ASSIGNMENT 1

Name: Deepak Nolastname
ASU ID: 1211301777

Task 1: Memory hierarchy performance measurement

PLOTS:

For the measurement of time for memory access I used two different methods: measuring time using system clock (clock_gettime) and using the perf tools. The results are presented below. The results presented below are obtained using perf tool. I calculated the number of cycles for each of my different array sizes and then calculated the access time for my system frequency (2.5 GHz).

NOTE: The very large array sizes have not been included in the plots as the running time for those was very high and could not be completed.

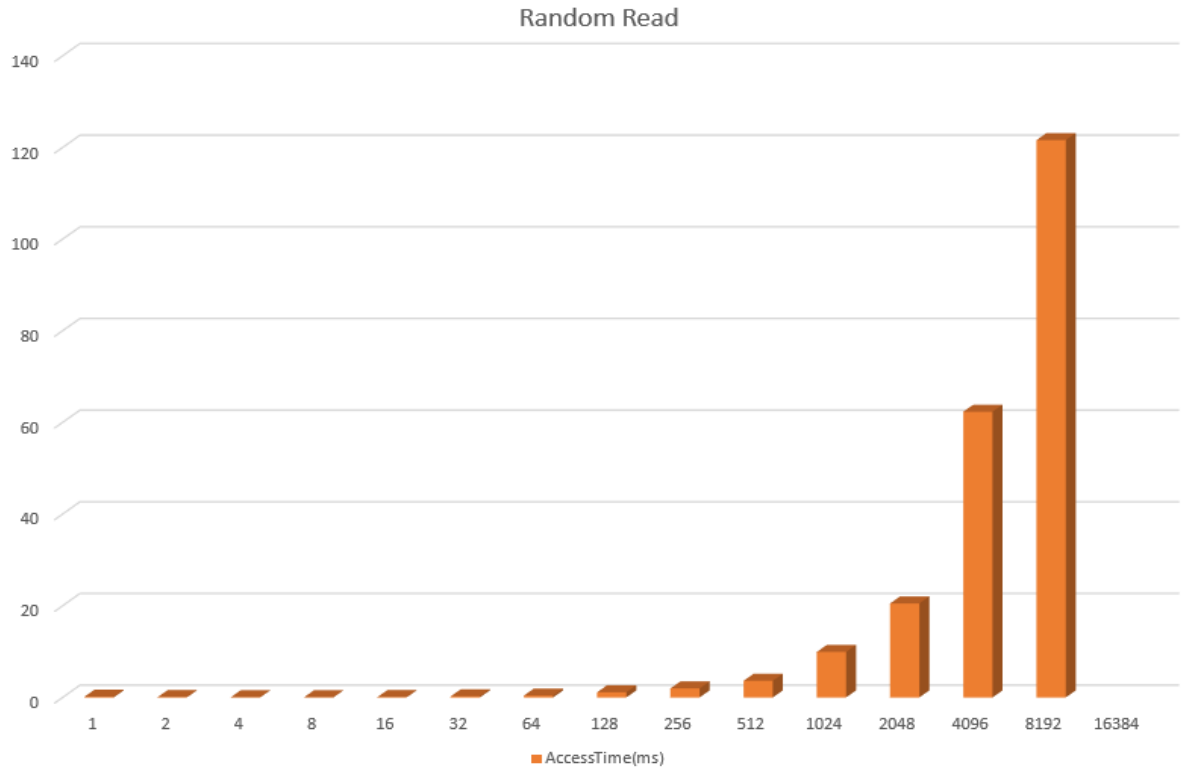

Fig 1: Memory Mountain for Random Read

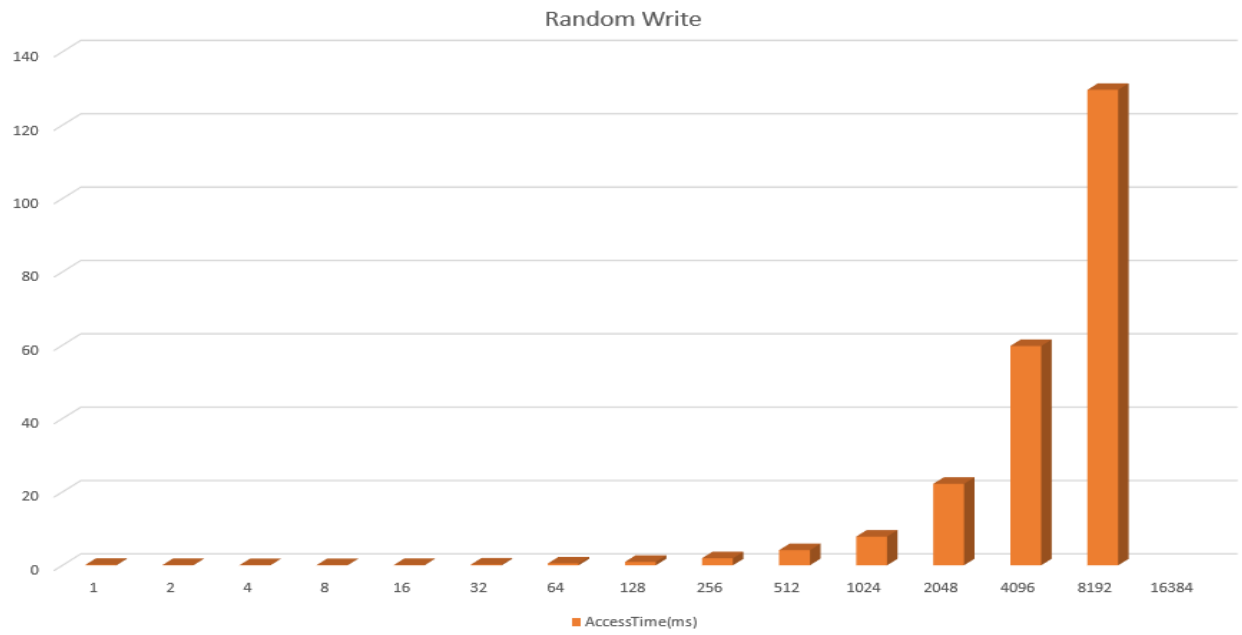

Fig 2: Memory Mountain for Random Write

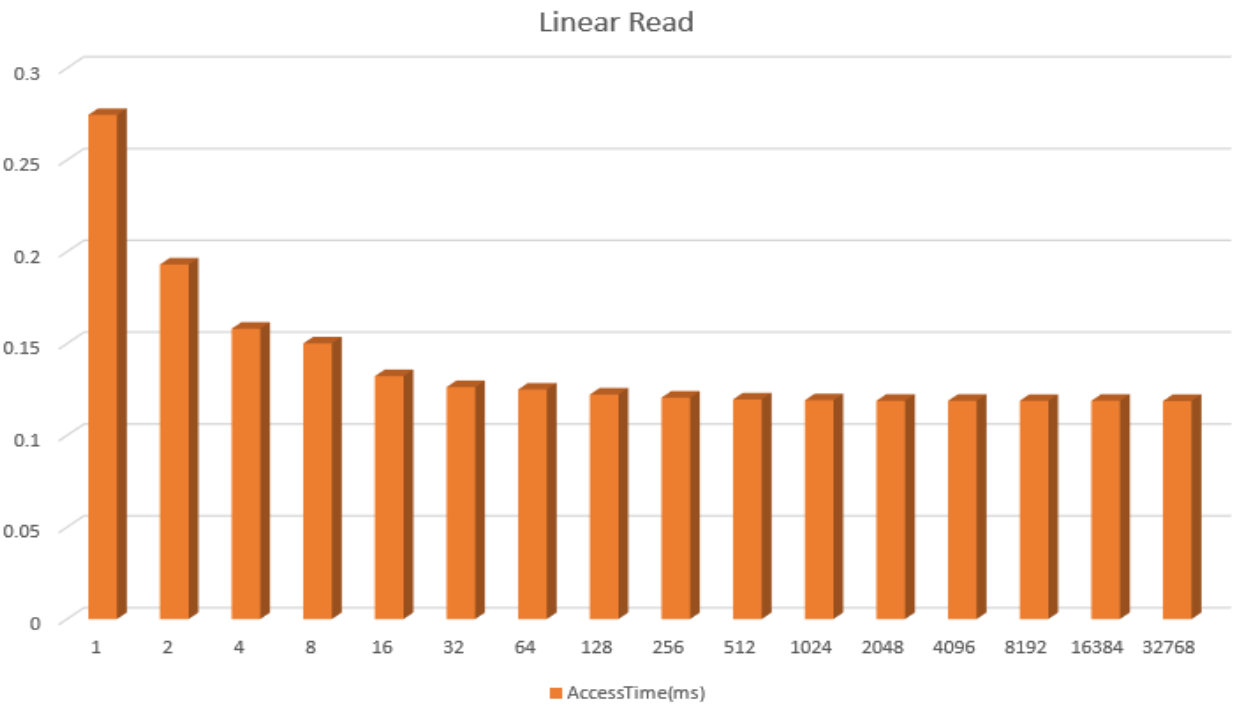

Fig 3: Memory Mountain for Linear Read

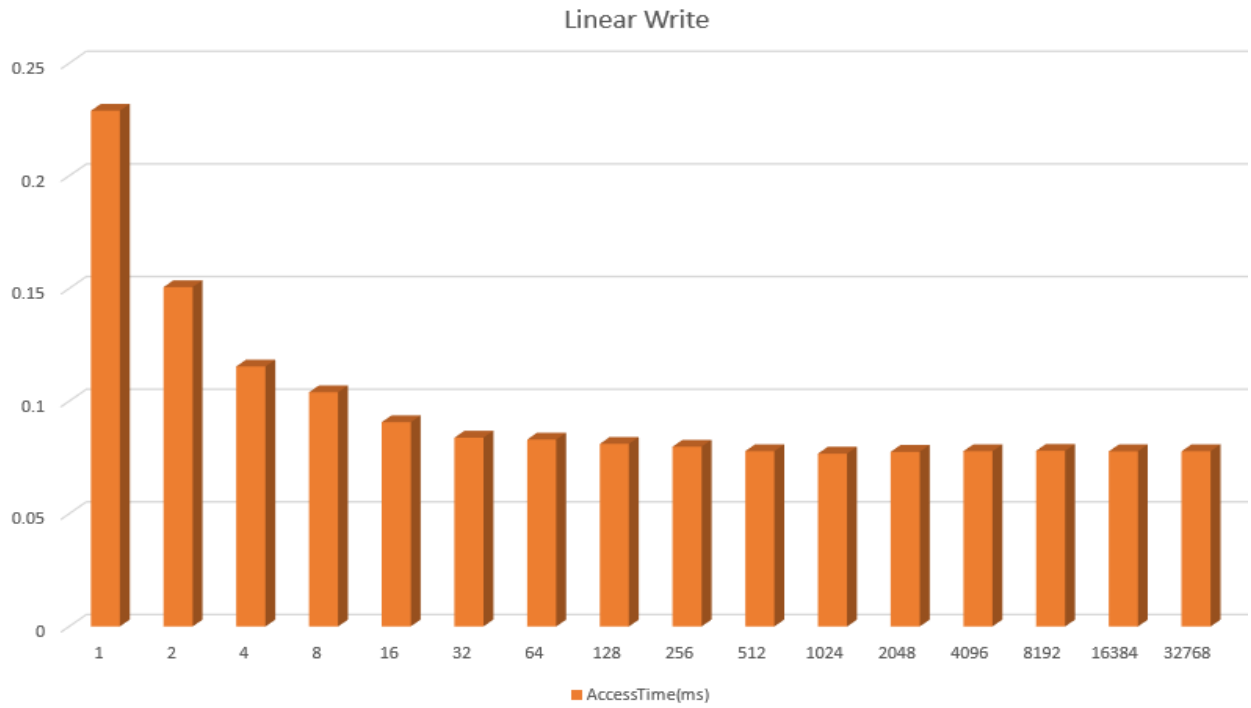

Fig 4: Memory Mountain for Linear Write

Observations:

It is observed that the average time per reference for random access keeps on increasing as we increase the array size. This is because as we increase the array size, it is firstly available in the L1 cache and has very low access time. But as we increase the array size, a part of it is moved to L2, L3 cache and main memory, respectively. As the access time of these higher level of memories is high, we get increase with array size.

For the case of the linear access, it is theoretically expected that the access time follows the same trend as in the case of the random access. But it was observed that the access time is almost constant for different array sizes. The reason for this can be the complier prefetching and optimization. I tried to avoid the complier optimization using the "o0" flag during the build but it does not make any difference. Also, I tried disabling the individual flags (one by one) for gcc optimization but the results remain unaltered.

Questions:

1). Memory mountain plot is proportional to the latency of the memory hierarchy. This is because as we increase the array size array, the access to different cache level is available. The access time of L2 is greater than L1 and L3is greater than L2 and so on. So as the data is made available to L2 and L3 cache the access time is supposed to increase which is seen in the plots for random read and write. The calculation presented here are averaged over a large number of iterations to give an ensemble access time. Due to the average, values are more stable and accurate. Also, the measurements have been verified using the "perf" tool output. The calculation of time are done using the system clock and only the operation time is observed to avoid any extra overhead timings.

2). The Access time for Linear and random is different. This is because when we do a linear access consecutive memory location are accessed. So there will not be much cache miss in case of linear access till the time the cache limit is exceeded. But in the case of random access, memory locations are not consecutive and the random access can cause more number of cache misses. Due to this large number of cache miss we will get more time for random access. The difference for read and write is not very visible in our observations of time (the difference is observed in the number of the clock cycles with write taking more number of system clocks). Also, theoretically the write should take a bit more time than write (considering the overhead of writing).

3). Immediate effect of the virtual memory has not been included in my observations.

Task 2: Reproduce the task 1 using Simple Scalar simulator

PLOTS:

Output of simplescalar for codes used in the task 1 is presented as follows:

The average time per reference for random access keeps on increasing as we increase the array size

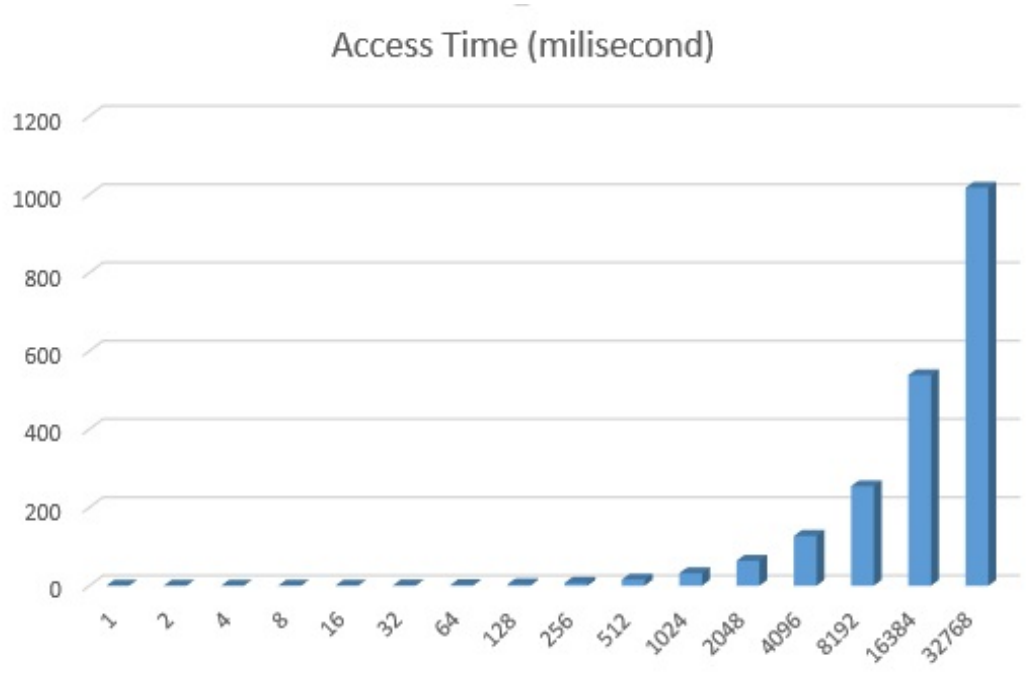

Fig 5: SimpleScalar output for Linear Read

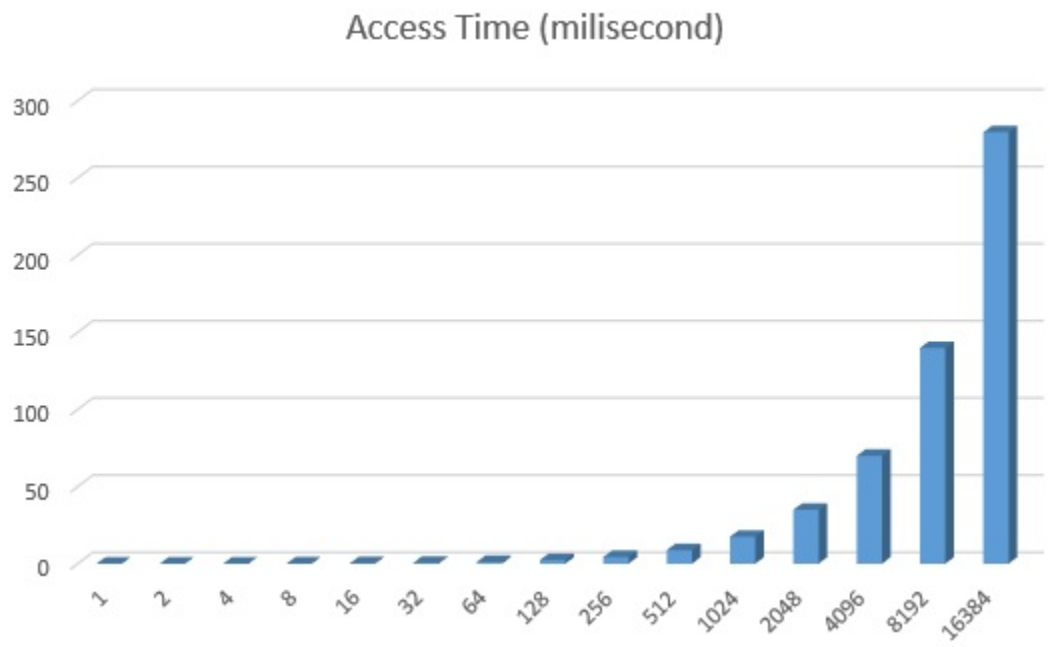

Fig 6: SimpleScalar output for Linear Write

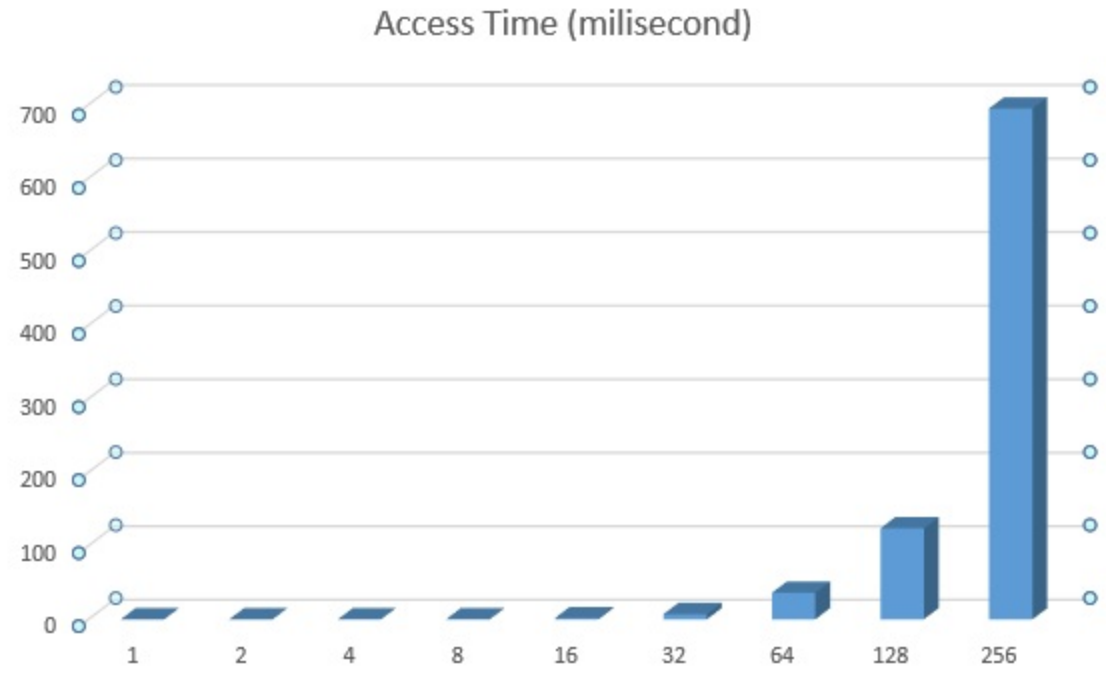

Fig 7: SimpleScalar output for Random Read

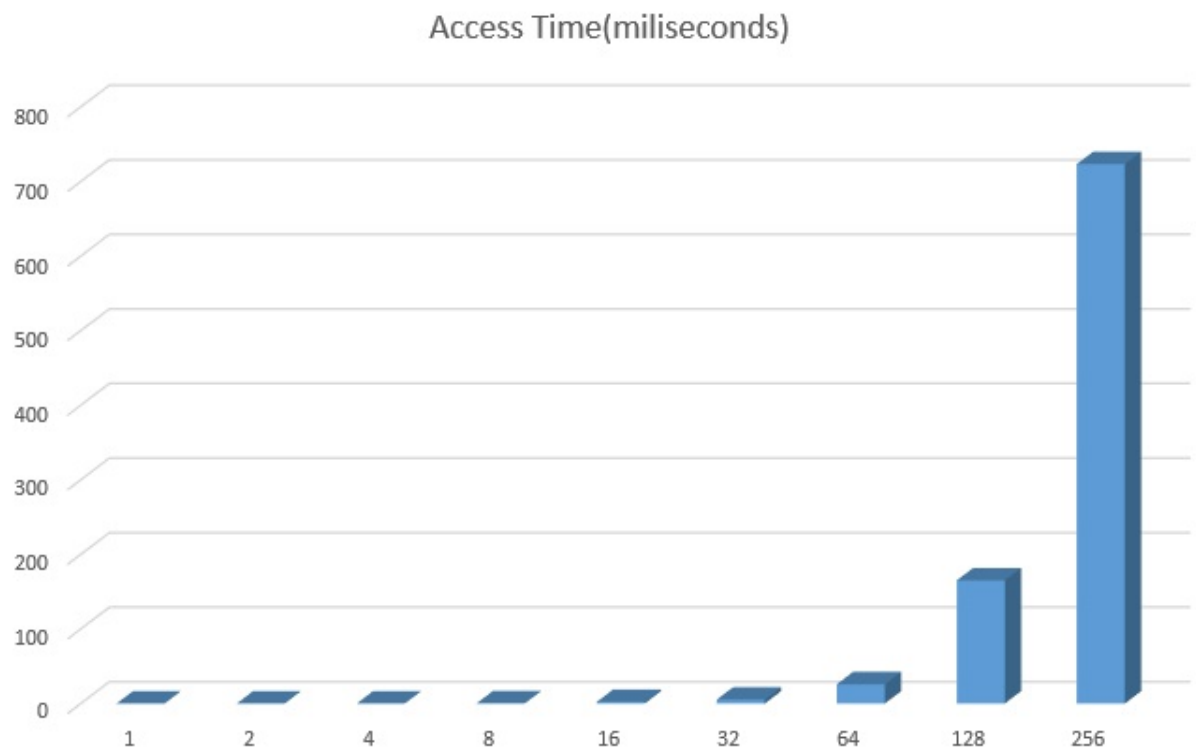

Fig 8: SimpleScalar output for Random Write

NOTE: The Tables representing the cache info (cache hit ratio) are included in the excel sheets provided in the folder "Table_SimpleScalar_Info"