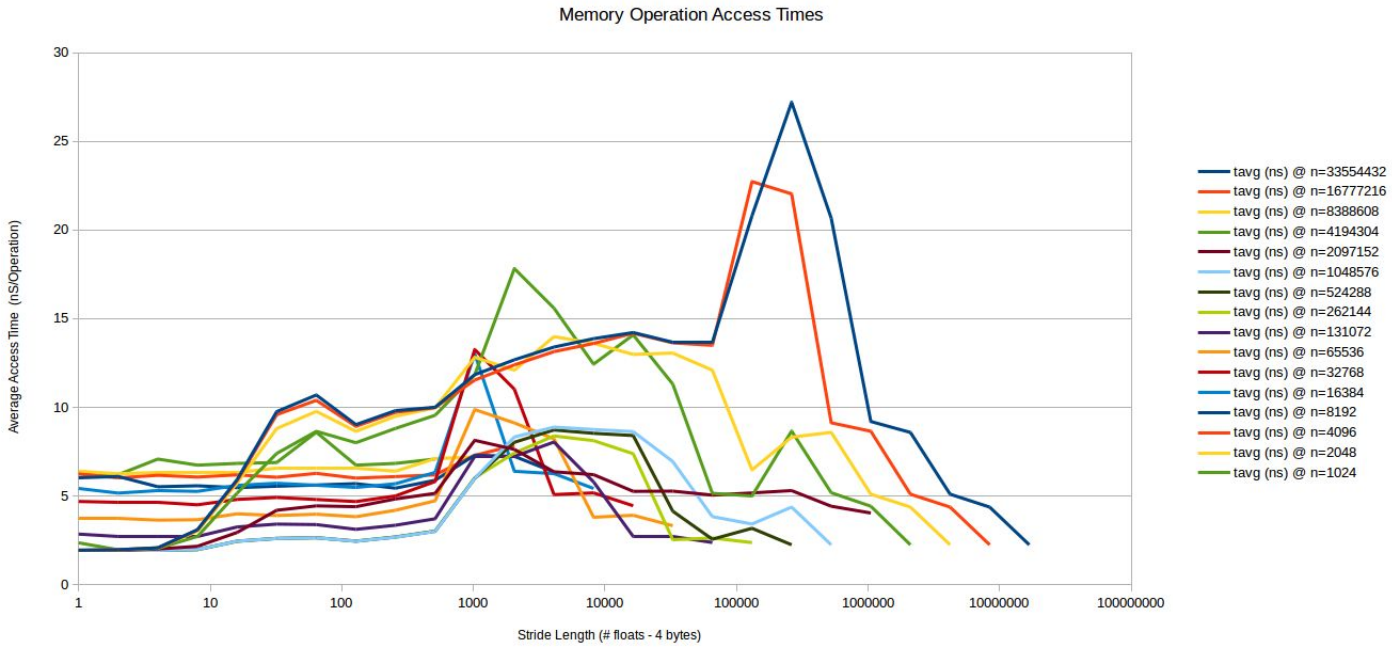
Benchmark #2

D. Aaron Wisner (daw268)
ECE 5720

Note: All tests were conducted on a desktop running an i7-3770k @ 4.8 GHz with Linux kernel version 4.4.0

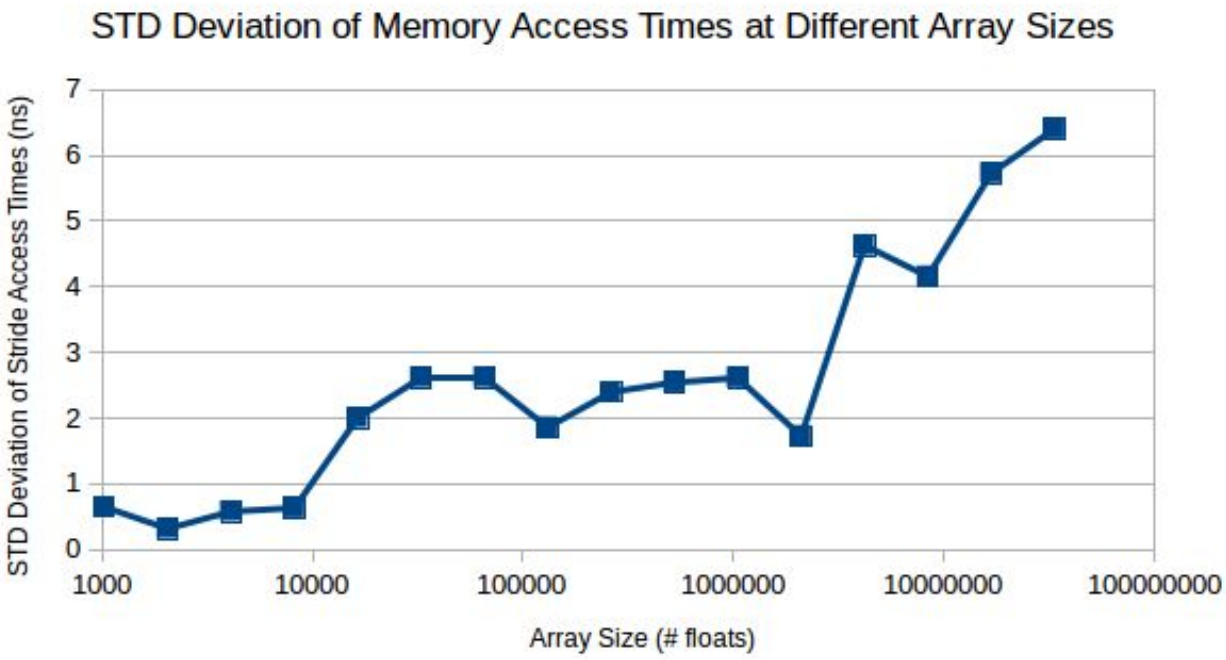
Results



Note: Full images of all graphs are available in the enclosing folder as well as the raw data

Questions:

1. The size of the L1 cache was determined by finding the largest size n, which consistently has the same access time across all step sizes. This was accomplished by finding the standard deviation of all the stride access times at each array size, then plotted on a graph, seen below.



Notice how the standard deviation of memory access times remains under 1ns for all array sizes at and under 8192 floats. This can also be seen by the relatively flat lines across all stride lengths for the n=8192 and under plots in the memory access time graph. At n > 8192, the standard deviation suddenly jumps to several times more. Again, this can also be seen in the memory access time graph by wild variance in access times across different stride lengths for cases where n > 8192. Ultimately, this suggests that 8192 is the maximum number of floats that can be kept in the entire L1 cache. Once the cache has warmed up, there will never be an L1 cache miss and thus changes in stride length will have little effect on the access times. Since, there are 4 bytes in a float, the L1 cache size is 4*8192 = 32KB

2. This was estimated by first ignoring the tests where n was small, as this will guarantee that the entire array of size 4*n bytes cannot fit into the entire L1 cache. Assuming a LRU replacement policy, this will hopefully trigger an L1

cache miss every time an element that resides in a different cache line than the previous accessed element is accessed. Consider the n=2^23 through n = 2^26 cases. Notice the sudden increase in average access times between the stride length of 8 floating going to 16 floats, but no significant increase for step sizes large than 16 floats. This suggests each cache line can hold 16 floats, or 64 bytes. When the stride length is at least 16 floats (64 bytes) every element will reside in a different cache line than the previous element, and with a large enough n, will generate a L1 miss for every single element access. This cache miss on every element access is what causes the increase in access times for STEP_SIZE >= 16 floats. Thus, the cache line size is 64 bytes.

3. This could not be determined, although it was was looked up to be 8-way set associative. A better understanding of CPU's mapping of addresses to L1 cache sets, virtual memory implementation, and TLB would be needed.

Neither method used to accurately calculate the L1 cache size yielded any results when attempted to be used to calculate L2 cache specs. However, the standard deviation of memory access times across different array sizes seemed to yield accurate results for the L3 cache. Notice the dramatic jump in standard deviation in memory access times going from N=2^21 to N > 2^21, with roughly double the standard deviation. This suggests that the max number of floats that can be held in cache is 2^21, or 8MB, which was confirmed as the size of the L3 cache by looking up actual the value. The ability to notice this jump in the jitter of access times likely stems from the massive difference in latency between accessing an element in L3 cache vs comparatively slow DDR3 memory.