace

Problem 2.3

1. The hit rate vs associativity plot flattens beyond a point because there is really not much again of increasing associativity beyond that point as the conflicts misses more or less are taken care of with that level of associativity.
2. Yes there is a difference in the plots of hit rate vs associativity for instruction and data references. The plots hint that **higher associativity has a greater benefit on the data cache than on the instruction cache**.


For Instruction Cache : Hit Rate vs Associativity
(Cache Size = 8KB, Block Size = 128B, Write back, Write allocate)
Hit Rate(%)
100
99.5
99
98.5
98
97.5
97
10^0
10^1
10^2
Associativity
spice.trace
cc.trace
tex.trace


For Data Cache : Hit Rate vs Associativity
(Cache Size = 8KB, Block Size = 128B, Write back, Write allocate)
Hit Rate(%)
100
99
98
97
96
95
94
93
10^0
10^1
10^2
Associativity
spice.trace
cc.trace
tex.trace

Problem 2.4 (a)

1. In all the example traces, the write back caches have a smaller memory traffic than the write through caches.
2. Yes it is possible for a write through cache to have a smaller memory traffic. This is possible where there are large number of misses in the cache which may cause the write back to write entire blocks back to memory on each miss.

An Example on cc.trace

	Data Cache Size	Block Size	Associativity	Demand Fetches (in words)	Copies Back (in words)	Total Memory access (in words)
Write back	8192	64	2	474256	39426	513682
Write through	8192	64	2	474256	83030	557286
Write back	16384	64	4	205120	19588	224708
Write through	16384	64	4	205120	83030	288150

Problem 2.4(b)

1. In general both write allocate and write no-allocate caches may have the smaller memory traffic. There is no clear winner in the given traces. However, though write no-allocate might have a slightly smaller memory traffic it would along with have a huge number of cache misses, more or less, nullifying the very use of having a cache.
2. The write no allocate might have smaller memory traffic for the kind of applications where the writes are such that the written data is never used. In this case the good work done by write allocate cache would backfire and write no allocate cache might show better memory access numbers.

An Example on cc.trace:

	Data Cache Size	Block Size	Associativity	Demand Fetches (in words)	Copies Back (in words)	Total Memory access (in words)
Write allocate	8192	64	2	156000	18880	174880
Write no allocate	8192	64	2	151104	13624	164728
Write allocate	16384	64	2	59856	6208	66064
Write no allocate	16384	64	2	57008	8252	65260