Understanding Software Dynamics Ch 3 Lab Report

Notes:
1. The cache line size is the smallest unit of data transferred between the cache and main memory.
When the stride is smaller than the cache line size, multiple accesses may hit the same cache line, resulting in lower access times. When the stride matches or exceeds the cache line size, each access requires fetching a new cache line, increasing the access time

2. L2 access time: 10-20 cycles
main memory: 100-300 cycles

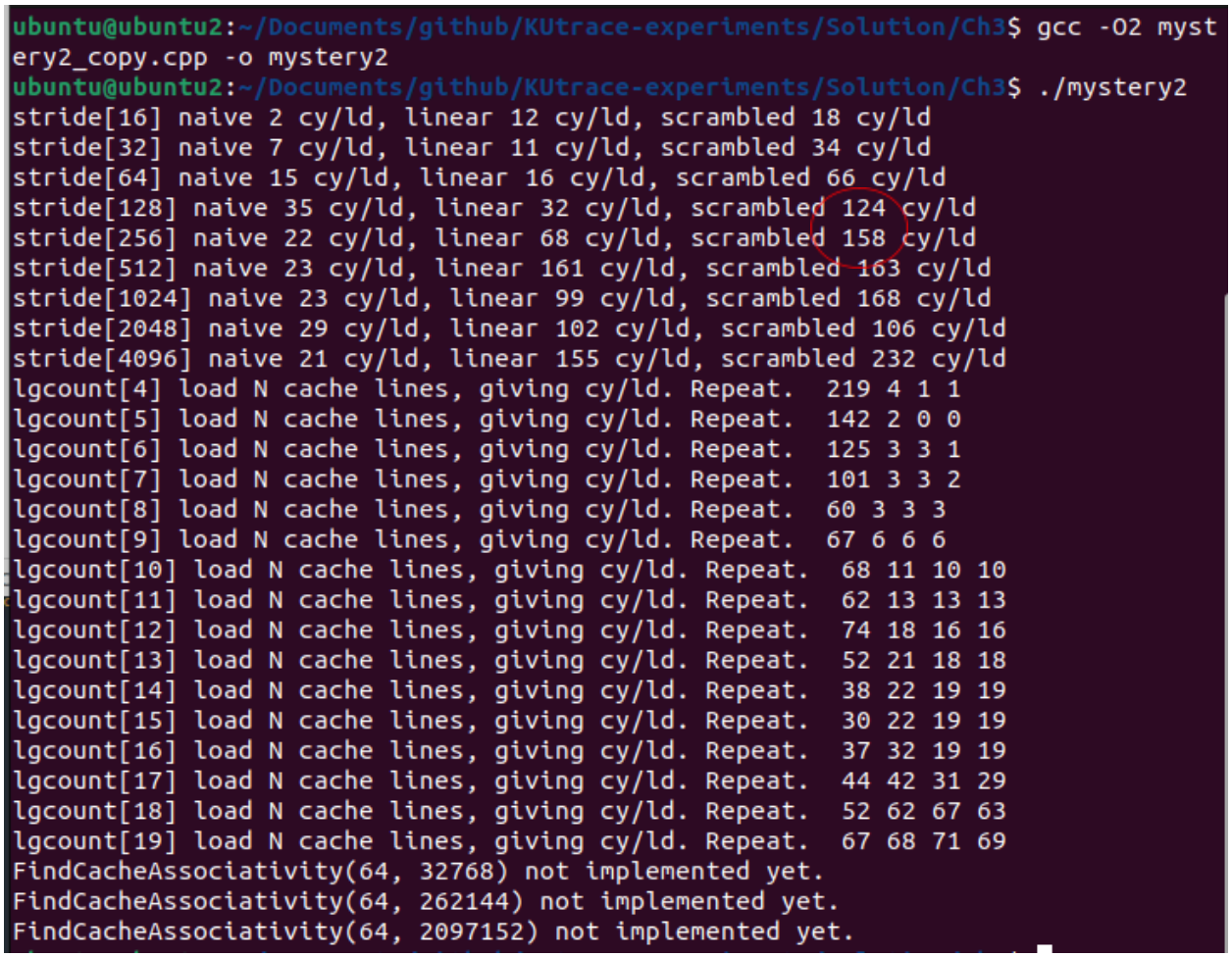
3.0) Run mystery2.cc

```
ubuntu@ubuntu2:~/Documents/github/KUtrace-experiments$ ./mystery2
stride[16] naive 2 cy/ld, linear 7 cy/ld, scrambled 17 cy/ld
stride[32] naive 8 cy/ld, linear 9 cy/ld, scrambled 38 cy/ld
stride[64] naive 13 cy/ld, linear 16 cy/ld, scrambled 77 cy/ld
stride[128] naive 19 cy/ld, linear 31 cy/ld, scrambled 158 cy/ld
stride[256] naive 22 cy/ld, linear 70 cy/ld, scrambled 188 cy/ld
stride[512] naive 23 cy/ld, linear 137 cy/ld, scrambled 185 cy/ld
stride[1024] naive 20 cy/ld, linear 148 cy/ld, scrambled 169 cy/ld
stride[2048] naive 19 cy/ld, linear 179 cy/ld, scrambled 208 cy/ld
stride[4096] naive 23 cy/ld, linear 120 cy/ld, scrambled 157 cy/ld
lgcount[4] load N cache lines, giving cy/ld. Repeat.  198 4 1 1
lgcount[5] load N cache lines, giving cy/ld. Repeat.  161 2 0 0
lgcount[6] load N cache lines, giving cy/ld. Repeat.  143 3 3 1
lgcount[7] load N cache lines, giving cy/ld. Repeat.  110 3 3 3
lgcount[8] load N cache lines, giving cy/ld. Repeat.  103 3 3 3
lgcount[9] load N cache lines, giving cy/ld. Repeat.  83 7 7 7
lgcount[10] load N cache lines, giving cy/ld. Repeat.  85 10 10 10
lgcount[11] load N cache lines, giving cy/ld. Repeat.  69 10 10 10
lgcount[12] load N cache lines, giving cy/ld. Repeat.  59 15 14 14
lgcount[13] load N cache lines, giving cy/ld. Repeat.  38 19 18 18
lgcount[14] load N cache lines, giving cy/ld. Repeat.  32 21 18 18
lgcount[15] load N cache lines, giving cy/ld. Repeat.  31 22 18 18
lgcount[16] load N cache lines, giving cy/ld. Repeat.  40 34 19 19
lgcount[17] load N cache lines, giving cy/ld. Repeat.  50 71 62 53
lgcount[18] load N cache lines, giving cy/ld. Repeat.  66 74 71 73
lgcount[19] load N cache lines, giving cy/ld. Repeat.  83 77 73 71
FindCacheAssociativity(64, 32768) not implemented yet.
FindCacheAssociativity(64, 262144) not implemented yet.
FindCacheAssociativity(64, 2097152) not implemented yet.
```

3.1) 64 bytes. The scrambled time jumps double from stride 64 to 128. It means that cache misses are more frequent as the stride spans multiple cache lines.

3.2) For stride 256, (256 bytes = 4 cache lines)
naive (22 cy/ld): CPU optimizing by issuing multiple loads in parallel or prefetching. So it masks the true cache miss penalty.
linear (70 cy/ld): a mix of L1 hits and L2 misses
scrambled (188 cy/ld): DRAM access with slightly buffering/slightly cache hits

3.3) mystery2_copy.cpp
The scrambled time at stride 128 improves from 158 to 124. Now, 2 successive accesses can be in the same row. The CPU and DRAM hardware implement a shortcut to just maintain column access.



3.4) (FindCacheSizes is accessing 2^4 up to 2^16 cache lines)
L1: 32KB (2^9 = 512 lines), because lgcount[10] and lgcount[11]'s cycles are both 10, indicating it starts to use L2 cache
L2: 256KB (2^12)
L3: 3MB (2^16), because its cycle increases to 34.

3.5) By taking averages
L1: 3-4 cycles
L2: 14-15 cycles
L3: 50-60 cycles

3.6) Change code line 283

```
int base = 3;
for (int exp = 1; i < exp; i++) {
        int count = 1;
        for (int i = 0; i < exp; i++) { // don't use pow()
                count *= base;
        }

        // same
}
```

3.7) lgcount[8] to [9] drop is about cache eviction. [10] to [11] drop is about the transition to the next cache level and LRU.
[8] (103 cy/ld): conflict misses where multiple lines compete for the same cache set
[9] (83 cy/ld): exactly fills 32KB L1 cache. fully utilized
[10] (85 cy/ld): starts filling the 256KB L2 cache. more L1 misses, but L2 cache's larger size and higher associativity reduce cache misses, so no observable changes
[11] (69 cy/ld): by LRU. It handles the overflow from L1 more efficiently when the load is distributed across L2

3.8) mystery2_find_cache_associativity.cpp
L1: 9-ways
L2: 11-ways/13-ways
L3: 10-ways
(shown in next page)

```
Finding associativity for cache size 32768 bytes, line size 64 bytes
A= 1: 2 cy/ld
A= 2: 2 cy/ld
A= 3: 2 cy/ld
A= 4: 3 cy/ld
A= 5: 3 cy/ld
A= 6: 2 cy/ld
A= 7: 2 cy/ld
A= 8: 2 cy/ld
A= 9: 2 cy/ld
A=10: 7 cy/ld
A=11: 4 cy/ld
A=12: 4 cy/ld
A=13: 3 cy/ld
A=14: 3 cy/ld
A=15: 3 cy/ld
A=16: 4 cy/ld
Finding associativity for cache size 262144 bytes, line size 64 bytes
A= 1: 2 cy/ld
A= 2: 2 cy/ld
A= 3: 2 cy/ld
A= 4: 2 cy/ld
A= 5: 3 cy/ld
A= 6: 3 cy/ld
A= 7: 3 cy/ld
A= 8: 3 cy/ld
A= 9: 3 cy/ld
A=10: 3 cy/ld
A=11: 5 cy/ld
A=12: 8 cy/ld
A=13: 3 cy/ld
A=14: 9 cy/ld
A=15: 5 cy/ld
A=16: 4 cy/ld
Finding associativity for cache size 2097152 bytes, line size 64 bytes
A= 1: 4 cy/ld
A= 2: 4 cy/ld
A= 3: 3 cy/ld
A= 4: 3 cy/ld
A= 5: 3 cy/ld
A= 6: 8 cy/ld
A= 7: 3 cy/ld
A= 8: 3 cy/ld
A= 9: 3 cy/ld
A=10: 4 cy/ld
A=11: 8 cy/ld
A=12: 5 cy/ld
A=13: 4 cy/ld
A=14: 4 cy/ld
A=15: 3 cy/ld
A=16: 3 cy/ld
```