Course: High Performance Computing Lab

Practical No 1

PRN: 2020BTECS00010

Name: Abhijeet Kamalekar

Batch: B8

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*

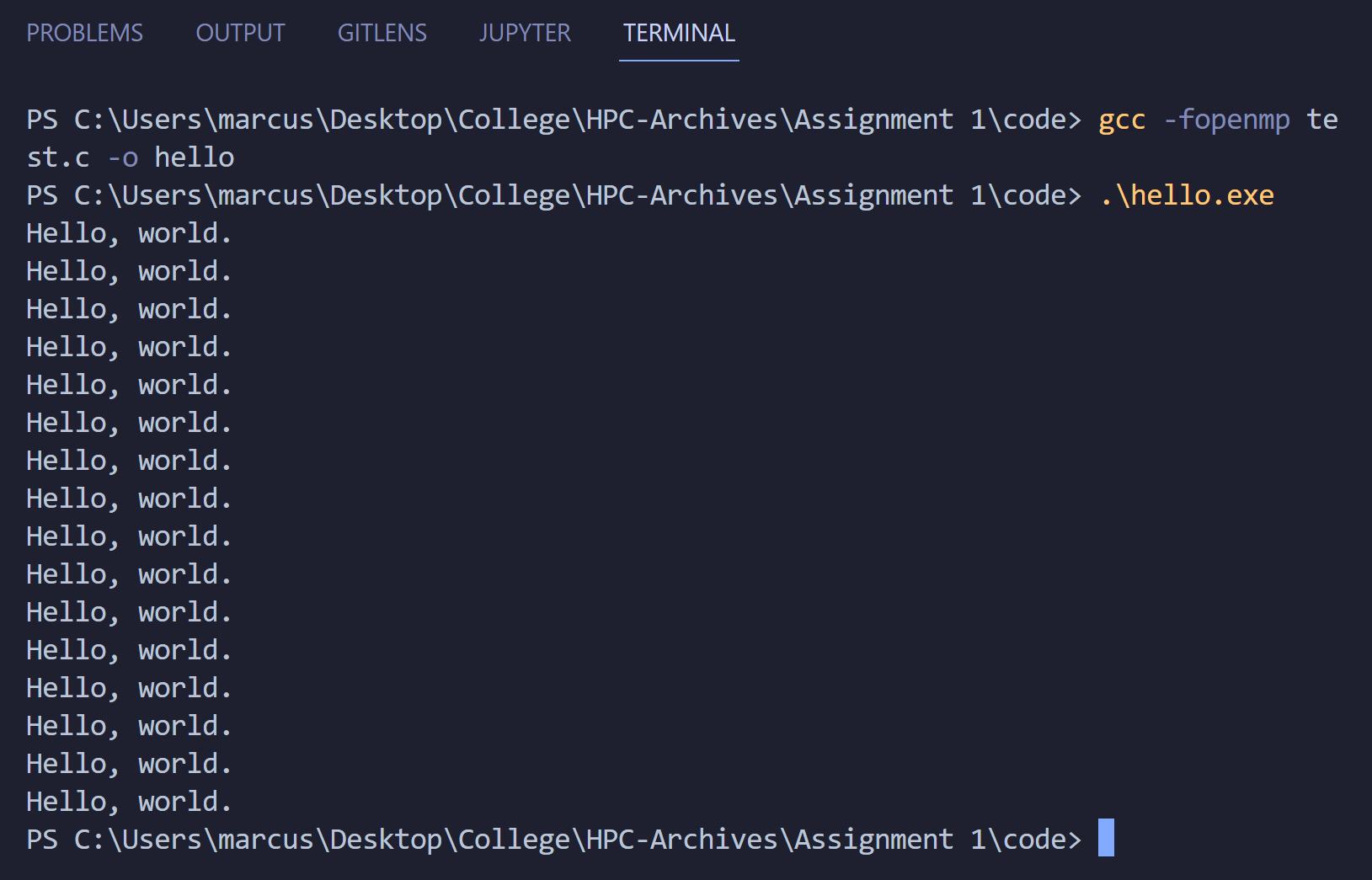
    printf("Hello, world.\n");

*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:

#include <omp.h>

#include <stdio.h>

int main() {

    int num\_threads;

    printf("Enter the number of threads: ");

    scanf("%d", &num\_threads);

    printf("\nSequential execution:\n");

    for (int i = 0; i < num\_threads; i++) {

        printf("Hello, World from sequential execution %d\n", i+1);

    }

    omp\_set\_num\_threads(num\_threads);

    printf("\nParallel execution:\n");

    #pragma omp parallel

    {

        int thread\_id = omp\_get\_thread\_num();

        printf("Hello, World from parallel execution by thread %d\n", thread\_id);

    }

    return 0;

}

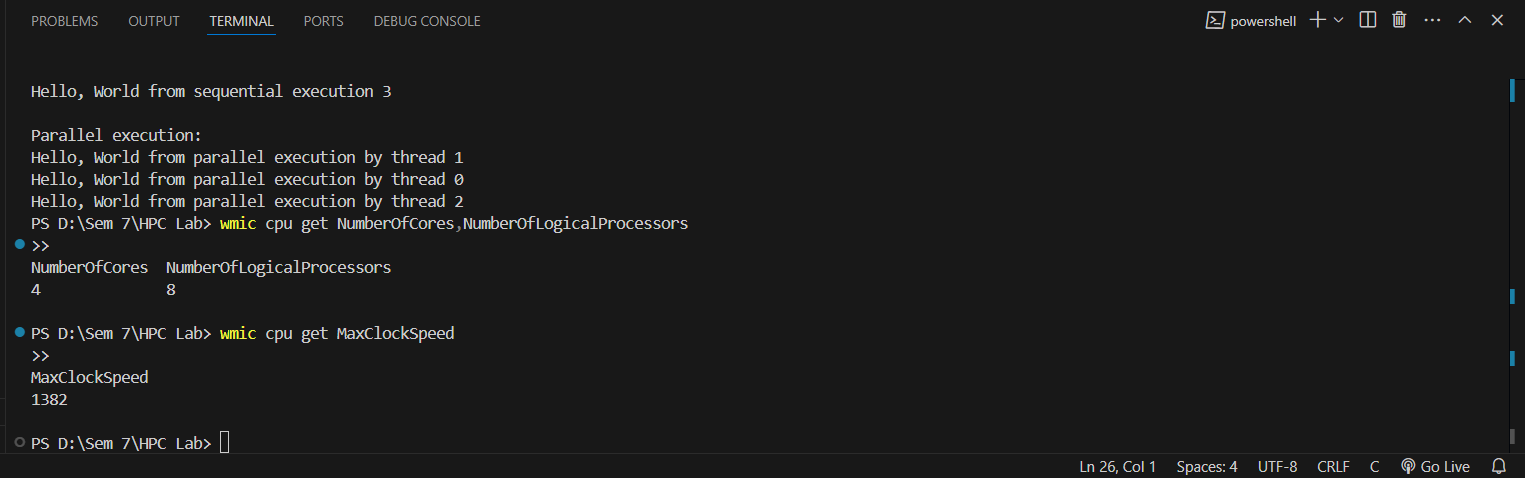
Output snapshot:



Analysis:

GitHub Link: [https://github.com/AbhijeetKamalekar15/HPC-Lab.git](%20https:/github.com/AbhijeetKamalekar15/HPC-Lab.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 = 1.382 GHz × 8 threads × 8 FLOPs/cycle = 88.448 GFLOPS**