In the real world, devices are expected to handle diverse data set sizes and diverse data access patterns. Because of this, well-written, efficient code is needed to reduce cache misses and TLB misses.

For this design project, I decided to investigate the performance characteristics of different levels of the memory hierarchy (L1, L2, L3 caches, and main memory). To do this, I found the latency and bandwidth of each memory level using benchmarks. I coded the benchmarks in C++ and evaluated the following:

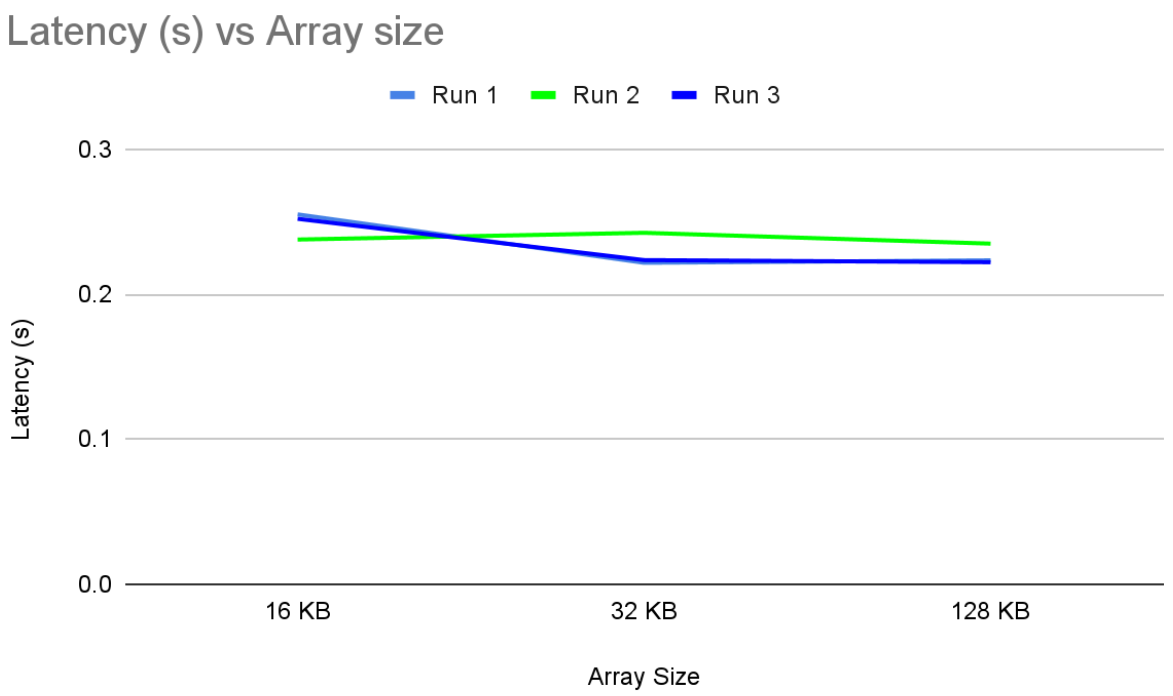
- Data Access Patterns: do different access patterns affect memory performance?
- TLB Performance: what is the impact of TLB misses on memory access latency?
- What is the correlation between memory hierarchy performance and application performance?

This project helped me understand how to interact with the memory hierarchy. From this, I was able to write a cache-friendly code that reduced latency.

Run benchmarking tests for latency (run multiple times for consistency):

L1 Array sizes: 16KB - 128KB

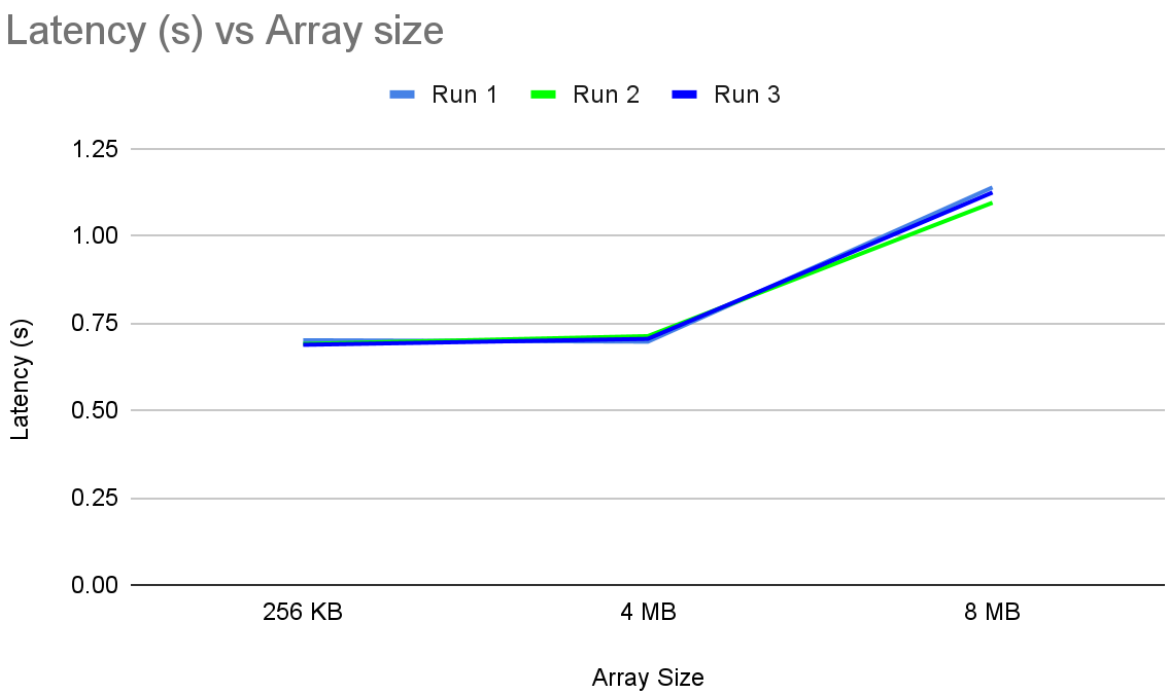
Array Size (L1)	16 KB	32 KB	128 KB
Time 1 (s)	.254984	.221616	.223326
Time 2 (s)	.23765	.242283	.234825
Time 3 (s)	.251944	.223473	.222023



Results show similar latency as array sizes increase in the L1 region. This is because the array size allows for the CPU to prefetch and access data efficiently.

L2 Array sizes: 256KB - 8MB

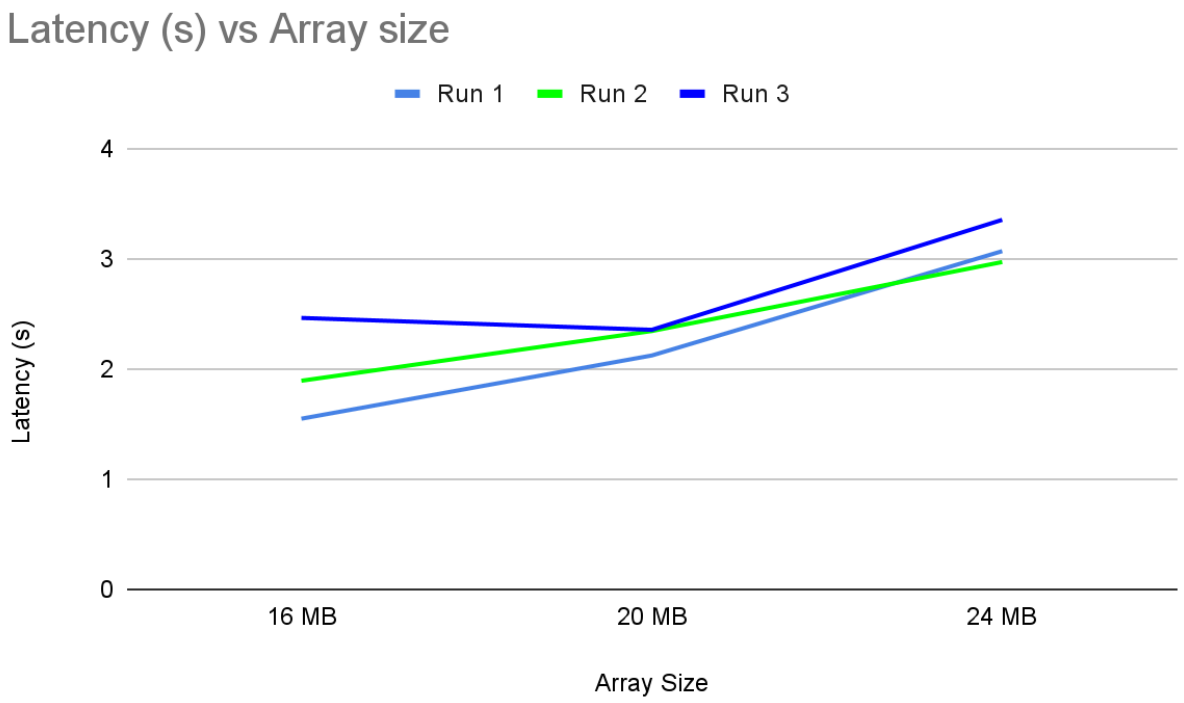
Array Size (L2)	256 KB	4 MB	8 MB
Time 1 (s)	.70099	.696764	1.13868
Time 2 (s)	.690337	.712685	1.09479
Time 3 (s)	.687811	.704842	1.12412



Results show that as array size in the L2 region increase, there is a large increase in latency

L3 Array Sizes: 16MB - 24MB

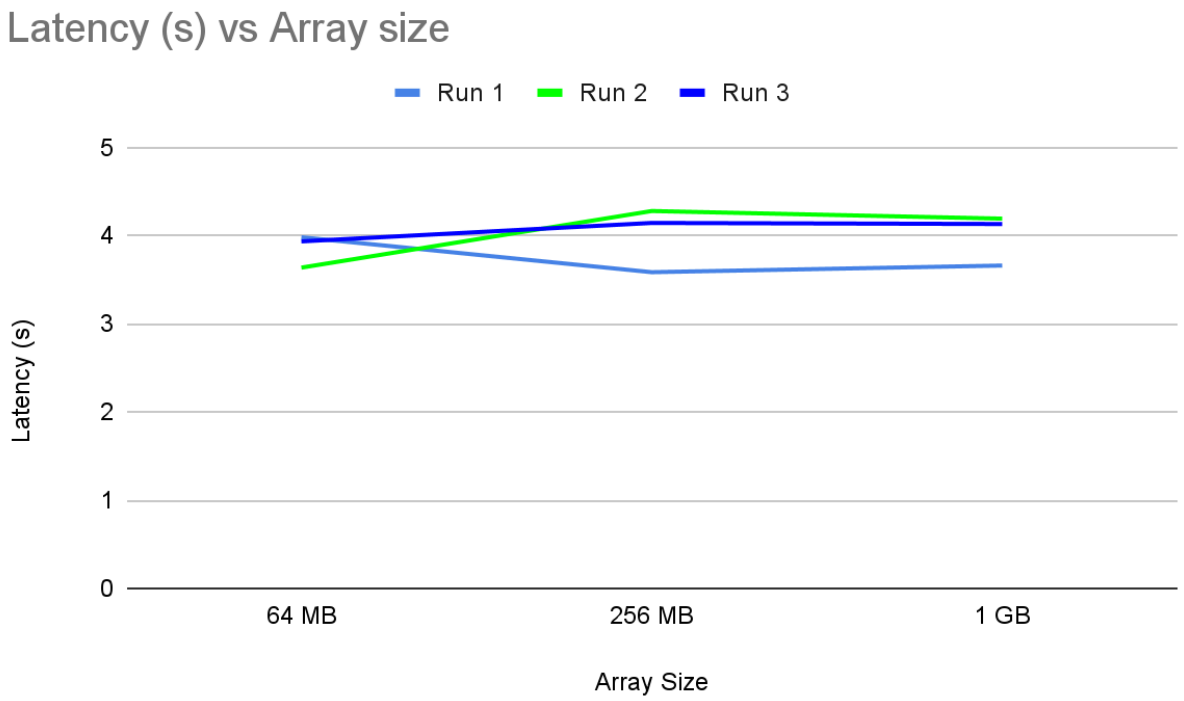
Array Size (L3)	16 MB	20 MB	24 MB
Time 1 (s)	1.54846	2.12195	3.06843
Time 2 (s)	1.8927	2.34362	2.96974
Time 3 (s)	2.46265	2.35487	3.35258



Latency also increases in the L3 array size. There is a large difference in latencies here, due to the large difference in size between each of the data sets.

Main memory Array sizes: 64 MB - 1GB

Array Size (Main memory)	64 MB	256 MB	1 GB
Time 1 (s)	3.98198	3.58556	3.66098
Time 2 (s)	3.63743	4.2793	4.19287
Time 3 (s)	3.93483	4.14398	4.13191



In the main memory array size, latency tends to stagnate, though it does increase slightly as array size increases.

Changing Array Access Order:

Access order: L1 > L2 > L3

Array Size	16KB (L1)	8 MB (L2)	24 MB (L3)
Time 1 (s)	0.250626	1.45919	3.10405
Time 2 (s)	0.251695	0.815804	3.21727
Time 3 (s)	0.254162	0.783208	3.74387

Access order: L2 > L3 > L1

Array Size	16KB (L1)	8 MB (L2)	24 MB (L3)
Time 1 (s)	0.22426	0.862456	2.67848
Time 2 (s)	0.225345	1.20318	2.73557
Time 3 (s)	0.233313	1.2422	3.81854

Access order: L3 > L2 > L1

Array Size	16KB (L1)	8 MB (L2)	24 MB (L3)
Time 1 (s)	0.224302	0.848373	3.17325
Time 2 (s)	0.23478	1.15286	2.9962
Time 3 (s)	0.224466	0.795514	2.86697

Do different access patterns affect memory performance?

Yes. when you access an L1 cache after accessing a larger cache (L2 or L3) the data is more likely to be loaded in the L1 cache already, which decreases latency. The same goes for an L2 cache accessing data from an L3 cache.

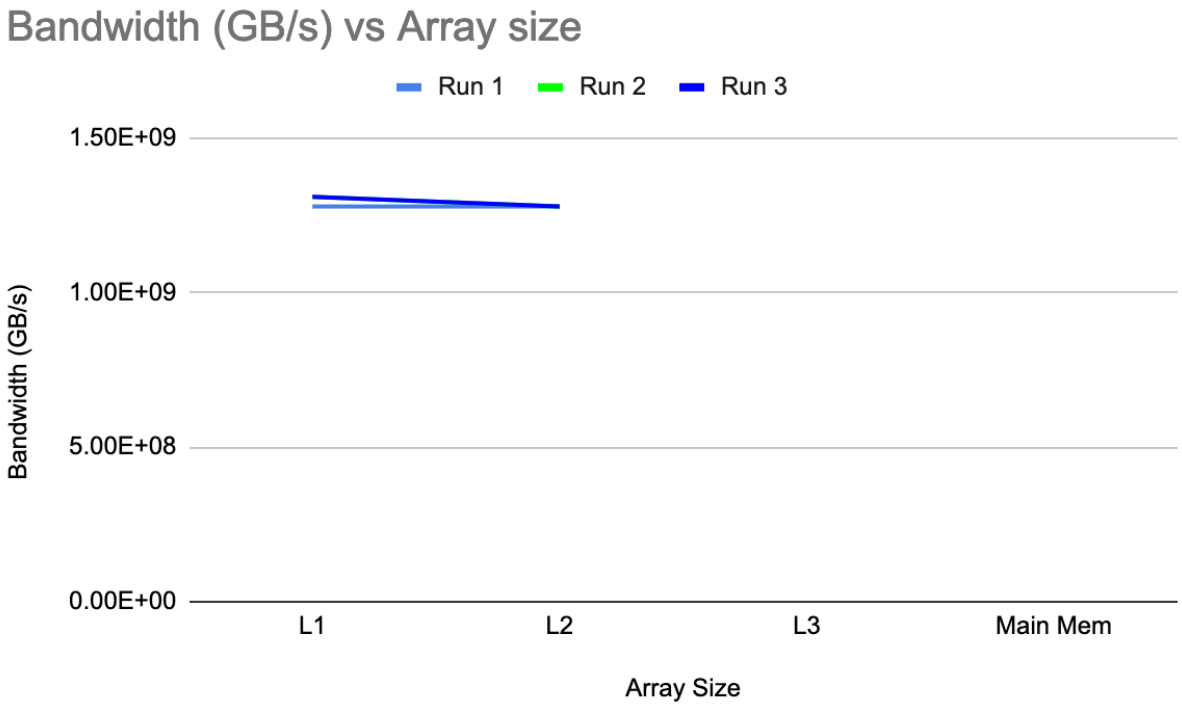
TLB:

Array Size	Number of pages	Time taken (s)
4 KB	1024	0.000107125
4 KB	4096	0.0007985
248 KB	4096	0.00572958
248 KB	8*1024	0.00871375
248 KB	16*1024	*Bad allocation error*

TLB Performance: what is the impact of TLB misses on memory access latency?
TLB misses can significantly impact memory access latency. When misses occur, it disrupts the translation of virtual addresses to physical addresses.

Run benchmarking test for Bandwidth (Run multiple times for consistency, change array size)

Array Size	Bandwidth (GB/s)			Time (s)
Run number	1	2	3	(time stays constant)
L1	1.30944e+09	1.30944e+09	1.27826e+09	4.1e-08
L2	1.27826e+09	1.27826e+09	1.27826e+09	4.2e-08
L3	1.27826e+09	1.27826e+09	1.27826e+09	4.2e-08
Main Memory	1.27826e+09	1.27826e+09	1.27826e+09	4.2e-08



Though it is hard to see, there is a decrease in bandwidth as the cache size moves from L1 to L2. This is because L1 has a faster access time compared to L2.