**Class:** Final Year B.Tech(Computer Science and Engineering)

**Year:** 2025-26 **Semester:** 1

**Course:** High Performance Computing Lab

**Practical No. 2**

**Exam Seat No:22510029**

**Title of practical: Study and implementation of basic OpenMP clauses**

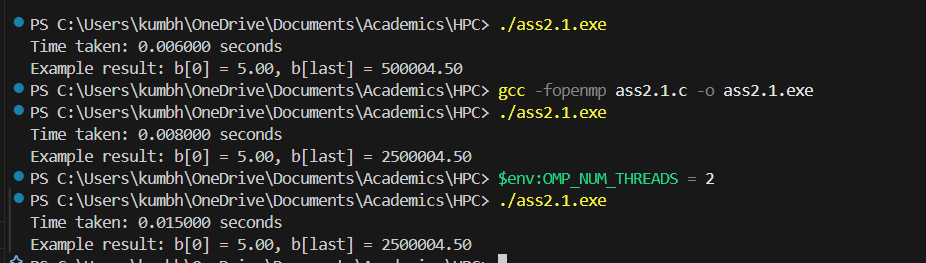
Implement following Programs using OpenMP with C:

1. Vector Scalar Addition
2. Calculation of value of Pi

Analyse the performance of your programs for different number of threads and Data size.

**Problem Statement 1:**

**Screenshots:**

****

**Information:**

Task: Add a scalar value to each element of a large vector.

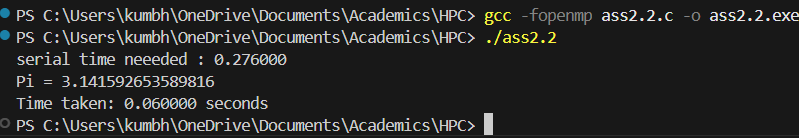
* Data Size: 1,000,000 elements.
* Scalar Value: 5.0
* Threads: Execution tested with different thread counts using OpenMP.
* Compiler & Flags: gcc -fopenmp
* Timer: omp\_get\_wtime() used for performance measurement.

**Analysis:**

* parallel execution generally decreases the execution time, although the speedup might be minimal for small data sizes due to the overhead of creating and managing threads.
* **memory-bound**, meaning its performance is limited more by the memory bandwidth (how quickly data can be moved from RAM to the CPU) rather than the CPU's raw processing speed. This is because the operation on each element is simple, but it requires a lot of data to be read and written from memory.Vector scalar addition is memory-bound; thus, performance is influenced by memory bandwidth more than CPU compute speed.

**Problem Statement 2:**

**Screenshots:**

****

**Information:**

* Task: Approximate π using numerical integration.
* Steps: 100,000,000 iterations.
* Threads: Execution tested using OpenMP with default threads.
* Compiler & Flags: gcc -fopenmp
* Timer: omp\_get\_wtime() used for performance measurement.

**Analysis:**

* CPU-bound, as it involves a large number of floating-point operations rather than extensive data movement. Because of this, dividing the computational work among multiple threads can lead to a significant decrease in execution time. The performance will scale with the number of available CPU cores until the point where the overhead of thread management outweighs the benefits of parallelization

**Github Link: https://github.com/Anjali1874/HPC-Lab**