Course: High Performance Computing Lab

Practical No 1

PRN: 22510012

Name: Jai Chowgule

Batch: B6

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,

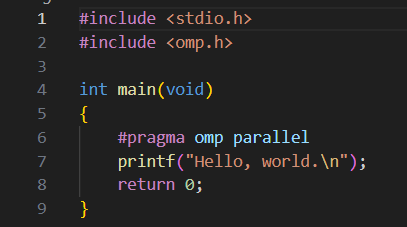
We need to first install C, C++ compiler if not already done. This is possible through the MinGW Installer.  
Reference: Article on GCC and G++ installer ([Link](https://www.scaler.com/topics/c/c-compiler-for-windows/))

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,



gcc -fopenmp test.c -o hello

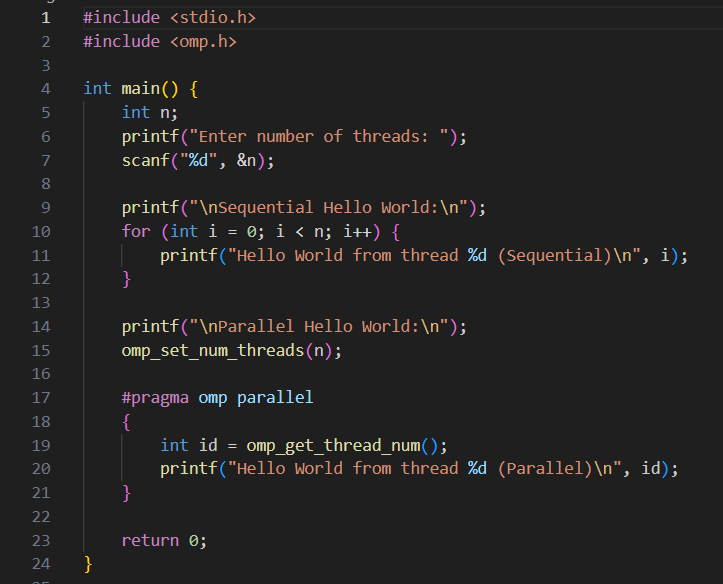
.\hello.exe



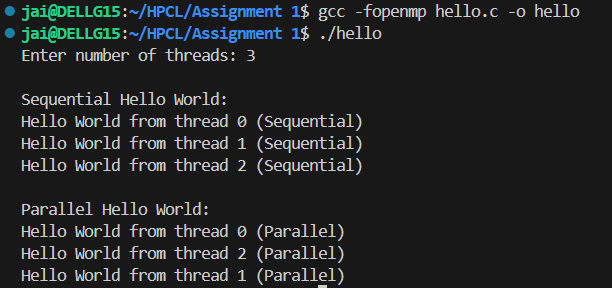
Problem Statement 2 – Print ‘Hello, World’ in Sequential and Parallel in OpenMP

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:

1. In Sequential execution, the order of the iternation is predictable, it happens one after the other.
2. In Parallel executtion, the print statement is executed as per the threads and when they are scheduled. Thus is execution order is not fixed.
3. Sequential execution happens in O(N) whereas Parallel happens in nearly O(1) if the threads are enough.

GitHub Link:

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

Elaborate the parameters and show calculation.

Ans:

Processor – 11th Gen Intel® Core™ i5-11260H

TheoreticalFLOPS=Number of Cores×Clock Speed (Hz)×Operations  per Clock Cycle per Core

Number of cores - 6

Highest clock speed - 4.4 GHz

Operation per clock cycle per hz - 32

TheroticalFLOPS = 6 x 4.4 x 10^9 x 32

= 0.845 GFLOPS

NanoReview(not Official) = ~0.4 TFLOPS