**Hardware & Software Setup:**

* **CPU Specs:**
* AMD Ryzen 7 9800X3D with 8 cores, each with 1MiB L2, and a 96MiB shared L3 cache.



* **OS/Environment:**
* Linux (WSL Ubuntu 22.04) on Windows 11.
* Java OpenJDK 17 headless installed via *sudo apt install openjdk-17-jdk-headless*

**Algorithm and Code break down:**

1. Set maximum array size to test to 512MiB.
2. As mentioned in the Assignment document, a typical Intel/AMD cache line is 64 Bytes. Since an int is 4 bytes, stepping by *step = 64/4 = 16* ints will touch one element per cache line.

A black background with white text

AI-generated content may be incorrect.

1. Ensure each array, regardless of its size is accessed the same number of times. This prevents bigger arrays taking more time simply because they need to be accessed more times.

A black screen with colorful text

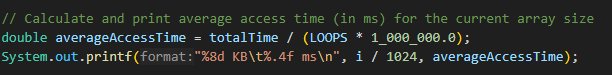
AI-generated content may be incorrect.

1. Measure the time from the uniform-access nested loops above, repeating 10 times to calculate an average and mitigate the impacts of outliers.

A screenshot of a computer program

AI-generated content may be incorrect.

1. Printing results (average time per array)



**Results and Evaluation:**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

* It is evident that from 4KiB through 64MiB times stay around 3-5.5 ms. This tells me that up to 64MiB, arrays fit in cache. Which lines up with the 96MiB L3 cache my CPU has.
* At 128MiB, time jumps considerably from 5.43 to 12.28ms. This sudden increase shows that the LLC has been exceeded and RAM is now being used to fetch from main memory.
* Since these results were gathered using WSL (Windows Subsystem for Linux), OS scheduling can introduce some jitter (in the ~1ms range). Furthermore, Java can also be a bit less precise than C as it cannot access direct CPU cycles like we did in the lab. Hence, I decided to measure the time taken to loop through each array 10 times, and finding the average. This should mitigate the effects of any outliers, and all array sizes would be tested in similar conditions.

**Conclusion:**As mentioned in the above section, timings remain flat up to 64MiB (indicating cache hits), and jumps considerably at 128MiB (indicating cache misses), hence we can infer a usable LLC of 64 – 96 MiB for this CPU. This matches the lscpu output showing 96MiB L3 cache. This shows that our algorithm successfully differentiates cache hits (3-5ms) from cache misses (>12ms)