Report on Lab3

Ayushi Agarwal
2018ANZ8503
ayushi.agarwal@cse.iitd.ac.in

1. INTRODUCTION

The goal of this report is to study and measure the different parameters of the memory hierarchy in different target machines. The parameters that we focus on in this report are the system cache configurations and some of the memory hierarchy parameters.

2. METHOD

We have focused our study on measuring the following parameters on different target machines:

1. The number of levels of the cache
2. The Size of each level of cache
3. Cache line size of each level
4. System Page Size
5. The number of entries in the TLB
6. The associativity of each cache level
7. Miss penalty of each cache level
8. Page Fault time to the secondary memory

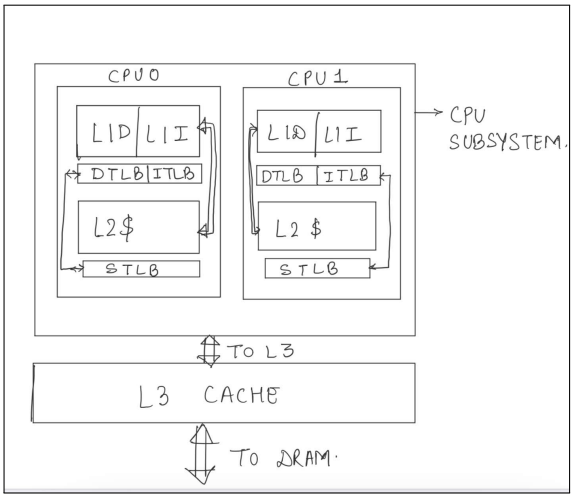


Figure 1. Simplistic View of Memory Hierarchy

Figure 1 shows a simplistic view of the memory hierarchy in a multiprocessor[i]. The figure demonstrates a 2-core machine where each CPU core has a private L1(Instruction and Data Cache) and a private Unified L2 cache. The L3 cache is shared between the two cores. The figure also demonstrates the two level of translation lookaside buffer. New generation processors have two levels (even three) of TLB's to limit the size of this cache as the virtual memory keeps getting bigger. The figure shows two levels of TLB, one is the DTLB and ITLB at the L1$ level and the other is a secondary TLB or a unified L2 TLB.

The sizes of these different components vary from machine to machine and have a direct impact on the performance of a processor. The goal of this study is to benchmark the memory hierarchy.

[i]This image is only for understanding purposes and does not depict all the components of the CPU subsystem

The Methodology that we have used is that we have tried to access arrays of different sizes with varying strides and tried to see the change in the access time[1]. As the stride changes for a fixed array size, the access time varies because of many factors which includes system pages and caches, etc.

Hypothesis:

- For an array of a certain size, if the stride is less than the size of one cache block or line, then the access time of elements would be faster because the whole line is fetched in the cache on a miss. But as soon as the $stride > Cachelinesize$, with every consecutive access a new line has to be fetched from the memory (or lower level cache), so we don't see any benefits of cache line prefetching. Hence, we see a sudden increase in the access time.

- For an array of a certain size, as soon as the stride crosses the page size of the system, we will see TLB misses and hence an increase in the access time.

- The size of the array also has impact on the access time. As soon as the array crosses the size of one level of cache, we see an increase in the access time until the point that the array becomes so large that it cannot fit in the L3 cache. In that scenario, the accesses would go to the DRAM hence increasing the access times by almost 40-50ns.

We report and analyse these results on the following machines:

1. **Intel i7-7500U** : This machine has the following configurations:
    - 3-Level Caches with L1D is 32KB(8-way), L2 is 256KB(4-way non-inclusive) and L3 is 4MB (2MB/core 16-way inclusive and shared among all cores) with 64B cache blocks.
    - The system page size is 4K.
    - Data TLB has 1G/4 entries and 4K/64 entries. L2 TLB has 4K/2M pages with 1536 entries.
2. **Intel i7-7700** : This machine has the following configurations:
    - 3-Level Caches with L1D is 32KB(8-way), L2 is 256KB(4-way non-inclusive) and L3 is 8MB (2MB/core 16-way inclusive and shared among all cores) with 64B cache blocks.
    - The system page size is 4K.
    - Data TLB has 1G/4 entries and 4K/64 entries. L2 TLB has 4K/2M pages with 1536 entries.
3. **AMD EPYC 7551P**: This machine has the following configuration[2]:
    - 3-Level Caches with L1D is 32KB(8-way), L2 is 512KB(8-way inclusive) and L3 is 64MB (8MB/core, 16-way inclusive and shared among all cores) with 64B cache blocks.
    - 64 entry L1 TLB, all page size
      1,532-entry L2 TLB, no 1G pages.

3. RESULTS

In this section, we have reported our analysis of the various parameters of the memory hierarchy. **Figure 2** shows the various levels of cache present in different machines that we have used for our analysis, as mentioned in Section 2.

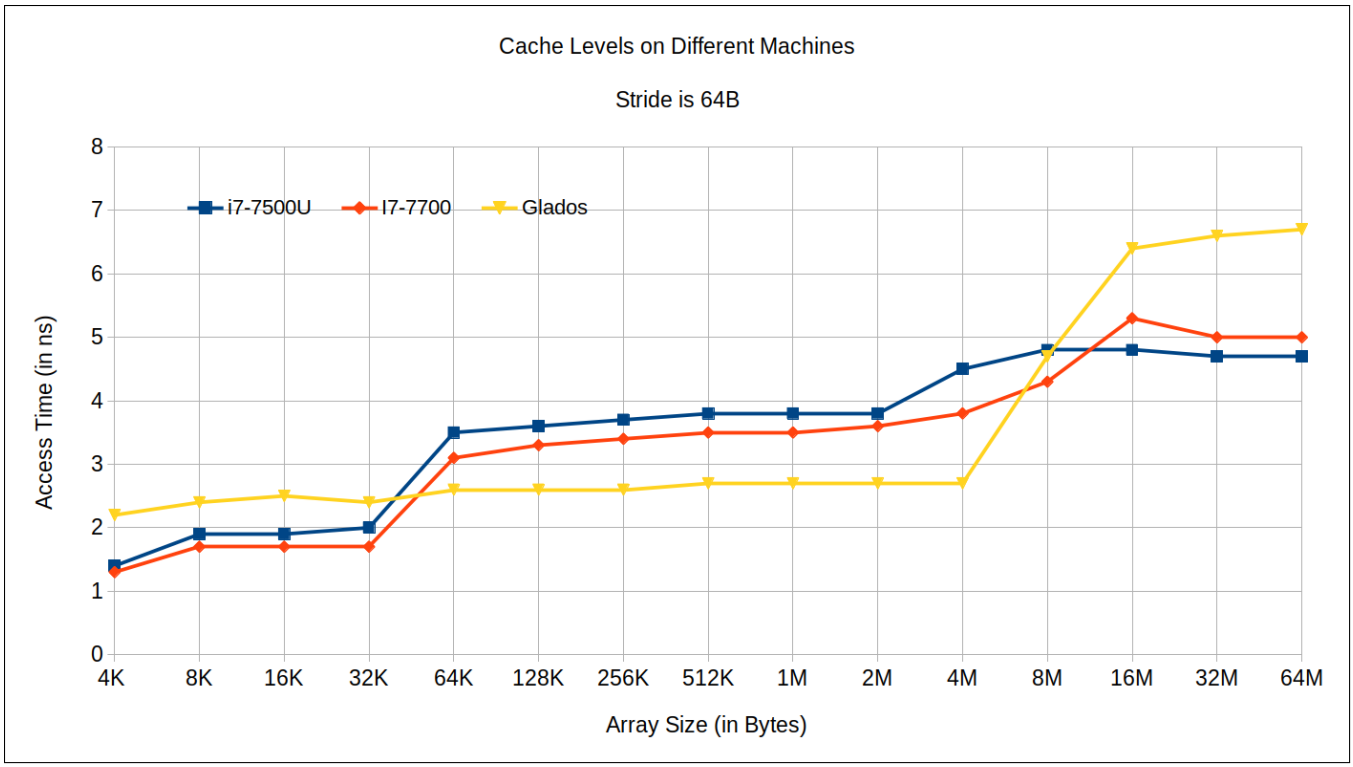


Figure 2. Levels of the Caches in Various Machines: Intel i7-7500U, Intel i7-7700 and AMD EPYC 7551P (Glados)

The figure shows that all these machines have three levels of caches. The X-axis shows the increasing array sizes and the Y-axis shows the access time.

- Intel i7-7500U has 32KB *L1 cache* as we see the first step increase in access time at 32KB. The second step increase (though not very significant) is seen at 256KB which is the *L2 cache* size. The *L3 cache* of this machine is divided into 2 banks (2MB/core). The figure shows a step increase at 2MB because after this the access would be to the non-local L3 cache bank. We see a similar step increase at 4MB after which the access would see L3 misses. So L3 size is 4MB. After this the access times become almost of the same order until we start getting memory misses and secondary disk starts getting accessed.
- Intel i7-7700 has similar behaviour and configurations as Intel i7-7500U except for the L3 cache. This machine has 4 physical cores so it has 2MB/core of banked L3 cache. So we see the first step increase in time at 2MB and another increase at 8MB. L3 cache is 8MB.
- AMD EPYC 7551P or Glados curve doesn't show very clear demarcations of the cache. This could be due to uncontrolled system setup for running our code (unable to reboot remotely to get good lines). We would try to analyse this machine in the next subsection when we discuss the target machines independently.

Target Machine: Intel i7-7500U

Figure 3 shows the results that we obtained on the target machine Intel i7-7500U by accessing arrays of different sizes at varying strides. In Section 2, we have mentioned the actual configuration of the memory hierarchy present on this machine which we will analyse using this section.

Observations:

- **Number of Cache Levels and Size:** We have seen in **Figure 2** that the number of cache levels in this machine are 3 so we have L1, L2 and L3 cache in the memory hierarchy of sizes 32KB, 256KB and 4MB respectively. This is also validated in **Figure 3** and in **Figure 4c** and **4d** where we see an increase in the access time as soon the memory needed by a particular array increases beyond a cache level. The Figure depicts the size of caches with dotted lines.

- **Cache Block Size:** All the three caches have 64B *cache line or block* because we see an increase in the access time at a stride of 64B for all the array sizes. This is because the effects of prefetching diminishes when we access a new cache line with every consecutive access.

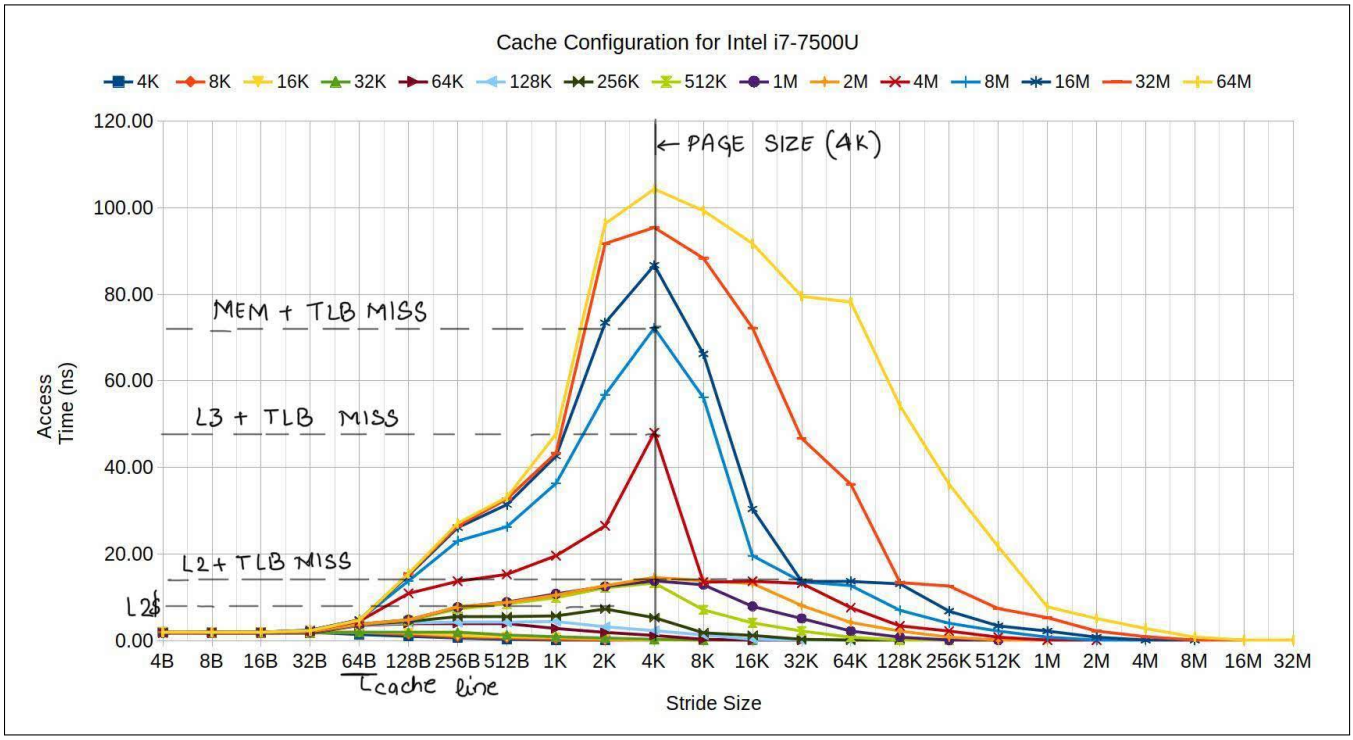


Figure 3. Cache configurations for Machine Intel i7-7500U

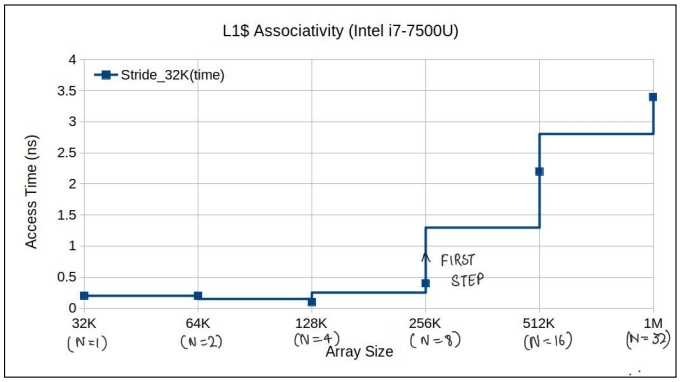
- **System Page Size:** We see in **Figure 3** that for array of size > 256KB(L2 cache) and at stride of >= 4KB, we see an extra penalty apart from L2 miss penalty. This is because as soon as the stride becomes equal to the system page size, we will get TLB misses because every consecutive access would be to a new page. We consistently see high access times at Stride = 4KB with increasing array sizes. Also we can see that for higher strides (> 4K) and for larger arrays the curves are showing constant access times when they incur TLB + L2 miss. So System Page size is 4KB.

- **Number of TLB entries: Figure 4e** shows how the access time increases at a fixed stride of 4KB with varying array sizes for both Intel i7 machines. The number of TLB entries can only be determined when the size of the array traversed is greater than the memory mapped by the TLB. This is evident in the figure. We see a step increase in the access time at Array size >= 256KB. So, the amount of memory mapped by the level 1 TLB is 256KB. **Since page size is 4KB, the number of TLB entries = 64**. We see another step increase at 2MB and 4MB for Intel i7-7500U. This can be due to 512/1024 small page entries in the STLB out of 1536.

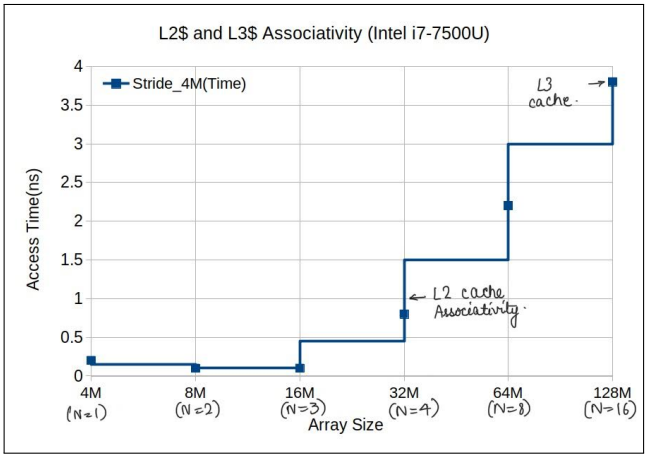
- **Cache Associativity: Figure 4a and 4b** shows the cache associativity for Intel i7-7500U. we have modified the code to derive the associativity of each cache. This depends on whether or not the lower level cache is

inclusive of the higher level, in which case we have used strides equal to the size of the lower level cache.

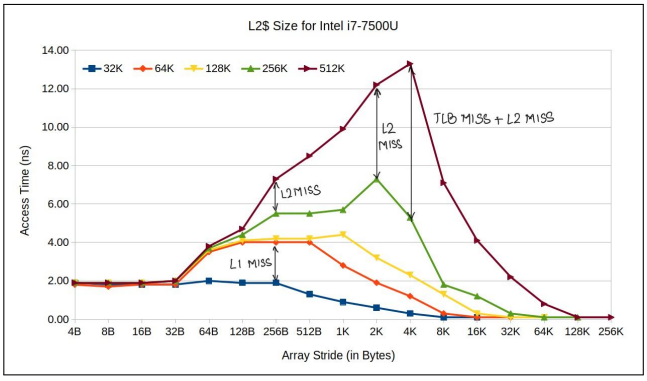
L1 cache: **Figure 4a** shows that when we traverse the array at strides equal to the size of L1 cache (32KB), all elements would have addresses that are 32K apart and so would map to the same set 0. So, we see a step increase in the access time as soon as the number of elements accessed are > associativity of the cache. This is because the 9th access would map to the same set but causing evictions.



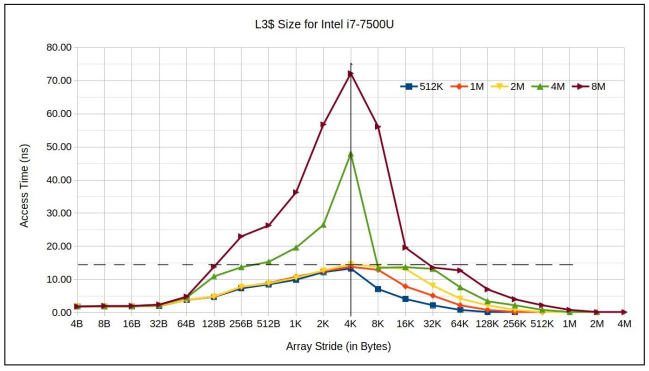
(a) L1$ Associativity



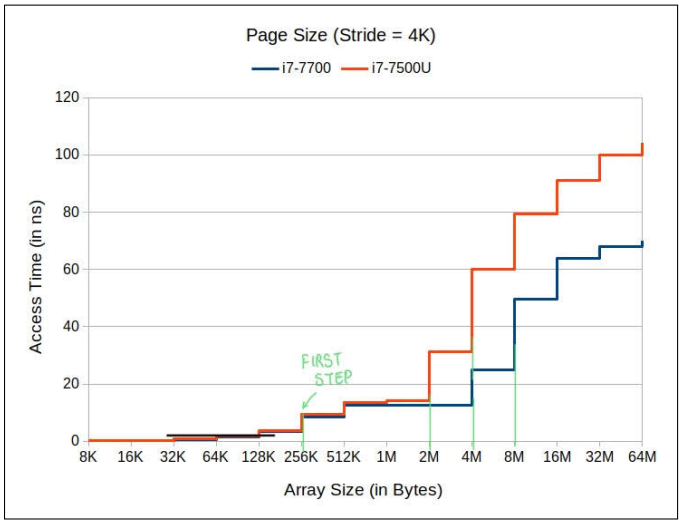
(b) L2$ and L3$ Associativity



(c) L2 Cache Size



(d) L3 Cache Size



(e) Page Size of Intel i7-7500U and i7-7700

Figure 4. Cache Associativity and Miss penalties of Intel i7-7500U Machine (e) shows the Page size for Intel i7-7500U and Intel i7-7700

L2 and L3 cache: **Figure 4b** shows that the L3 cache is inclusive of the L2 cache. When we access the array at stride of 4M (size of L3 cache), all the elements at addresses 4M apart, map to the same set in both L2 and L3 cache. We see a step increase at N=4 which is the associativity of L2 cache. And we see a step increase at N=16 which is associativity of L3 cache.

-**Miss Penalties: Figure 4c and 4d** shows the miss penalties of various caches and TLB. The L1 miss penalty is 2-6 ns. The L2 miss penalty is 5-30 ns. This is because if the L2 miss is handled by the local L3 bank then penalty is upto 12ns. However, different bank L3 hit has more overhead. The overhead can go to 40 ns in case of TLB miss. The L3 miss penalty is upto 80-100 ns. The TLB miss penalty is upto 15-20 ns.

-**Access Latencies:** We can similarly measure the access latencies of all the cache levels using Figure 4c and 4d. L1 access latency is 2ns. L2 access time is upto 7ns. And L3 access time is upto 45ns (with TLB miss). The L3 access time for local core is less than the L3 access time for non-local core in the order of 10ns.

-**Page Fault Time:** This happens when the page accessed is not currently mapped to a physical address in the main memory. The OS handles Pages faults since the page has to be read from the secondary memory. To program this we have to take an array of size that is larger than the size of the main memory of the machine or make sure that other processes congest the RAM enough for our program to show page faults. We don't explicitly see any page faults in our execution.

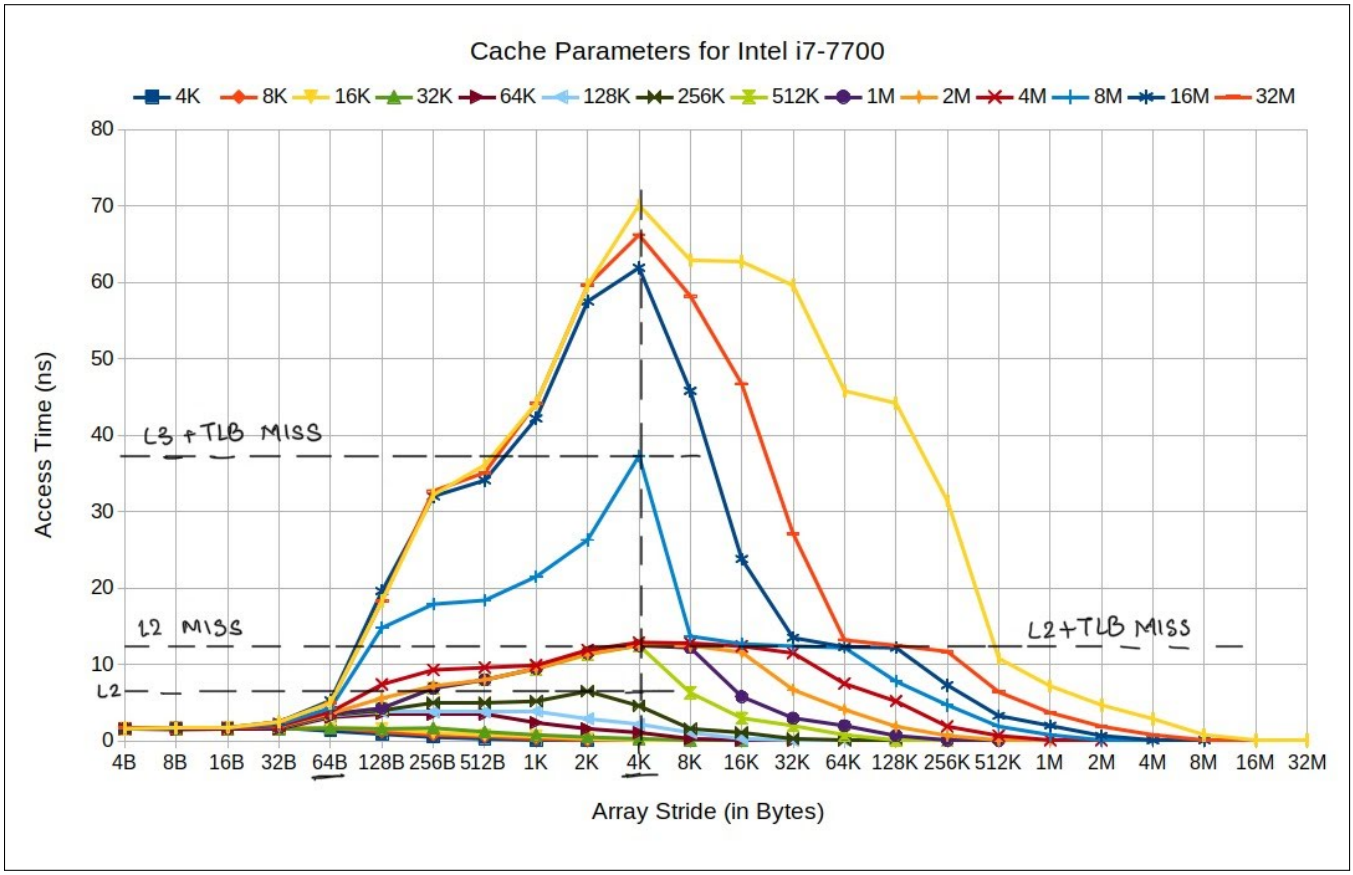


Figure 5. Cache configurations for Machine Intel i7-7700U

Target Machine: Intel i7-7700

Figure 5 shows the results that we obtained on the target machine Intel i7-7700 by accessing arrays of different sizes at varying strides. In Section 2, we have mentioned the actual configuration of the memory hierar-

chy present on this machine which we will analyse using this figure.

Observations:

- **Number of Cache Levels and Size:** We have seen in **Figure 2** that the number of cache levels in this machine are 3 so we have L1, L2 and L3 cache in the memory hierarchy of sizes 32KB, 256KB and 8MB respectively. This is also validated in **Figure 5** where we see an increase in the access time as soon the memory needed by a particular array increases beyond a cache level. The Figure depicts the size of caches with dotted lines.

- **Cache Block Size:** All the three caches have 64B *cache line or block* because we see an increase in the access time at a stride of 64B for all the array sizes.

- **System Page Size:** The system page size for this machine is equal to Intel i7-7500U (which is 4KB) since they belong to the same family.

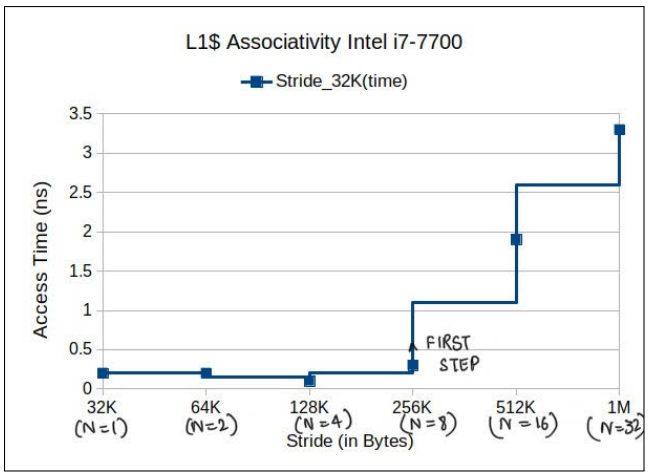
- **Number of TLB entries:** The number of TLB entries are also same as Intel i7-7500U.

- **Cache Associativity: Figure 6a and 6b** shows the cache associativity for Intel i7-7700.

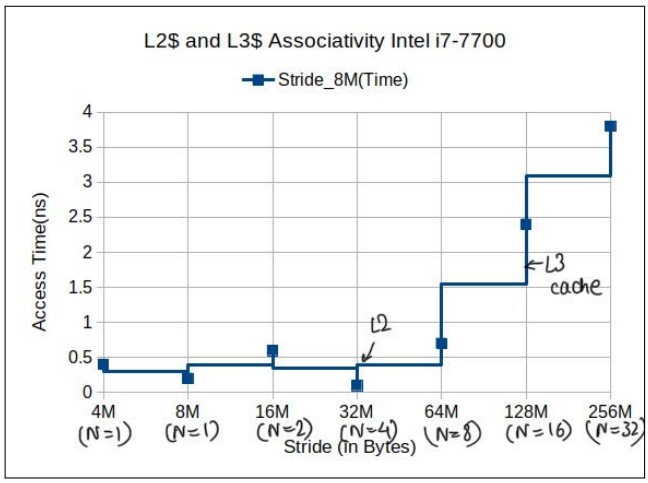
L1 cache: **Figure 6a** shows that when we traverse the array at strides equal to the size of L1 cache (32KB), all elements would map to the same set 0. So, we see a step increase in the access time as soon as the number of elements accessed are > associativity of the cache(which is 8). This is because the 9th access would map to the same set but causing evictions.

L2 and L3 cache: **Figure 6b** shows that the L3 cache is inclusive of the L2 cache. When we access the array at stride of 8M (size of L3 cache), all the elements map to the same set in both L2 and L3 cache. We see a step increase at N=4 which is the associativity of L2 cache. And we see a step increase at N=16 which is associativity of L3 cache.

-**Miss Penalties: Figure 5** shows the miss penalties of various caches and TLB. The L1 miss penalty is 2-6 ns. The L2 miss penalty is 5-35ns ns. The L3 miss penalty is upto 80-100 ns. The TLB miss penalty is upto 15-20 ns.



(a) L1$ Associativity



(b) L2$ and L3$ Associativity

Figure 6. Cache Associativity Intel i7-7700U Machine

Target Machine: AMD EPYC 7551P

Figure 7 shows the results that we obtained on the target machine AMD EPYC 7551P by accessing arrays of different sizes at varying strides. In Section 2, we have mentioned the actual configuration of the memory hierarchy present on this machine which we will analyse using this figure.

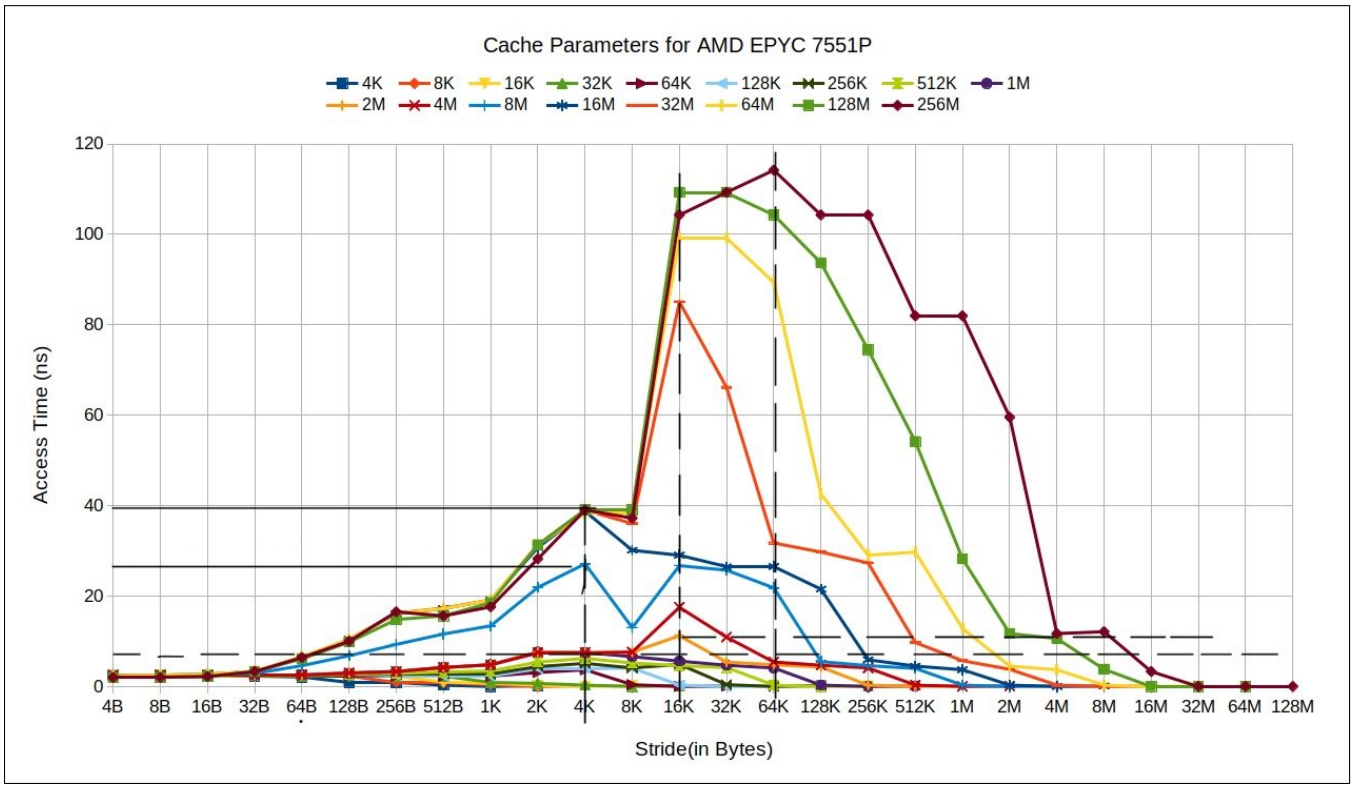


Figure 7. Cache configurations for Machine AMD EPYC 7551P

Observations:

- **Number of Cache Levels and Size:** From **Figure 7** we estimate the number of cache levels in this machine to be 3, however there is not much difference in execution time. We see a step (not very significant) increase at 32KB array size and at 512KB. Then we see a step increase at 4MB and 8MB. We suspect this to be because of the banked L3 cache among various cores. So we have L1, L2 and L3 cache in the memory hierarchy of sizes 32KB, 512KB and 64MB respectively.

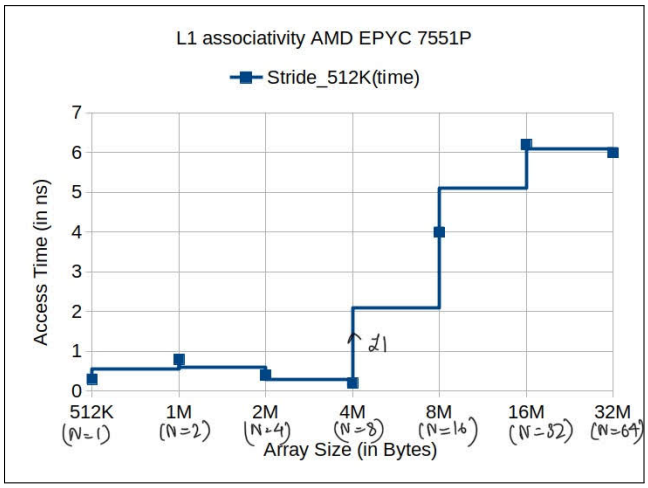
- **Cache Block Size:** All the three caches have 64B *cache line or block* because we see an increase in the access time at a stride of 64B for all the array sizes.

- **System Page Size:** As we see in the plot, we see that there are multiple page sizes in this machine. We see extra miss penalty apart from the cache at strides of 4K, 16K and 64K. This can be because this machine supports page of all sizes.

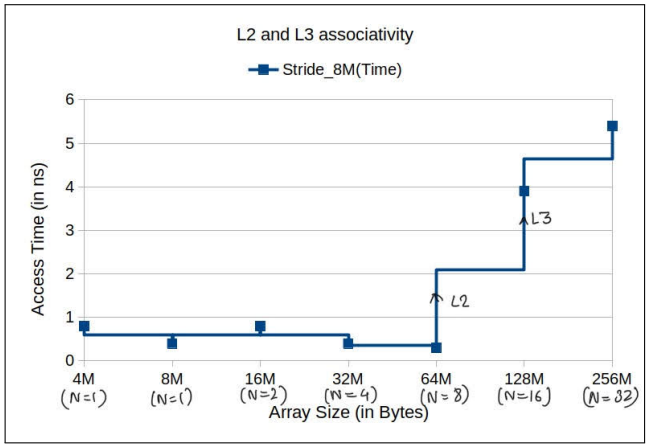
- **Cache Associativity: Figure 8a and 8b** shows the cache associativity for AMD EPYC 7551P.

L1 cache: **Figure 8a** shows that L2 is inclusive of L1 cache and when we traverse the array at strides equal to the size of L2 cache (512KB), all elements would map to the same set 0 in both L1 and L2 cache. So, we see a step increase in the access time as soon as the number of elements accessed are > associativity of the cache(which is 8). This is because the 9th access would map to the same set but causing evictions.

L2 and L3 cache: **Figure 8b** shows that the L3 cache is inclusive of the L2 cache. When we access the array at stride of 8M (size of L3 cache bank), all the elements map to the same set in both L2 and L3 cache. We see a step increase at N=8 which is the associativity of L2 cache. And we see a step increase at N=16 which is associativity of L3 cache.



(a) L1$ Associativity



(b) L2$ and L3$ Associativity

Figure 8. Cache Associativity of AMD EPYC Server Machine

-Miss Penalties: Figure 7 shows the miss penalties of various caches and TLB. The L1 miss penalty is 2-6 ns. The L2 miss penalty is 5-100ns. The L2 miss penalty if the L3 access is to local L3 bank is 5-25ns. However, the L3 access can go to upto 100ns if non-local L3 is accessed.

CONCLUSIONS

We have analyzed in depth the various memory hierarchy parameters using C benchmark code. Some other parameters that can be studied are TLB associativity, Page fault times, Memory size, memory read and write latency, etc.

REFERENCES

1. Hennessy,JohnL.and David A. Patterson.2006.Computer Architecture,Fourth Edition: A Quantitative Approach.Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
2. AMD EPYC 7551P, `https://en.wikichip.org/wiki/amd/epyc/7551p/`