10 Reverse Engineering Caches [40 points]

You are trying to reverse-engineer the characteristics of a cache in a system, so that you can design a more efficient, machine-specific implementation of an algorithm you are working on. To do so, you have come up with three sequences of memory accesses to various *bytes* in the system in an attempt to determine the following four cache characteristics:

- Cache block size (8, 16, 32, 64, or 128 B).
- Cache associativity (1-, 2-, 4-, or 8-way).
- Cache size (4 or 8 KB).
- Cache replacement policy (LRU or FIFO).

The only statistic that you can collect on this system is *cache hit rate* after performing each sequence of memory accesses. Here is what you observe:

Sequence	Addresses Accessed (Oldest → Youngest)								Hit Rate
1.	31	8192	63	16384	4096	8192	64	16384	3/8
2.	32768	0	129	1024	3072	8192			0
3.	0	4	8	4096	64	128			1

Assume that the cache is initially empty at the beginning of the first sequence, but *not* at the beginning of the second and third sequences. The sequences are executed back-to-back, i.e., no other accesses take place in between the three sequences. Thus, **at the beginning of the second (third) sequence, the contents are the same as at the end of the first (second) sequence**.

Based on what you observe, what are the following characteristics of the cache? Explain to get points. If a characteristic cannot be known, then write "Unknown" and explain.

(a) [10 points] Cache block size (8, 16, 32, 64, or 128 B)?

64 B.

Explanation:
Cache hit rate is 3/8 in sequence 1. This means that there are 3 hits. As two of them should be the second accesses to 8192 and 16384, the other hit is the access to 63. With a cache block of 64 B, the access to address 64 results in a miss.

(b) [10 points] Cache associativity (1-, 2-, 4-, or 8-way)?

4-way.

Explanation:
We already know that the cache block size is 64 B. Thus, there are 6 offset bits.

Regardless of cache size or associativity, addresses 0, 8192, 16384, and 32768 map to the same set. Thus, the cache cannot be 1-way, because we would not see hits on 8192 and 16384 in sequence 1.

If the cache were 2-way, 4096 would also map to the same set as 0, 8192, 16384, and 32768. This would make impossible a cache hit on 8192 in sequence 1.

If the cache were 8-way, 0, 1024, 3072, 4096, 8192, 16384, and 32768 would all map to set 0. With 8 ways, address 0 would not be replaced, so it would hit in sequence 2.

Therefore, the cache is 4-way associative.

(c) [10 points] Cache size (4 or 8 KB)?

8 KB.

Explanation:
We know that the cache is 4-way associative. In the beginning of sequence 2, 32768 replaces 0 (regardless of the replacement policy).

The fact that 8192 misses in sequence 2 can be explained by two possible cases:

1. If the replacement policy is FIFO, the access to 0 in sequence 2 replaces 8192. Thus, the cache size can be either 4 or 8 KB.
2. If the replacement policy is LRU, the access to 0 in sequence 2 replaces 4096. If the cache size is 4 KB, 1024 and 3072 map to the same set as 0 and 8192, and 1024 replaces 8192.

Since there is a hit on 4096 in sequence 3, the size should be 8 KB. Otherwise, 3072 would have replaced 4096.

(d) [10 points] Cache replacement policy (LRU or FIFO)?

FIFO.

Explanation:
As explained above, if the cache size is 8 KB, only FIFO can make address 0 replace address 8192 in sequence 2.