9 Caches (Reverse Engineering) [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 two 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 KiB).
- 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.	0	32	128	73	8192	255	16384	196	1/2
2.	127	4096	8192	32768	196	16384	0	512	3/8

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

128 B.

Explanation:
Cache hit rate is 1/2 in sequence 1. This means that there are 4 hits. Depending on the cache block size, we can group addresses that belong to the same cache block as follows:

- **8–32 B:** {0}, {32}, {128}, {73}, {8192}, {255}, {16384}, {196}. ∴ Number of possible hits = 0.
- **64 B:** {0, 32}, {128}, {73}, {8192}, {255, 196}, {16484}. ∴ Number of possible hits = 2.
- **128 B:** {0, 32, 73}, {128, 255, 196}, {8192}, {16384}. ∴ Number of possible hits = 4.

Therefore, we can know that the cache block size is 128 B.

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

4-way.

Explanation:

Cache hit rate is 3/8 in sequence 2, which means that there are 3 hits.
We already know that the cache block size is 128 B. Thus, there are 7 offset bits.

The access to address 196 in sequence 2 would hit because the cache block would not be replaced.
The access to address 512 in sequence 2 would miss because address 512 does not belong to any cache block previously accessed.
Therefore, the accesses to addresses 0, 127, 4096, 8192, 16834 and 32768 in sequence 2 would hit 2 times.
Regardless of cache size, those addresses will never hit when the cache were 1-way or 2-way.
If the cache were 8-way, those addresses would all map to set 0. With 8 ways, addresses 127, 9182, 16384 would not be replaced, so the three addressess would hit.
Therefore, the cache is 4-way associative.

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

LRU.

Explanation:
From questions (a) and (b), we already know the following facts:

- The cache block size is 128 B.
- The cache is 4-way.
- The accesses to addresses 0, 127, 4096, 8192, 16384, and 32768 in sequence 2 would hit 2 times.

Regardless of the cache size, 0, 127, 4096, 8192, 16834, and 32768 in sequence 2 would all map to set 0.
With the FIFO policy, accesses to addresses 127, 8192, and 16484 in sequence 2 would hit.
With the LRU policy, accesses to addresses 127 and 8192 would hit.
Therefore, the cache adopts the LRU policy.

(d) [10 points] To identify the cache size, you execute the following sequence right after sequence 2 (i.e., the contents are the same as at the end of the second sequence) and measure the cache hit rate:

Addresses Accessed (Oldest → Youngest): 8192 → X → Y

Which addresses should you use for X and Y?

$X = 1024 \times (2k - 1)$ where k is a positive integer.
$Y = 32768$

Explanation:
If the cache is 4-KiB, all addresses that are multiples of 1024 would map to set 0. If the cache is 8-KiB, all addresses that are multiples of 2048 would map to set 0.
After the access to 8192 in sequence 3, the LRU address in set 0 is 32768.
If the cache is 4-KiB, access to $X = 1024 \times (2k - 1)$ would replace 32768, so access to 32768 would miss. If the cache is 8-KiB, such access would not replace 32768, so access to 32768 would hit.