After writing a script to brute force every possible combination of C, S, and B parameters, I got the same combination of optimal parameters regardless of which test trace was being used, such that C = 8, S = 8, and B = 0. These parameters indicate that the cache being used has 2^8 (256) bytes for data, 2^8 (256) blocks per set, and 2^0 (1) byte per block. The resulting average access times for each test trace area as follows:

1. astar: 2.000199
2. bzip2: 2.000184
3. mcf: 2.000197
4. perlbench: 2.000197

I believe these parameters are optimal regardless of the workload, primarily because this is a hypothetical cache that does not have the drawbacks associated with physical caches in the real world. For example, in our simulation, there is no access time penalty associated with having a larger cache that a real-world cache would have. As a result, having a bigger cache will always result in a lower average access time in our simulation. Additionally, with S = C – B (8), the cache becomes fully associative, which when combined with a block size 2^0 = 1 byte, and very fast to search. Normally, a fully associative cache would need to search through every single block once it indexes into the correct cache line. However, since there's only one byte in each cache line, and consequently only one byte to search through, the cache will always find what it is looking for.