Course: High Performance Computing Lab

Practical No 1

PRN: 23520001

Name: Aishwarya M. Suryawanshi

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,

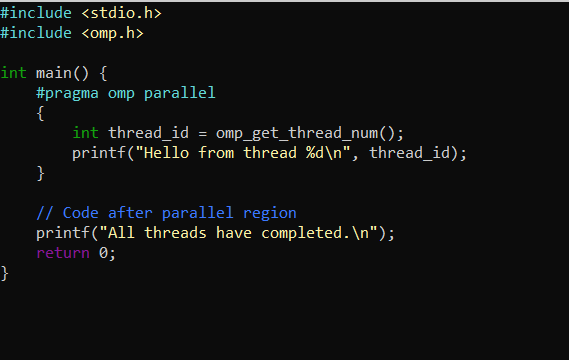
We need to first install C, C++ compiler if not already done. This is possible through the MinGW Installer.  
Reference: Article on GCC and G++ installer ([Link](https://www.scaler.com/topics/c/c-compiler-for-windows/))

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

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

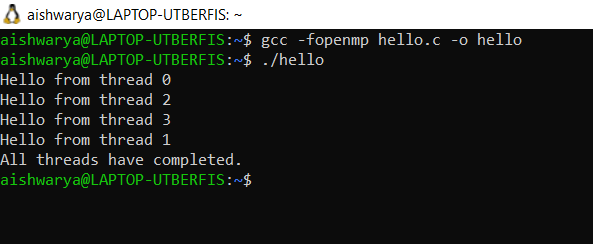
Example:

To run a basic Hello World,



gcc -fopenmp test.c -o hello

.\hello.exe

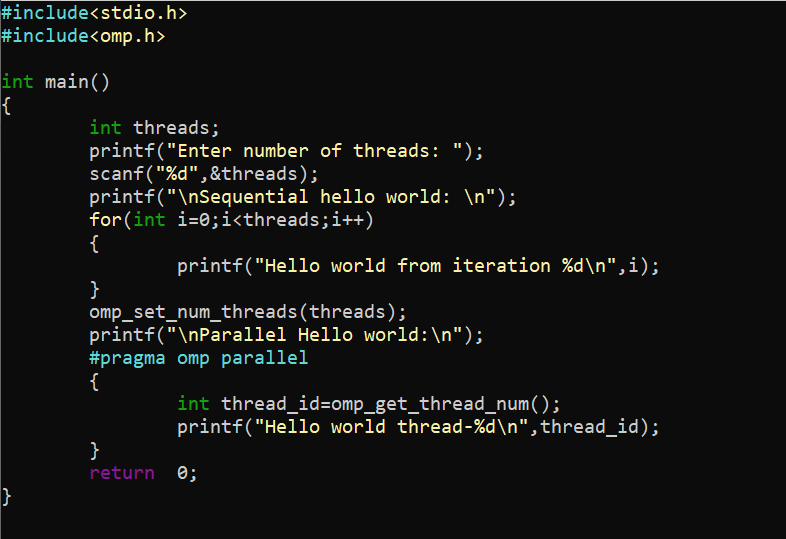


As openmp by default sets the number of threads equal to number of logical processors in system. Actually the cores are two and logical processors = 4.

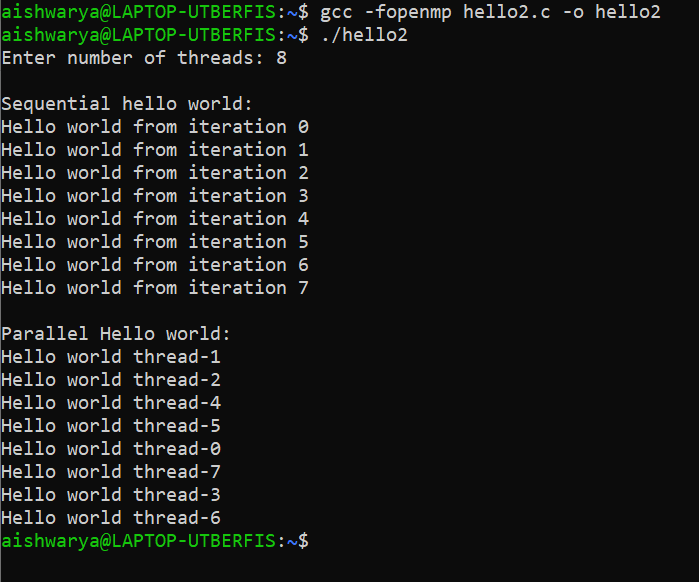
Problem Statement 2 – Print ‘Hello, World’ in Sequential and Parallel in OpenMP

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:

Number of threads are taken from user and in openmp parallel threads count is set by omp\_set\_num\_threads

While executing sequential hello world the program is executed by main thread and executes line by line the number of thread is sequential on each line.

While threads are executed in parallel multiple threads works independently here sequential printing is not there as concurrent execution is there.

GitHub Link: <https://github.com/TechieAishwaryaa/HPC_Practical_1.git>

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

Elaborate the parameters and show calculation.

FLOPS (Floating point operations per second)

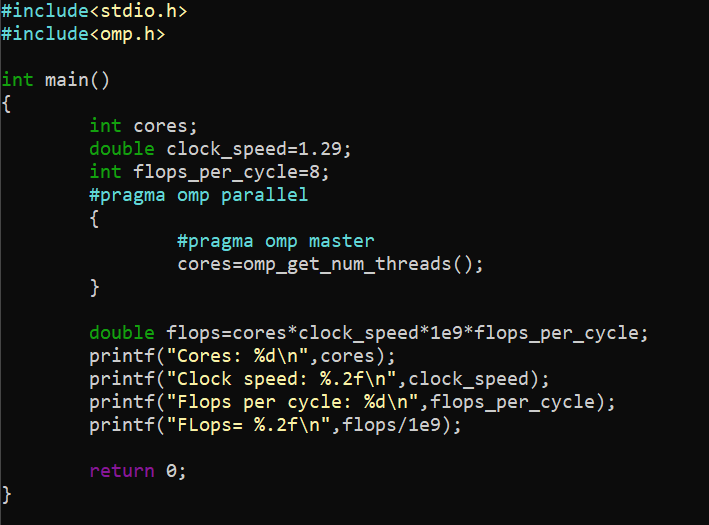
It measures how many floating point calculations processor can theoretically perform every second

Formula for theoretical FLOPS:

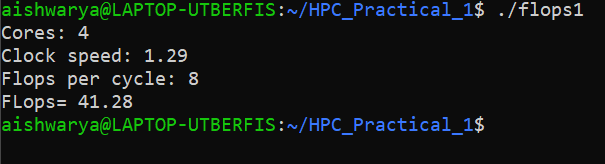
Flops = number of CPU cores \* clock speed \* flops per cycle

* Cores: number of physical or logical cores used
* Clock speed: measured in GHz convert it to Hz by multiplying by 109
* Flops per cycle: depends on CPU architecture. Modern computer- 8 to 32 cycles per core

Code snapshot:



Output snapshot:



Analysis

After flops calculations using the formula the result is shown on screen.