Course: High Performance Computing Lab

Practical No 1

Name:Suyash Vasantrao Yadav

Batch:B1

PRN:22510070

Title: Introduction to OpenMP

Problem Statement 1 – Demonstrate Installation and Running of OpenMP code in C

Recommended Linux based System:

Following steps are for windows:

OpenMP – Open Multi-Processing is an API that supports multi-platform shared-memory multiprocessing programming in C, C++ and Fortran on multiple OS. OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms ranging from the standard desktop computer to the supercomputer.

To set up OpenMP,

We need to first install C, C++ compiler if not already done. This is possible through the MinGW Installer.  
Reference: Article on GCC and G++ installer ([Link](https://www.scaler.com/topics/c/c-compiler-for-windows/))

Note: Also install `mingw32-pthreads-w32` package.

Then, to run a program in OpenMP, we have to pass a flag `-fopenmp`.

Example:

To run a basic Hello World,

*#include* <stdio.h>

*#include* <omp.h>

*int* main(*void*)

{

*#pragma* *omp* *parallel*

    printf("Hello, world.\n");

*return* 0;

}

gcc -fopenmp test.c -o hello

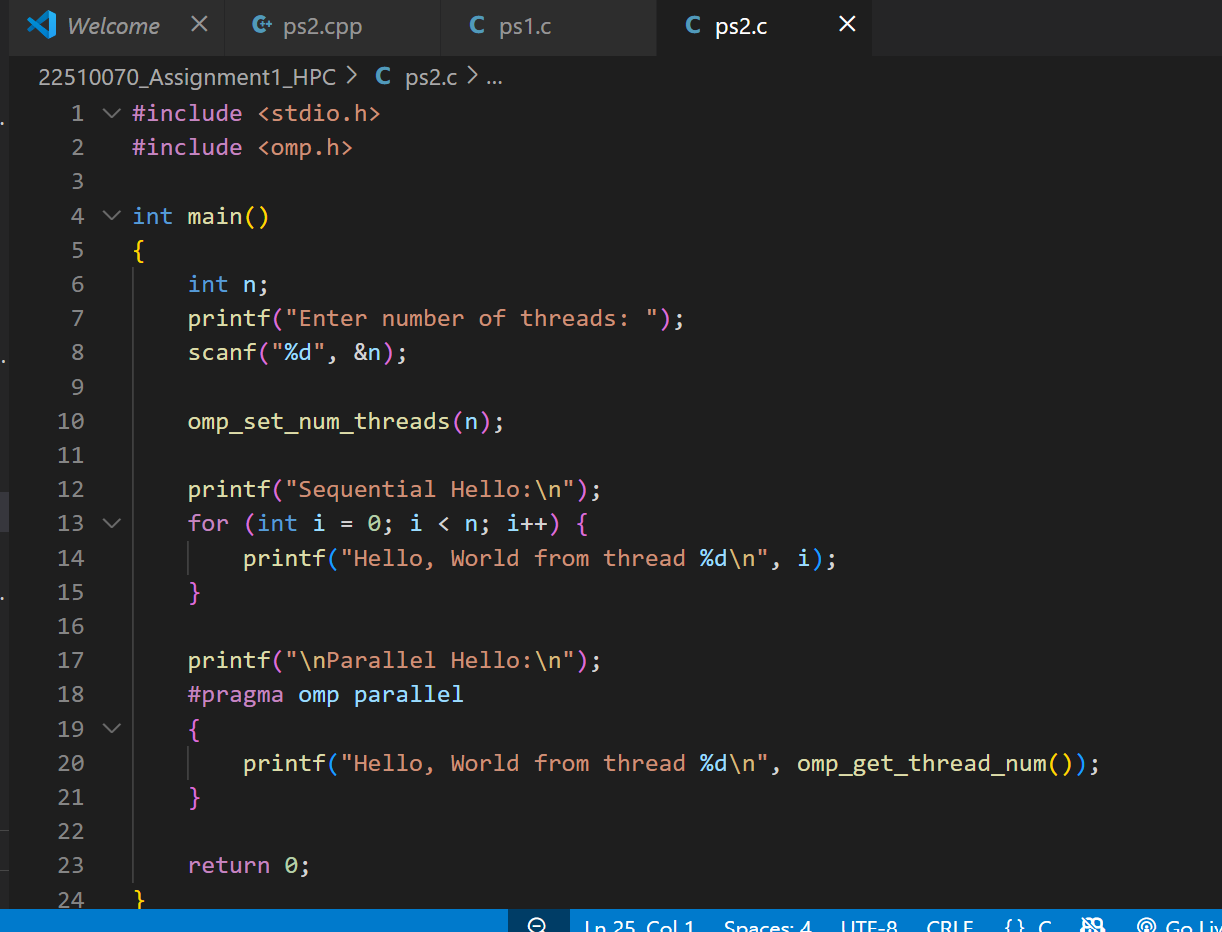
.\hello.exe



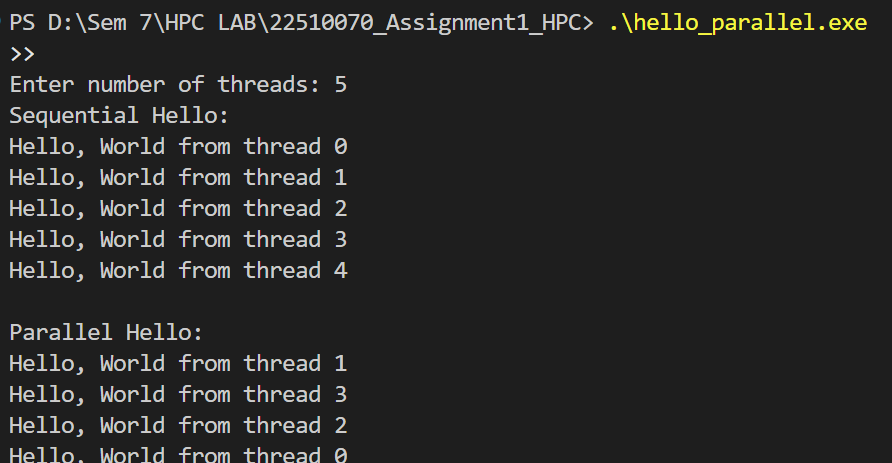
Problem Statement 2 – Print ‘Hello, World’ in Sequential and Parallel in OpenMP

We first ask the user for number of threads – OpenMP allows to set the threads at runtime. Then, we print the Hello, World in sequential – number of times of threads count and then run the code in parallel in each thread.

Code snapshot:



Output snapshot:



Analysis:

* Sequential Execution prints messages in fixed order using a loop.
* Parallel Execution uses OpenMP’s #pragma omp parallel, and each thread prints independently.
* The thread number is obtained using omp\_get\_thread\_num().

GitHub Link: https://github.com/Suyashyadav07/HPC

Problem statement 3: Calculate theoretical FLOPS of your system on which you are running the above codes.

**Calculating FLOPS**

**1. Get CPU Details**

Open **Task Manager** → **Performance tab** → **CPU**:

* **Cores:** (e.g., 4)
* **Clock Speed:** (e.g., 2.4 GHz)

**2. Use Formula**

FLOPS=Cores×Clock speed×FLOPs/cycle

Assuming:

* Cores = 4
* Clock = 2.4 GHz = 2.4 × 10⁹
* FLOPs/cycle = 8

FLOPS=4×2.4×109×8=76.8 GFlops