Course: High Performance Computing Lab

Practical No 1

PRN: 22510092

Name:Vrushali Sangale

Batch:B1

Title: Introduction to OpenMP

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

Recommended Linux based System:

| Action | Command |
| --- | --- |
| Install GCC | sudo apt install build-essential |
| Create code | nano hello\_omp.c |
| Compile with OpenMP | gcc -fopenmp hello\_omp.c -o hello\_omp |
| Run | ./hello\_omp |
| Set thread count | export OMP\_NUM\_THREADS=4 |

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

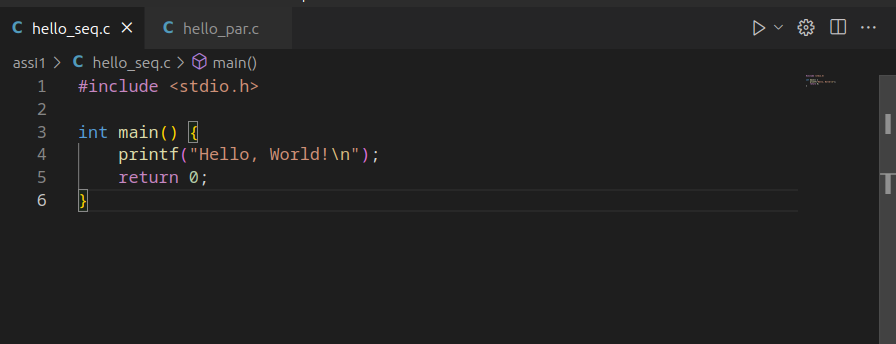
.\hello.exe

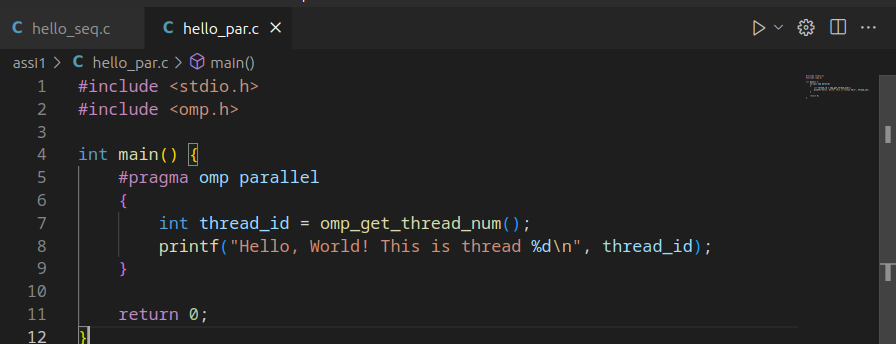


Problem Statement 2 – Print ‘Hello, World’ in Sequential and Parallel in OpenMP

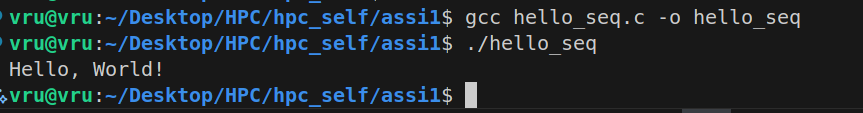
We first ask the user for number of threads – OpenMP allows to set the threads at runtime. Then, we print the Hello, World in sequential – number of times of threads count and then run the code in parallel in each thread.

Code snapshot:





Output snapshot:





Analysis: multiple threads run the same code in the same amout of time it takes to run in sequential

GitHub Link: **https://github.com/Vru01/HPC\_22510092.git**

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

To calculate the theoretical FLOPS (Floating Point Operations Per Second) of your system, you need to consider the following parameters:

1. Clock Speed (in Hz): The frequency at which your CPU operates.

2. Number of Cores: The total number of cores in your CPU.

3. Instructions per Cycle (IPC): The average number of instructions executed per clock cycle.

4. Floating Point Operations per Instruction (FLOPI): The average number of floating-point operations performed per instruction.

The formula to calculate theoretical FLOPS is:

FLOPS = Clock Speed \* Number of Cores \* IPC \* FLOPI

We can obtain the clock speed, number of cores, and IPC from your CPU specifications. The FLOPI value depends on the specific instructions used in code.

