1.

a. No. of blocks = 4; Associativity = 1; L1: 8, 8

	Block size (double words)	Hit Rate	Hits	Misses
Program 1	2	0.5	33	33
Program 2	2	0.01515	1	65
Program 1	4	0.7424	49	17
Program 2	4	0.0303	2	64
Program 1	8	0.8636	57	9
Program 2	8	0.04545	3	63

Program 1: The hit rate of program-1 increases with increase in the block size. It accesses data in a row-wisefashion, thus it has high spatial locality.
Program-2: The hit rate of program-2 doesn't change significantly with increase in block size. It accesses data in a column-wise fashion, thus it has no spatial locality.

b. Block size = 8 double-words; Associativity = 1; L1: 8, 8

	No. of blocks	Hit Rate	Hits	Misses
Program 1	8	0.8636	57	9
Program 2	8	0.8636	57	9
Program 1	16	0.8636	57	9
Program 2	16	0.8636	57	9
Program 1	32	0.8636	57	9
Program 2	32	0.8636	57	9

Program 1: The program accesses data from a 8 x 8 matrix, where each cell is 1 double-word. When the first element of a row is accessed, it is a miss, while the remaining 7 double-words are hits. As the number of blocks is greater than 8 there are no conflict misses among the rows. Thus, increasing the number of blocks any more than 8 does not change the hit rate.
Program 2: The reason for the constant hit rate is the same as above. While here the number of hits is the same as program-1, unlike program-1, the first 8 accesses are misses and the remaining are hits. (The extra miss the the compulsory miss)

c. No. of blocks = 4; Block size = 8 double-words; L1: 8, 8

	Associativity	Hit Rate	Hits	Misses
Program 1	1	0.8636	57	9
Program 2	1	0.04545	3	63
Program 1	2	0.8636	57	9
Program 2	2	0.8636	57	9

Program 1: The number of hits and misses for the first four rows of the matrix is straightforward, as the number of blocks is 4. For the 5th row, if the associativity is 1, then we get a conflict miss in the first block of the cache, but if the associativity is 2, we get a compulsory miss in the block which has the same index as the first block but is empty. So, either way we get a miss, thus, having the associativity 1 or 2 doesn't matter for program 1. We get no further misses in case of associativity 1 because the program doesn't access data from the evicted block, due to the row-wise data accessing of program 1.
Program 2: The hit increases significantly by increasing the associativity. In case of associativity 1, the hit rate is very low because while accessing the 2nd double-word in the first row, the first block in the cache would have already been evicted by the fifth row. Similarly, the subsequent blocks will keep evicting each other, resulting in the high miss rate. In case of 2-way associativity, there are effectively 8 blocks in the cache, thus we get a high hit rate.

d. No. of blocks = 4; Associativity = 1; L1: 16, 16

	Block size (double-words)	Hit Rate	Hits	Misses
Program 1	2	0.5	129	129
Program 2	2	0.003876	1	257
Program 1	4	0.7481	193	65
Program 2	4	0.007752	2	256
Program 1	8	0.8721	225	33
Program 2	8	0.007752	2	256

Program 1: The hit rate of program-1 increases with increase in the block size. It accesses data in a row-wisefashion, thus it has high spatial locality.
Program-2: The hit rate of program-2 doesn't change significantly with increase in block size. It accesses data in a column-wise fashion, thus it has no spatial locality.

e. Block size = 8 double-words; Associativity = 1; L1: 16, 16

	No. of blocks	Hit Rate	Hits	Misses
Program 1	8	0.8721	225	33
Program 2	8	0.007752	2	256
Program 1	16	0.8721	225	33
Program 2	16	0.01163	3	255
Program 1	32	0.8721	225	33
Program 2	32	0.8721	225	33

Program 1: In case of 8-block cache, the misses are mainly conflict misses. Increasing the number of blocks doesn't change the number of misses, as the misses are not eliminated but change from conflict to compulsory misses, when they occupy the new cache blocks. Thus, the hit rate doesn't change.
Program 2: The hit rate increases significantly when the number of blocks changes from 16 to 32 because now, the entire matrix can fit inside the cache. The 33 misses are almost all compulsory misses.

f. No. of blocks = 4; Block size = 8 double-words; L1: 16, 16

	Associativity	Hit Rate	Hits	Misses
Program 1	1	0.8721	225	33
Program 2	1	0.007752	2	256
Program 1	2	0.8721	225	33
Program 2	2	0.007752	2	256

Program 1: In case of 1-way associative cache, the misses are mainly conflict misses. Increasing the associativity doesn't change the number of misses, as the misses are not eliminated but change from conflict to compulsory misses, when they occupy the new cache blocks. Thus, the hit rate doesn't change.
Program 2: The hit-rate doesn't change for the same reason as above. But, increasing the associativity to 8 will significantly increase the hit-rate as the conflict misses are reduced.

2.

Program 1

	Write Hit	Write Miss	Number of write-backs
8, 8	Write-through	Write-allocate	64
		No write-allocate	64
	Write-back	Write-allocate	0
		No write-allocate	62
16, 16	Write-through	Write-allocate	256
		No write-allocate	256
	Write-back	Write-allocate	33
		No write-allocate	254

In case of **write-through**, the memory is updated whenever the cache is updated, regardless of the write-allocate or no write-allocate, which is seen in both (8, 8) and (16, 16).

For **write-back + write-allocate**, there are no evictions in (8, 8) so 0 write-backs, while in case of (16, 16) whenever a dirty block is evicted due to conflict, the updated data is written-back to the memory, which resulted in the 33 write-backs. For **write-back + no write-allocate**, the value is updated in the memory directly, which resulted in the 62/254 write-backs. This is not seen in case of write-back + write allocate, because the missed block is brought into the cache and then updated, which does not count as a write-back.

Program 2

	Write Hit	Write Miss	Number of write-backs
8, 8	Write-through	Write-allocate	64
		No write-allocate	64
	Write-back	Write-allocate	0
		No write-allocate	62
16, 16	Write-through	Write-allocate	256
		No write-allocate	256
	Write-back	Write-allocate	223
		No write-allocate	254

In case of **write-through**, the memory is updated whenever the cache is updated, regardless of the write-allocate or no write-allocate, which is seen in both (8, 8) and (16, 16).
For **write-back + write-allocate**, there are no evictions in (8, 8) so 0 write-backs, while in case of (16, 16) whenever a dirty block is evicted due to conflict, the updated data is written-back to the memory, which resulted in the 223 write-backs. For **write-back + no write-allocate**, the value is updated in the memory directly, which resulted in the 62/254 write-backs. This is not seen in case of write-back + write allocate, because the missed block is brought into the cache and then updated(not directly in the memory), which does not count as a write-back.

3.

		Hit Rate	Hits	Misses
8, 8	32-entry 4-word direct mapped	0.01538	2	128
	32-entry 4-word 2-way set associative	0.7385	96	34
	32-entry 4-word fully associative	0.7385	96	34
16, 16	32-entry 4-word direct mapped	0.06809	35	479
	32-entry 4-word 2-way set associative	0.7471	384	130
	32-entry 4-word fully associative	0.7471	384	130

(8, 8): The two arrays are of size 512 bytes each. The starting address of array2 is array1+1024, thus there is no overlap of addresses in this case. So, for direct mapped, the hit rate is very less, as array2[i] keeps on evicting array1[i]. In case of 2-way or fully associative, the previously mentioned eviction is reduced, thus, increasing the hit rate.
(16, 16): The two arrays are now of size 2048 bytes each. The last 1024 bytes of array1 and the first 1024 bytes of array2 overlap, which result in some hits, 35 specifically. For the same reason as in (8, 8) the hit rate increases by increasing the associativity.