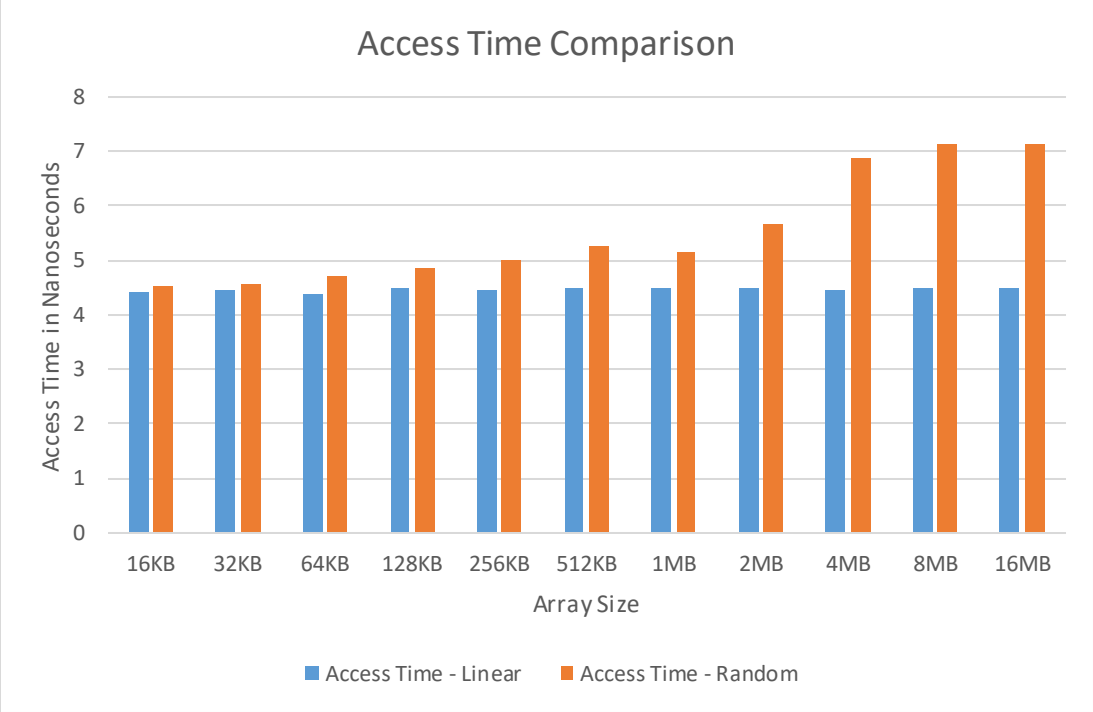
Cache Testing

Processor: Intel Core M-5Y10c

Cache Sizes: 4MB L3 cache, 256KB L2 cache, 64KB L1 cache

N	Size of a	Access Time – Linear(ns)	Access Time – Random(ns)
4096	16KB	4.42	4.54
9192	32KB	4.44	4.57
18384	64KB	4.37	4.72
36768	128KB	4.48	4.85
73536	256KB	4.46	5
147072	512KB	4.48	5.27
294144	1MB	4.51	5.17
588288	2MB	4.49	5.67
1176576	4MB	4.46	6.89
2353152	8MB	4.49	7.13
4706304	16MB	4.49	7.13



Differences

- It can be observed that the linear access time is quite consistent for all array sizes.
- It can be observed that the random access time worsens noticeably when it reaches the size of the L3 cache 4MB.
- It can also be observed that as array size increases the random access time steadily increases.

Matrix Testing

Straightforward Multiplication runtime : 18.98 seconds

The problem in this access pattern is that for the second matrix B, we are accessing the first element of a row, then skipping to the first element of the next row accessing elements downward which is very inefficient for an array which is stored in row major. This is because a new cache line must be loaded for each access and the prefetching of further values in the row may not be used before they are overwritten. [If column major storage of arrays was used, matrix A would have a similar problem]

Use of a temporary matrix: 5.51 seconds

Avoid the access downwards through the array by storing the values along the rows rather down the columns through use of transpose.

Use of Blocking: 11.71 seconds

Approach: Use of block size 8x8 as a double is of length 8 bytes, the cache lines are of length 64 bytes for my processor, therefore ideally 8 doubles may be placed in a cache line and a block with rows of length 8 is the fastest approach. Additionally through testing of other block sizes this was the fastest option with 7x7 being a close second.