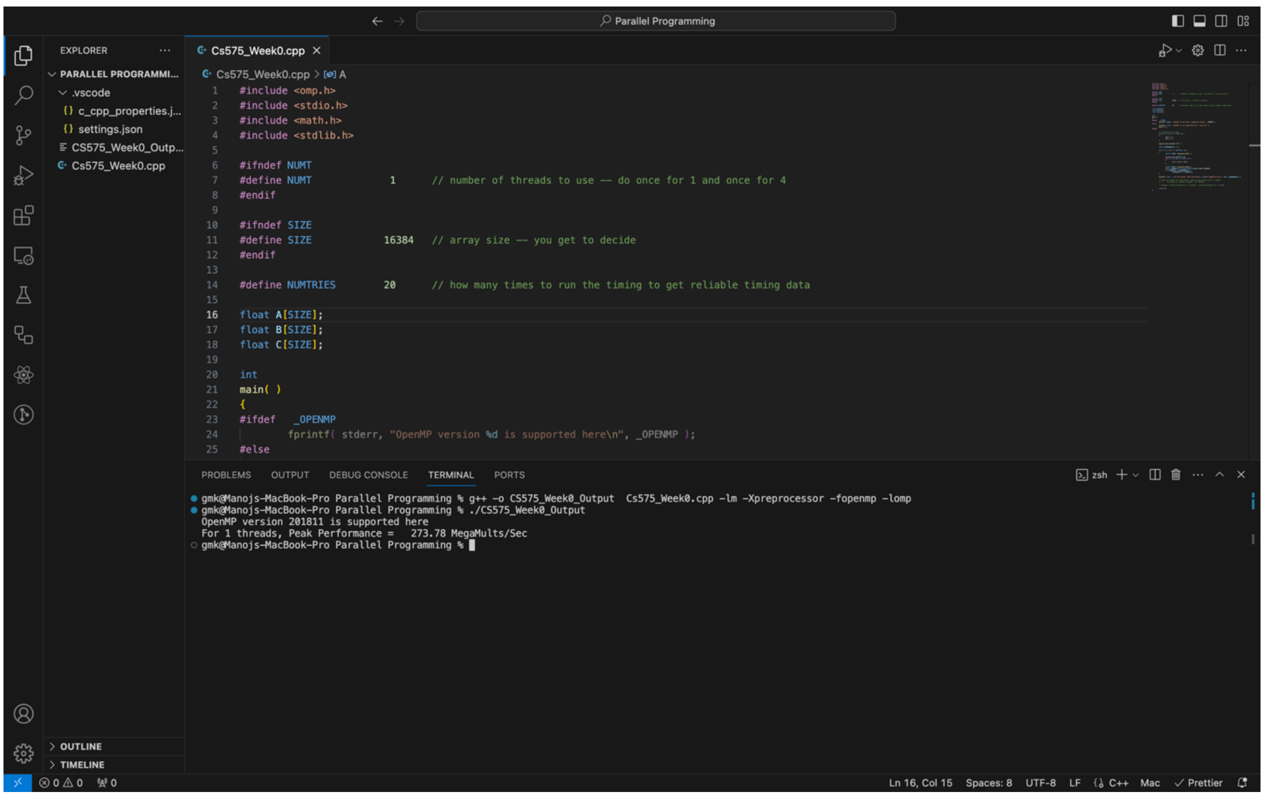
PROJECT #0
Simple OpenMP Experiment

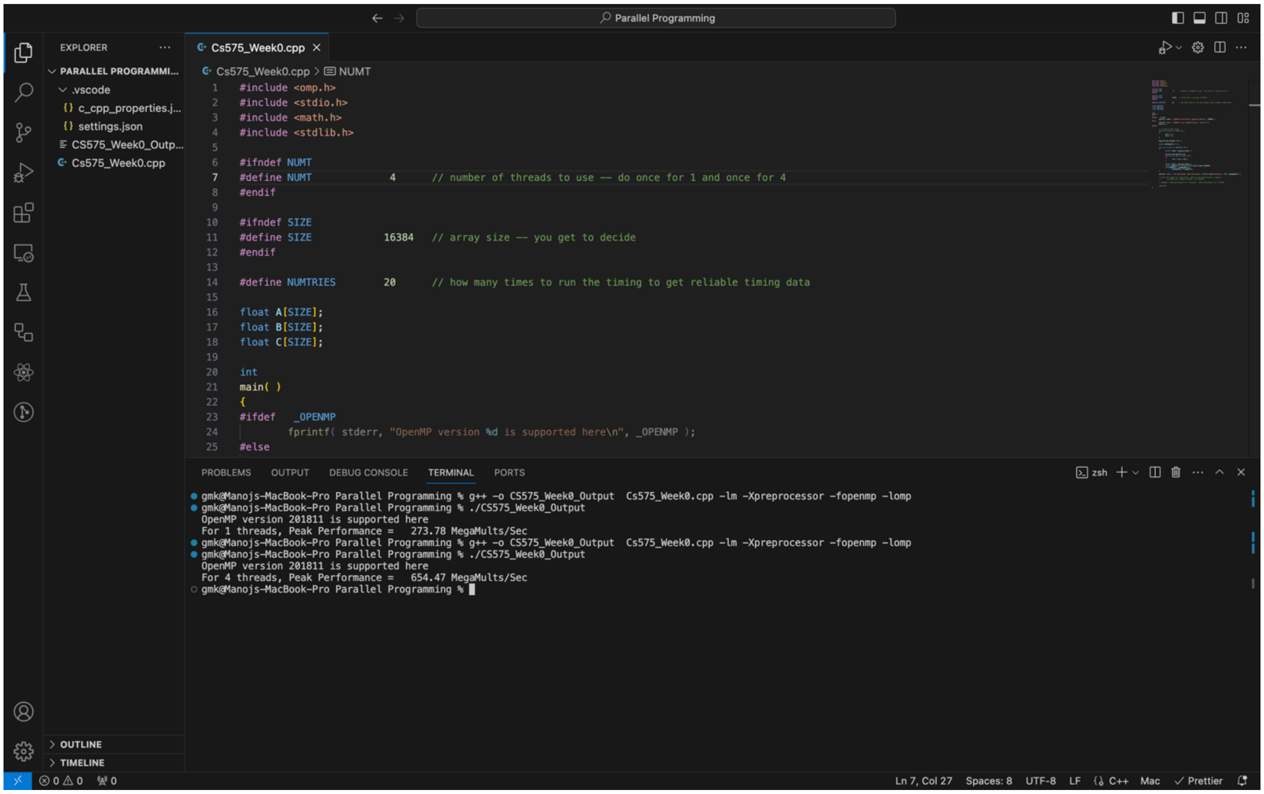
Name: Manoj Kumar Gummadi
Email: gummadm@oregonstate.edu
Date: 08/04/2024

1) Performance or Execution time results for 1 thread



Ans: 273.78 MegaMults/Sec

2) **Performance or Execution time results for 4 threads**



Ans: 654.47 MegaMults/Sec

3) **One-thread-to-four-threads Speedup (>1.)?**

Ans: (Peak performance for 4 threads) / (Peak performance for 1 thread)

Speedup (S) = 654.47/273.78 => 2.39

4) **Parallel Fraction (Fp)?**

Ans: (4.0/3.0) * (1.0 – (1.0/S)) = (4.0/3.0) * (1.0 – (1.0/2.39))
= 1.333 - 0.582 = 0.751

5) **Commentary**

a) **Tell what machine you ran this on.**

Apple M3 Macbook **Pro**, OS: Sanoma 14.4.1

b) **What performance results did you get?**

1 thread - 273.78 MegaMults/Sec

4 threads - 654.47 MegaMults/Sec

c) **What was your 1-thread-to-4-thread speedup?**

S = 2.39

d) **Your 1-thread to 4-thread speedup should be less than 4.0. Why?**

Managing multiple threads incurs overhead due to setup, synchronization, and teardown, diminishing the benefits of parallelization. Additionally, Amdahl's Law dictates that some parts of a program must run sequentially, limiting the overall speedup achievable through parallelization and resulting in less than linear improvement. Furthermore, when threads compete for shared resources like CPU caches and memory bandwidth, it can lead to contention, causing delays and reducing overall efficiency. Ensuring data consistency among threads through synchronization mechanisms such as locks and barriers introduces additional processing overhead, detracting from the desired speedup. Moreover, load balancing challenges arise when tasks are unevenly distributed among threads, leading to inefficient resource utilization and hindering optimal parallel execution.

e) **What was your Parallel Fraction, Fp?**

Ans: 0.751