You should design a set of experiments that will quantitatively reveal the following:

(1) the read/write latency of cache and main memory when the queue length is zero (i.e., zero queuing delay
(2) the maximum bandwidth of the main memory under different data access granularity (i.e., 64B, 256B, 1024B) and different read vs. write intensity ratio (i.e., read-only, write-only, 70:30 ratio, 50:50 ratio)
(3) The trade-off between read/write latency and throughput of the main memory to demonstrate what the queuing theory predicts
(4) the impact of cache miss ratio on the software speed performance (the software is supposed to execute relatively light computations such as multiplication)
(5) the impact of TLB table miss ratio on the software speed performance (again, the software is supposed to execute relatively light computations such as multiplication)

For this project, we were suggested to use Intel's Memory Latency Checker. However, Intel MCL does not run on Mac M1 chips, and after many hours of struggling to set up a Linux emulator on my computer, I decided to create the experiments using C++ code (with the help of Chatgbt). To ensure the results from my experiments were correct, I compared them to the expected average results expected for my machine.

Experiment 1)

In order to test the read and write latencies for L1 L2 and L3 caches, I changed the size of the arrays using the formula: $\frac{Cache\ size\ (bytes)}{size\ of\ each\ element\ (bytes)}$. from this formula, I got the following results:

	L1 Cache	L2 Cache	L3 Cache
Avg Size	32KB - 64KB	256KB - 1 MB	4MB - 64 MB
Min Elements	8192	65536	1048576
Max Elements	16384	262144	167777216
Array Size	32 x 1028	256 x 1028	4 x 1028 x 1028
Read Latency	1.54332 ns	1.53606 ns	1.53613 ns
Write Latency	1.53426 ns	1.53542 ns	1.54114 ns

Experiment 2)

Granularity	Read Ratio	Write Ratio	Bandwidth
64B	1	0	3388.97 MB/s
64B	0	1	2821.47 MB/s
64B	.7	.3	5842.7 MB/s
64B	.5	.5	4376.55 MB/s
256B	1	0	8363..02 MB/s
256B	0	1	3133.98 MB/s
256B	.7	.3	6442.85 MB/s
256B	.5	.5	7271.71 MB/s
1024B	1	0	19940.2 MB/s
1024B	0	1	12180.6 MB/s
1024B	.7	.3	1998.83 MB/s
1024B	.5	.5	2003.52 MB/s

Experiment 3)

Read Ratio	Write Ratio	Avg Read Latency	Read Throughput	Avg Write Latency	Write Throughput
1	0	4.36741 ns	873.446 MB/s	∞ ns	0 MB/s
0	1	∞ ns	0 MB/s	4.12228 ns	925.386 MB/s

In general, higher write-load ratios lead to higher write latencies, and throughput tends to increase with the load until it hits a saturation point. When that point is hit, there might be a decrease in its value due to contention.

Experiment 4)

Array Size	Time elapsed	Performance (ops/s)
1024 x 1024	.00385 s	2.30254 e8
256 x 1024	.00175 s	1.49594 e8
32 x 1024	.00012 s	2.77401 e8

In general, Larger arrays cause more cache misses and take longer compared to smaller arrays. If I had more time, I would have liked to test out different cache access patterns to see if they affected the performance.

Experiment 5)

Array Size	Time elapsed	Performance (ops/s)
1024 x 1024 x 128	.0273 s	3.66242 e7
1024 x 1024 x 64	.0263 s	3.8042 e7
1024 x 1024 x 32	.0328 s	3.04819 e7

As the array size increases, there are fewer TLB misses and it takes less time. If I had more time, I would have liked to test out different cache access patterns to see their effect on the TLB misses.