**Course:** High Performance Computing Lab

**Practical No. 3**

PRN: 22510057

Name: Ashutosh Gundu Birje

Batch: B8

**Title of practical:**

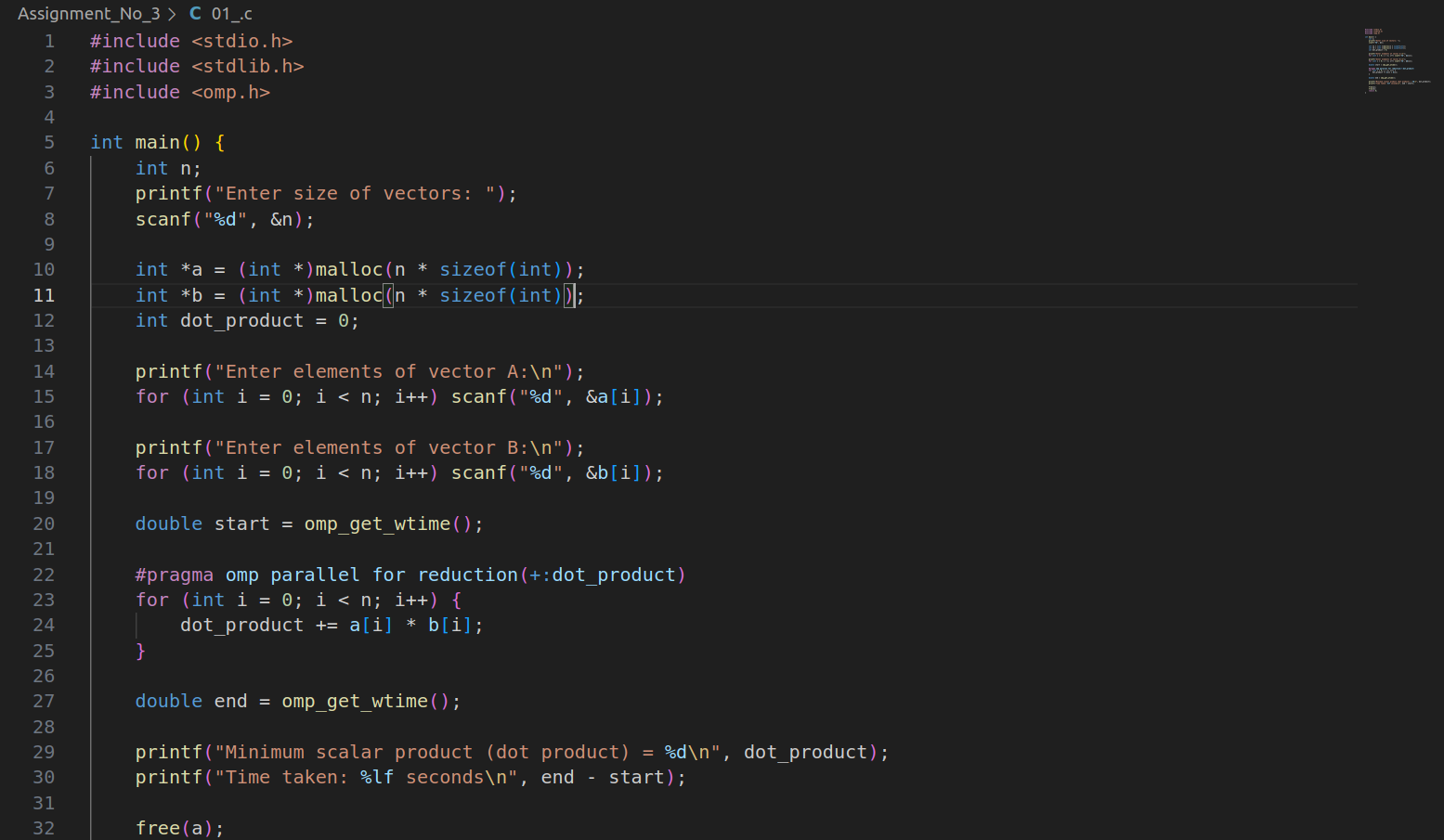
Study and Implementation of schedule, nowait, reduction, ordered and collapse clauses

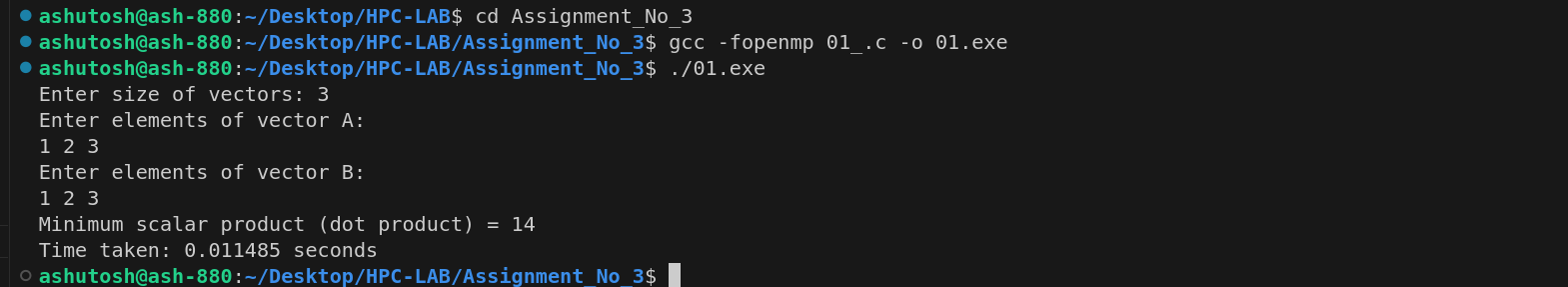
**Problem Statement 1:**

Analyse and implement a Parallel code for below program using OpenMP.

// C Program to find the minimum scalar product of two vectors (dot product)

**Screenshots:**



****

**Information and analysis:**

Approach**: Parallelized dot product using reduction to avoid race conditions.**

* Time complexity: O(n), but runtime improves with threading.
* Analysis: Performance improves with vector size and higher threads. Reduction clause helps in combining partial results efficiently.

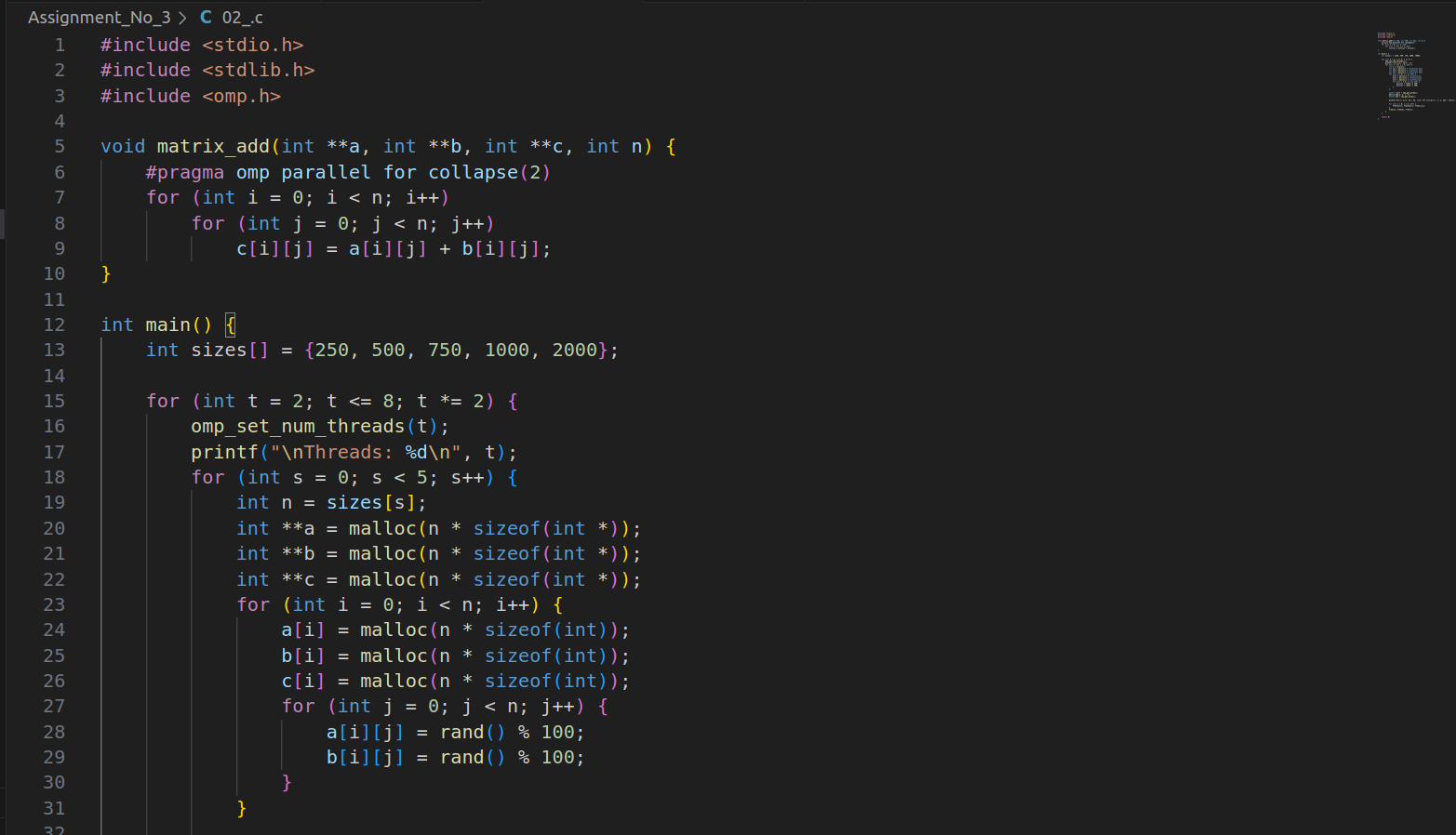
**Problem Statement 2:**

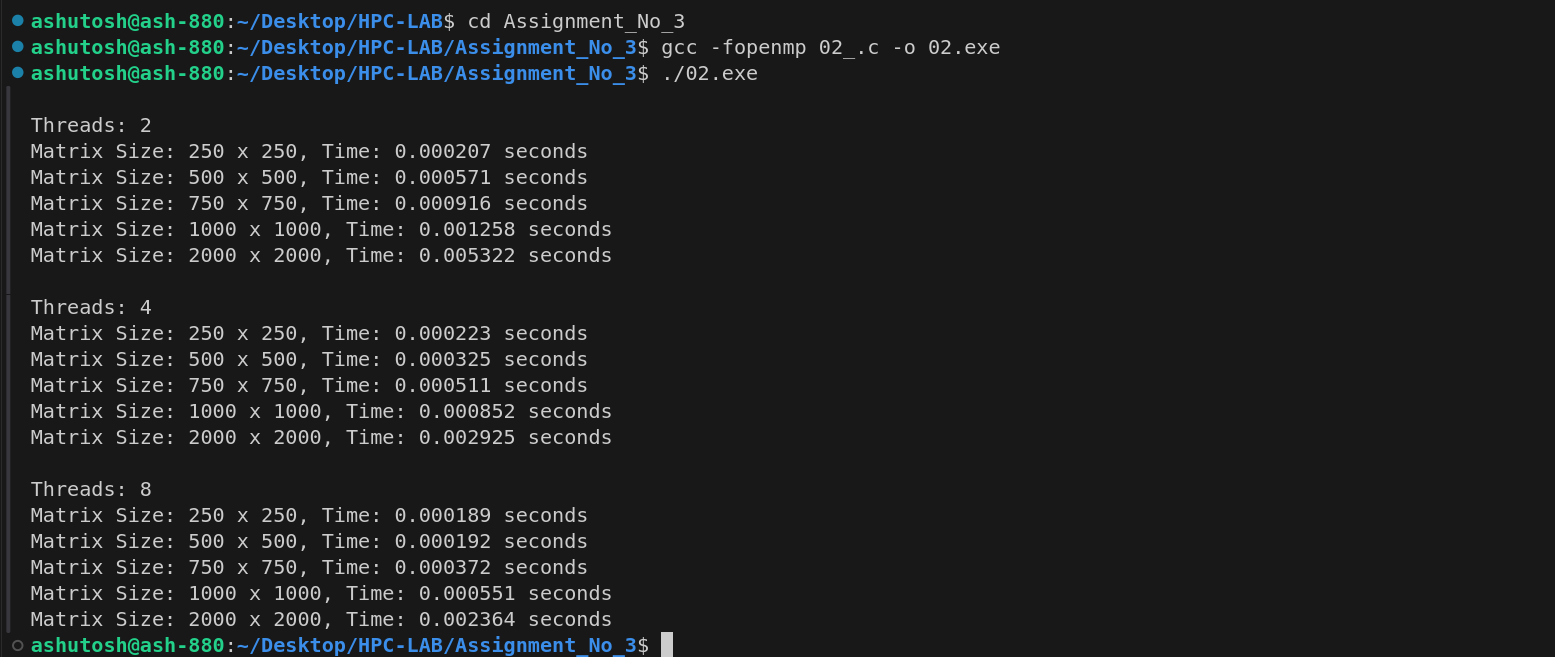
Write OpenMP code for two 2D Matrix addition, vary the size of your matrices from 250, 500, 750, 1000, and 2000 and measure the runtime with one thread (Use functions in C in calculate the execution time or use GPROF)

i. For each matrix size, change the number of threads from 2,4,8., and plot the speedup versus the number of threads.

ii. Explain whether or not the scaling behaviour is as expected.

**Screenshots:**



****

**Information and analysis:**

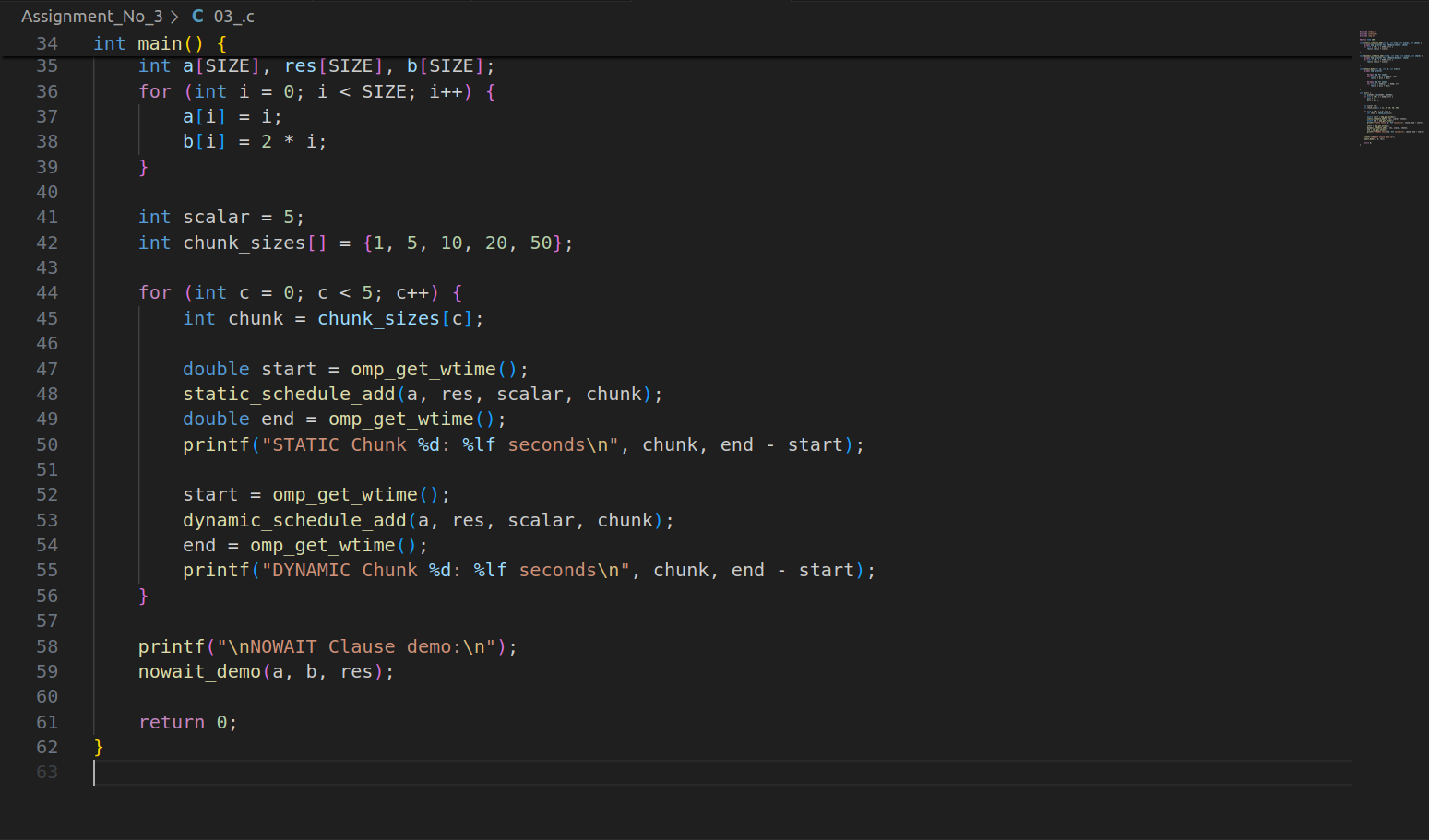
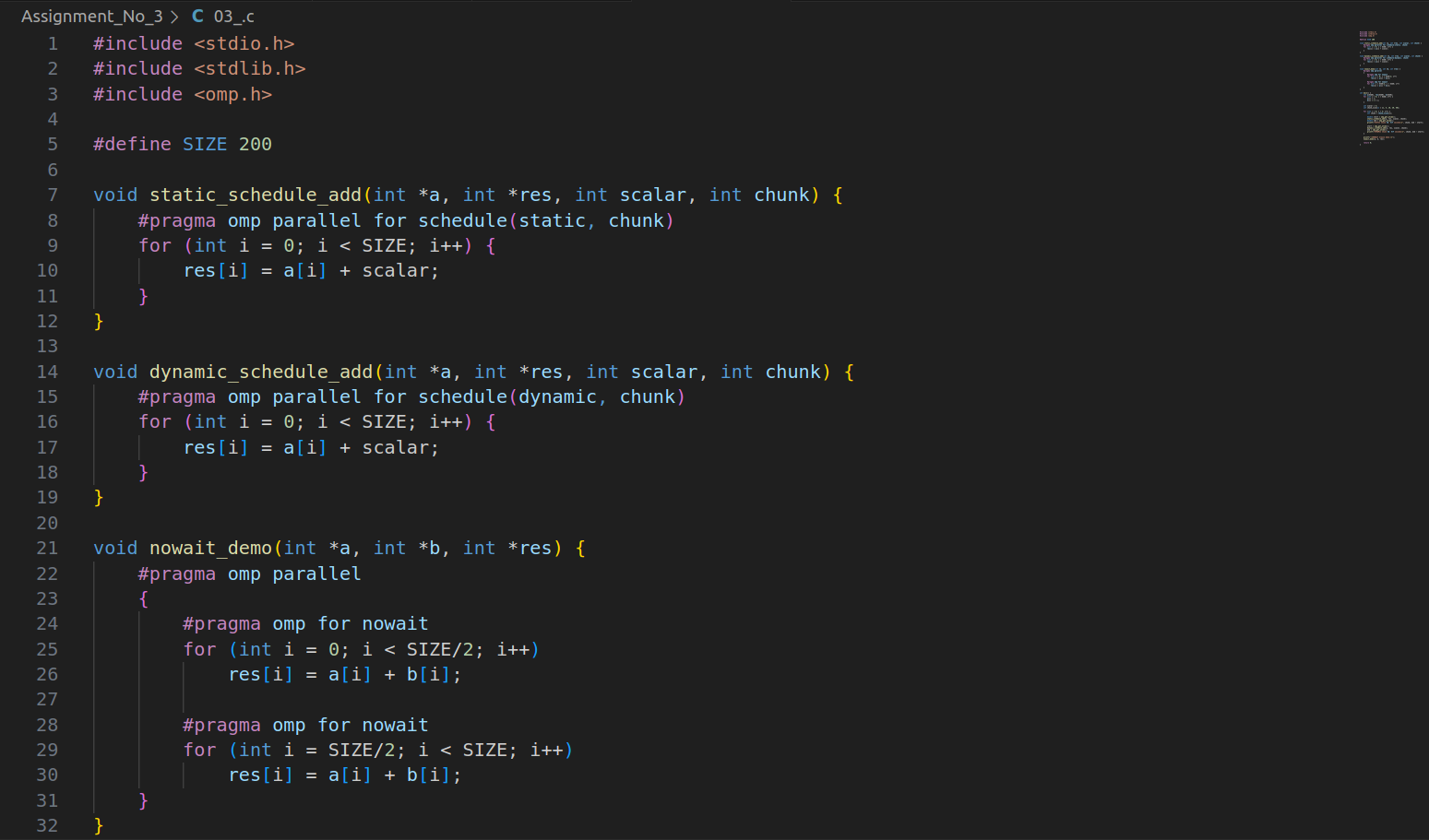
**collapse(2) allows flattening of nested loops.**

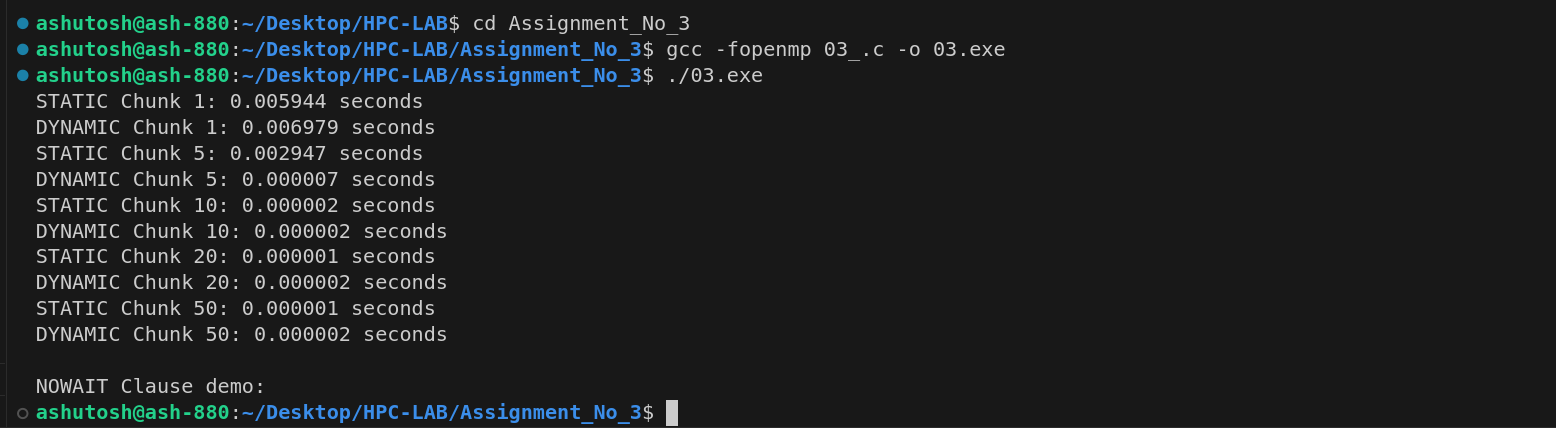
* Expected Behavior: Speedup improves with larger matrix size and thread count, up to a limit.
* Scaling Limit: Diminishing returns beyond 4-8 threads due to memory/cache bottlenecks.

**Problem Statement 3:**

For 1D Vector (size=200) and scalar addition, Write a OpenMP code with the following: i. Use STATIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup. ii. Use DYNAMIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup. iii. Demonstrate the use of nowait clause.

**Screenshots:**

****

**Information and analysis:**

STATIC Schedule**: Best when iteration workload is uniform.**

* DYNAMIC Schedule: Helps when workload is unpredictable or uneven.
* Chunk Size: Impacts load balancing and thread overhead.
* Nowait Clause: Prevents implicit barrier, useful for independent tasks.

**Github Link:**