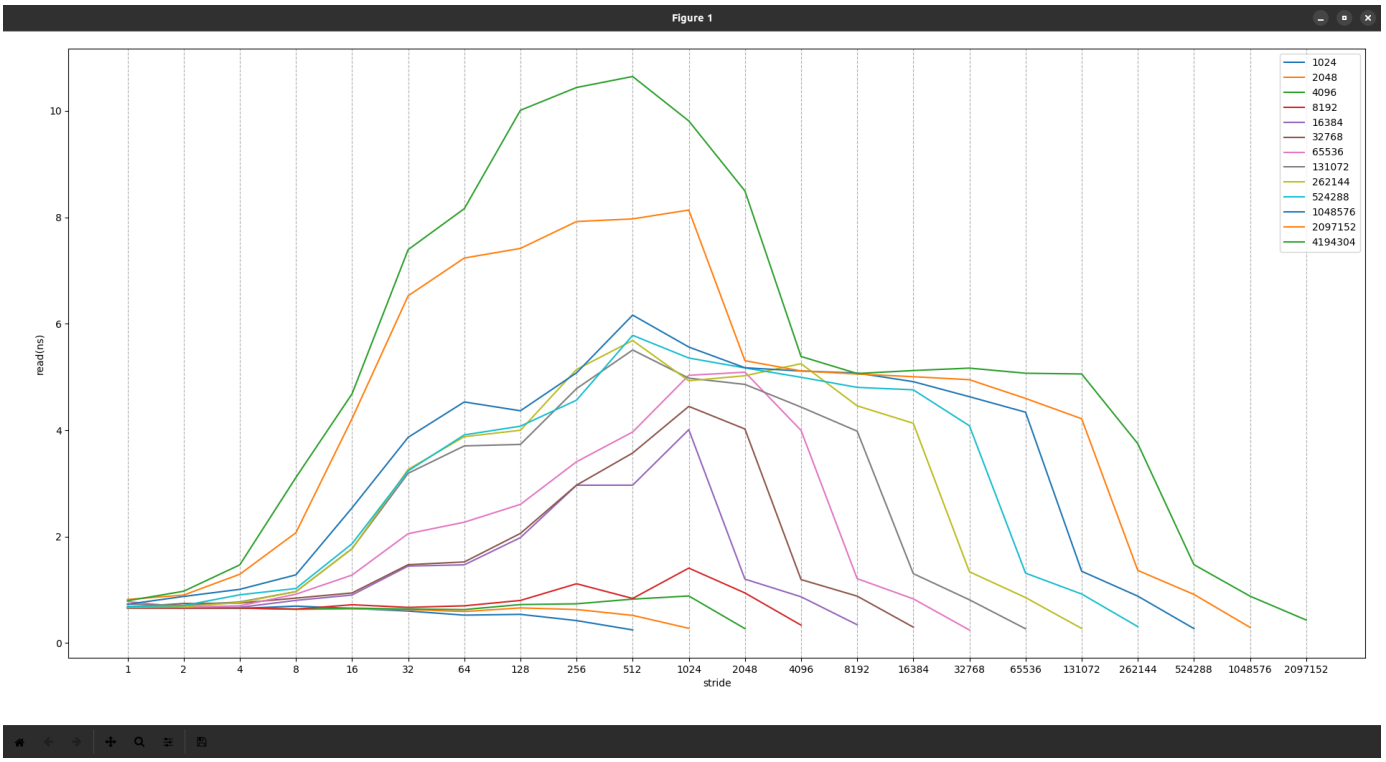
CS4032D Computer Architecture: Assignment 1

Name: Pavithra Rajan
Roll Number: B190632CS

2.4) Using the program results:



Initially, let's determine the number of cache levels. From the chart, we can see that for a csize equal to **1024, 2048, 4096, 8192,** the memory accesses are approximately 0.8 ns. This means that the entire x array fits inside of the **L1 cache,** and no L2 cache requests are required. For csize of **16384, 32768, 65536,131072** the memory access time has a jump. This suggests that the memory access time involves the **L2 cache** since it increased drastically. In this case the x array does not fit in the L1 cache, and access must resort to the L2 cache. For csize of **131072, 262144, 1048576,** there is again an increase in memory access. This indicates that the access is via **L3 cache.** Any further access is initiated via the main memory. All of these observations depict that my system has **three levels** of cache: **L1, L2 and L3.**

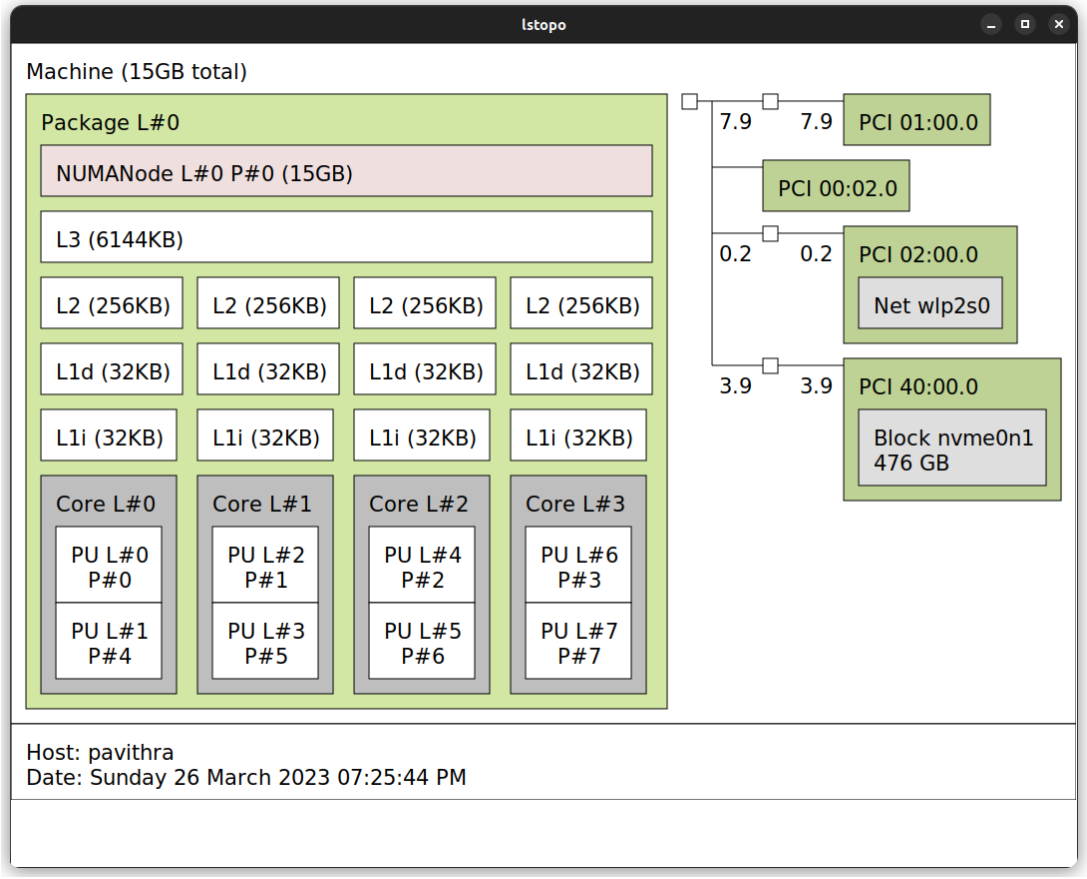
a. What is the overall size and block size of the second-level cache?
The overall size of the L2 cache is determined from the second jump in access time. This occurs at **csize = 131072.** For all smaller sizes, the L2 cache is large enough to hold the entire x array. This means that the size of the L2 cache is

L2_cache_size = 65536*sizeof(int) = 131072*4= 262144 bytes = 256 KiB.

The block size can be determined in the following way. We can see that for csize between **16384 and 65536,** the number of L2 accesses increases until the stride size is equal to **16.** When the stride is 16, the access begins to saturate. The smaller access time is due to prefetching a block. With small strides, several elements from the same block are accessed. When the stride is 1, all elements of the block are accessed. A reduced access time occurs until the stride is equal to the block size and only one element is stored inside a block. The block size of the L2 cache is

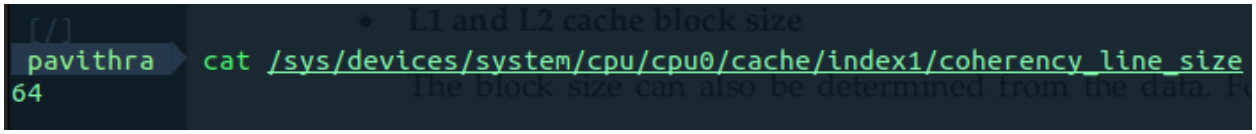
L2_block_size = 16*sizeof(int) = 16*4 = 64 bytes

The above results can be verified via running lstopo. lstopo command is used to show the topology of the system. It gives information about the NUMA memory nodes, shared caches, CPU packages, processor cores and threads and much more.



Here, we can see that there are 3 levels of caches. The cache size of L2 cache per core is **256KB.**

To find the blocksize, we need to find the coherence line size of the cache for the level we want. We can run:



This command gives the block size of the L2 cache which is at index1. It is 64.

Hence, the results obtained via observing the graph are correct.

b. What is the miss penalty of the second-level cache?

The miss penalty is approximately **3.1 seconds**. From the graph we can observe that once the x array does not fit the L2 cache it resorts to the next level of cache. The memory access time at that juncture gives the miss penalty of the required cache level.

c. What is the associativity of the second-level cache?

The associativity of the L2 cache is **4-way set associative**.

The associativity can be determined by looking at access time when the stride is larger than the cache size. When the stride is larger than the cache size each access will be mapped to a different set. As the stride varies a different number of elements get mapped to the same set, and this can be used to determine the associativity. For a stride of csize / 4 the access time is still equal with a L2 memory access time. For a stride size equal with csize/ 8, this is not true anymore, and L3 accesses are needed since more than 4 blocks are mapped in the same set.

d. What is the size of the main memory?

The size of the main memory is **16 GB**. This is obtained by inferring from the graph. We can see that at a csize = 4194304 there is a significant jump. This is because the x array does not fit the x array and must resort to the main memory instead of the L3 cache.

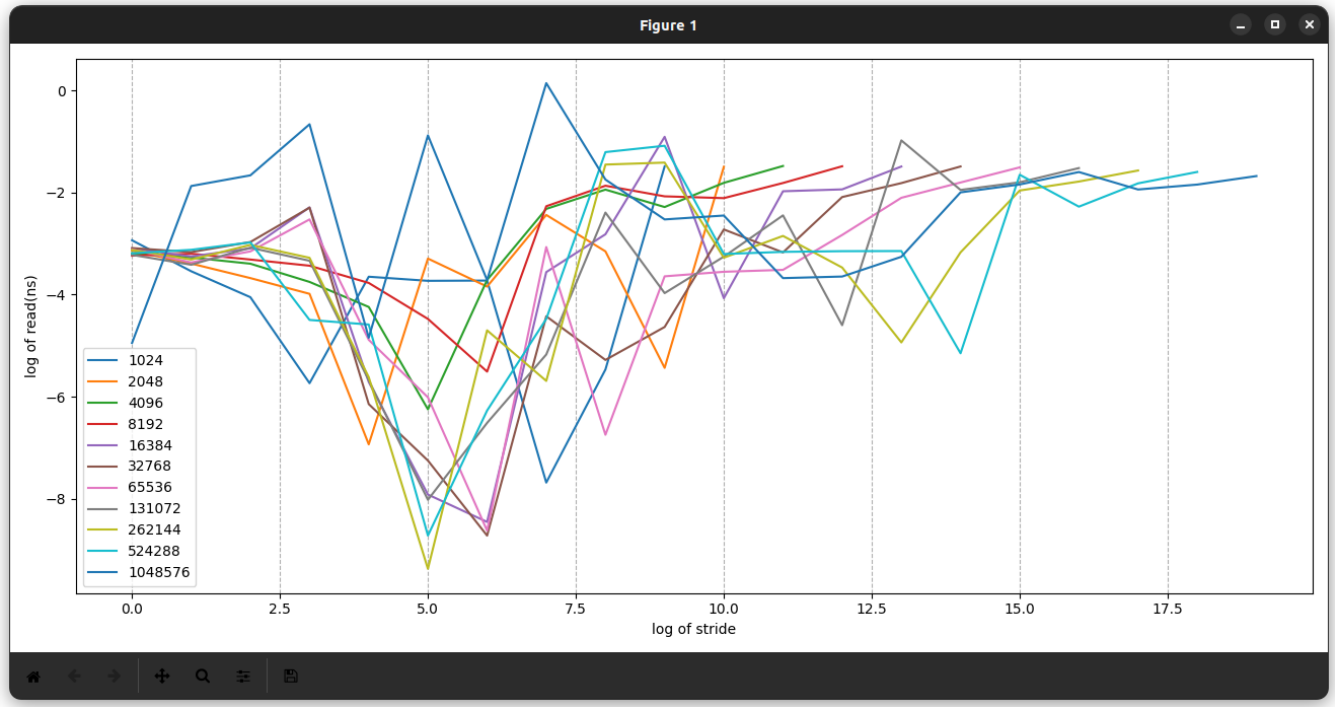
Main memory = 4194304*sizeof(int) = 4194304*4 = 16GB

e. What is the paging time if the page size is 4 KB?

Sum up all the points that come between the 4 KB to upward on read (ns), which is equal to approximately **37 ns.**

Paging Time if the paging size is 4KB is 37 ns.

2.5) If necessary, modify the code to measure the following system characteristics. Plot the experimental results with elapsed time on the y-axis and the memory stride on the x-axis. Use logarithmic scales for both axes, and draw a line for each cache size.



a. What is the system page size?

The system page size is **4096B = 4kiB.** For a stride size of 1024, the memory access time increases when csize is greater than or equal to 1048576. This suggests that there is a TLB entry miss. Before access to the cache memory is done, the TLB is accessed. When the stride is greater than the page size, each access will be to a different page and a separate TLB entry is referenced. This means

Page Size = 1024*sizeof(int) = 1024*4 = 4 KiB.

b. How many entries are there in the TLB?

The number of entries is **64.** TLB misses start when csize = 1048576. This means that starting with this size, the number of TLB accesses exceeds the available TLB entries. A cache size of 1048576 integers is equal to **1048576*sizeof (int) = 1048576 *4 = 1 MiB.** So the largest number of elements that can be referenced by the TLB cannot exceed **1 MiB.** The page size is **4 KiB.** This means that the total number of pages that can be referenced by the TLB is

Number of entries = 1 MiB/4 KiB = 2^18/2^12 = 2^6 = 64 entries.

c. What is the miss penalty for the TLB?

The miss penalty for the TLB is approximately **4ns.** This is because the L2 miss service begins to increase at 4ns.

d. What is the associativity of the TLB?

The associativity is an **8-way set associative.**