Assignment No 1.

Name: Manaswi Mahesh Devekar

PRN: 22510035

High Performance Computing Lab

Batch: B-1

Title: Introduction to OpenMP.

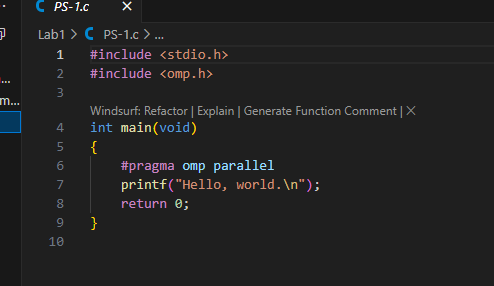
GitHub Link

**What is OpenMP?**

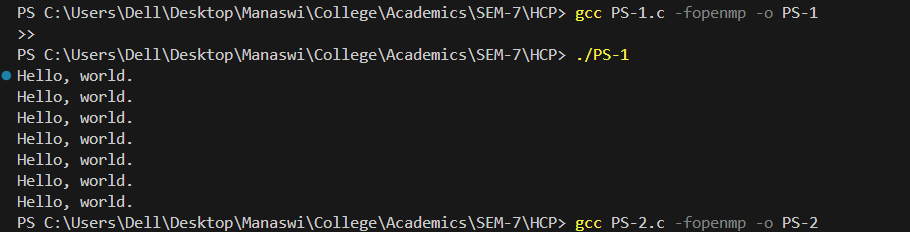
OpenMP (Open Multi-Processing) is an API that enables shared-memory multiprocessing programming in C, C++, and Fortran across multiple operating systems. It provides a portable and scalable approach to parallel programming, offering developers a simple yet powerful interface for building high-performance applications. Whether you're targeting a standard desktop or a large-scale supercomputer, OpenMP makes it easier to harness the power of multiple processor cores efficiently.

1. Demonstrate Installation and Running of OpenMP Code in C

Code:

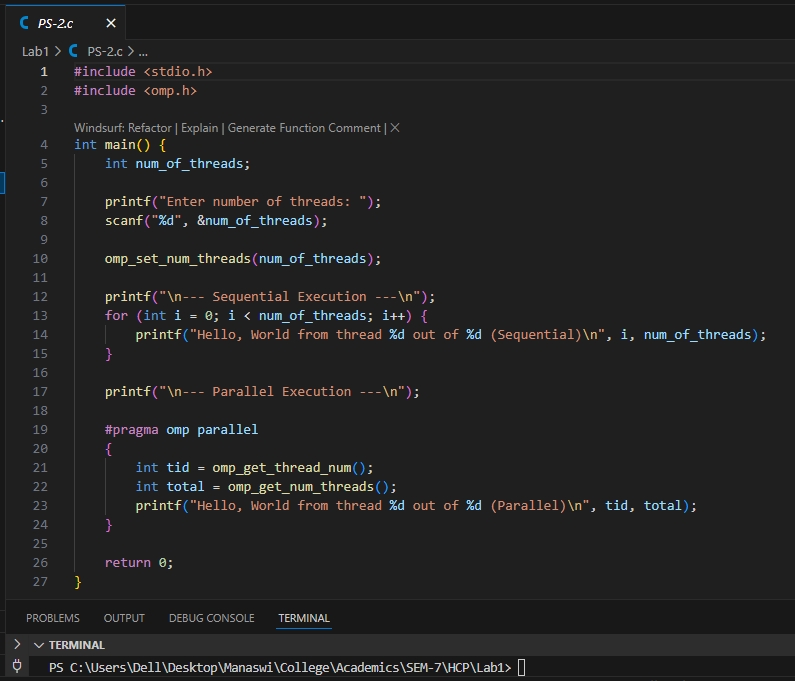


Output:

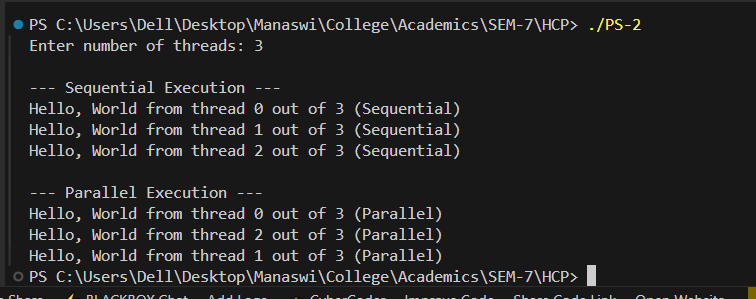


2). Print `Hello, World! ` in sequential and parallel in OpenMP

Code:



Output:



Analysis:

* In **sequential execution**, messages are printed one by one in order from thread 0 to num\_threads - 1.
* In **parallel execution**, multiple threads run concurrently.

3). Calculate Theoretical FLOPS of your system. Elaborate Parameters and Show Calculation.

|  |  |  |
| --- | --- | --- |
| Parameter | Symbol | Significance |
| Clock Speed |  | The number of CPU cycles per second (in Hz). Higher clock speed means more operations per second. |
| Number of Cores |  | The number of **physical CPU cores**. More cores enable more operations to be executed in parallel. |
| SIMD Width / Factor |  | Number of floating-point values processed simultaneously by one vector instruction (based on SIMD register width). |
| FMA Factor |  | Accounts for **Fused Multiply-Add** instructions. If supported, each instruction counts as **2 FLOPs** (1 multiply + 1 add). |
| Superscalar Factor |  | Indicates how many SIMD instructions a core can issue per clock cycle (instruction throughput). Modern CPUs may issue 1-2 per cycle. |