**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**

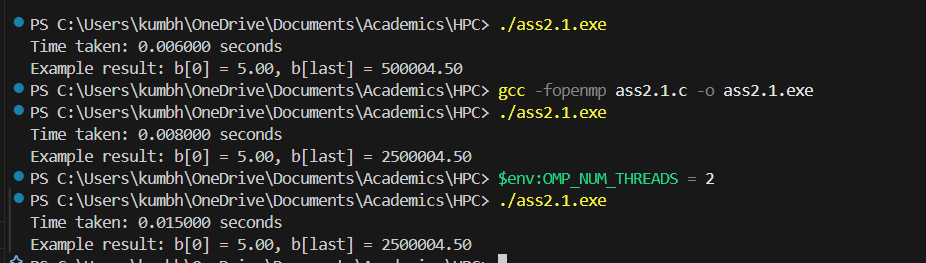
Implement following Programs using OpenMP with C:

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.

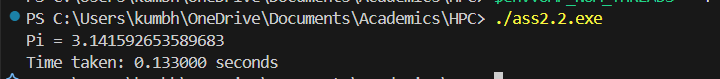
* Data Size: 1,000,000 elements.
* Scalar Value: 5.0
* Threads: Execution tested with different thread counts using OpenMP.
* Compiler & Flags: gcc -fopenmp
* Timer: omp\_get\_wtime() used for performance measurement.

**Analysis:**

* The program showed a decrease in execution time with parallel threads, although for small data sizes, speedup may be minimal due to thread creation overhead.
* When OMP\_NUM\_THREADS was set to 2, the time slightly changed compared to default (possibly due to OS scheduling and CPU load).
* 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:**

* **The result for π was accurate up to several decimal places (3.141592653589683).**
* **Execution time decreases with parallel execution due to loop division among threads.**
* **The task is CPU-bound (heavy floating-point operations), so increasing threads can significantly improve performance until limited by CPU cores.**

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