Course: High Performance Computing Lab

Practical No 1

PRN: 21510043

Name: Shweta Avinash Kakade

Batch: B-2

Title: Introduction to OpenMP

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

Recommended Linux based System:

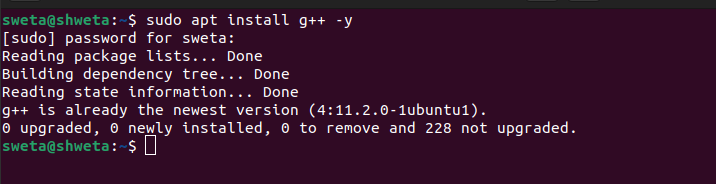
Following steps are for windows:

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*

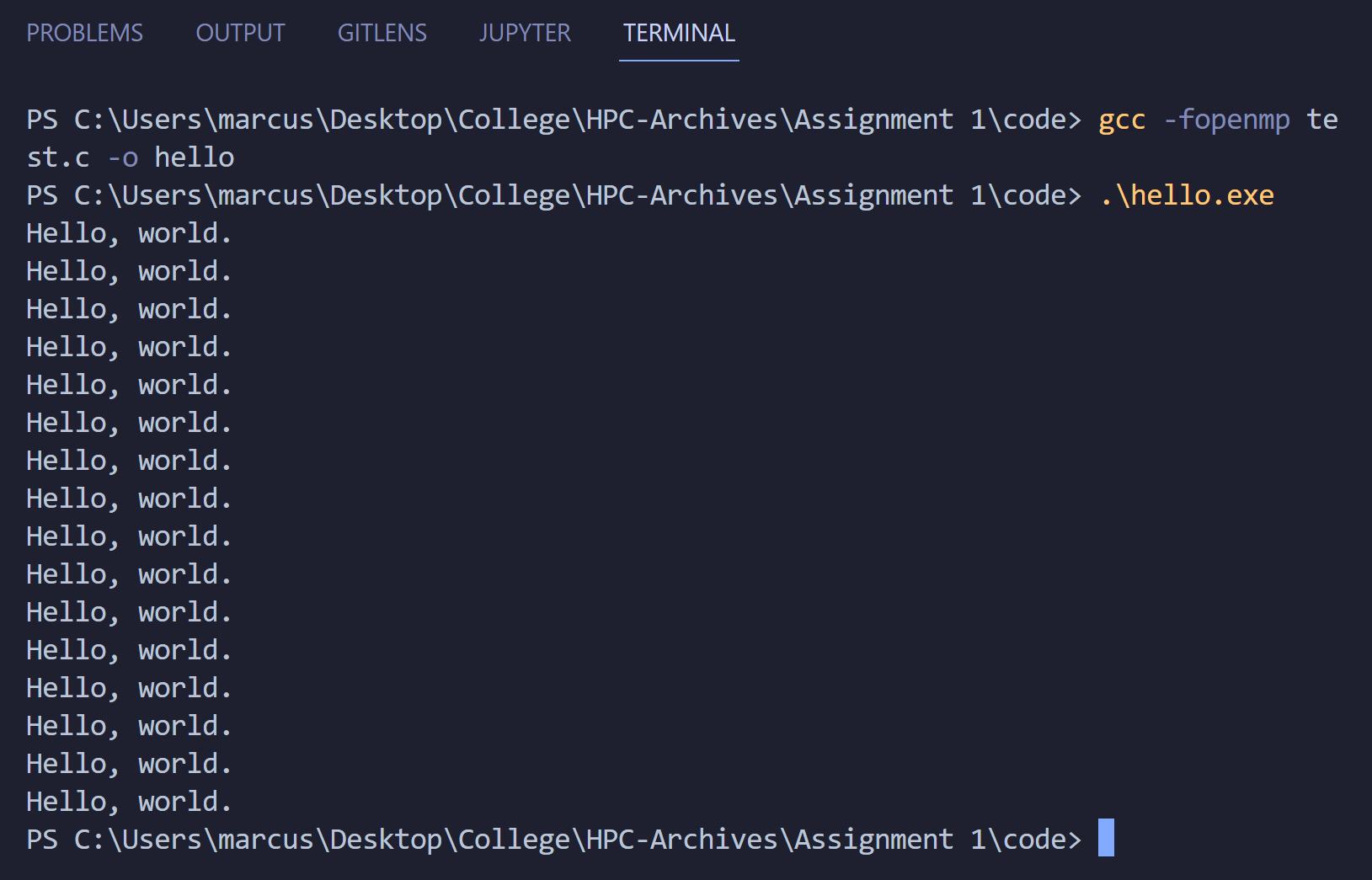
    printf("Hello, world.\n");

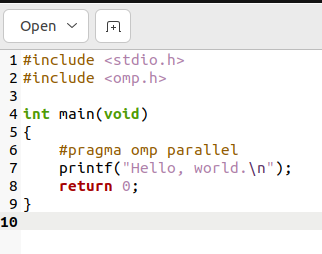
*return* 0;

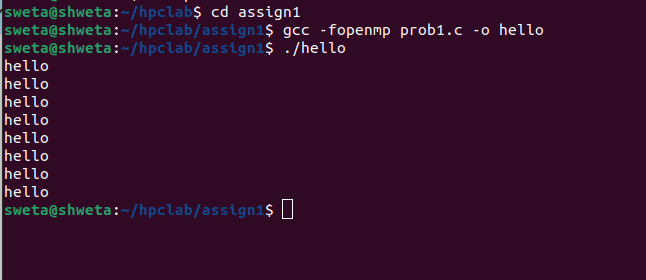
}

gcc -fopenmp test.c -o hello

.\hello.exe



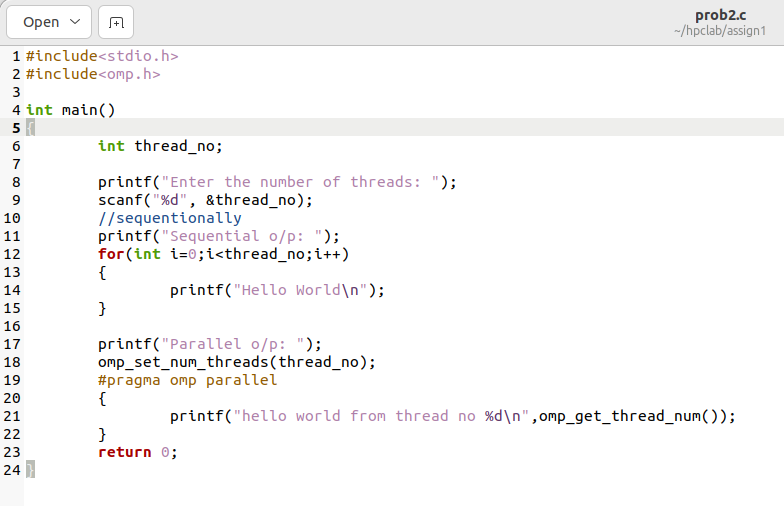




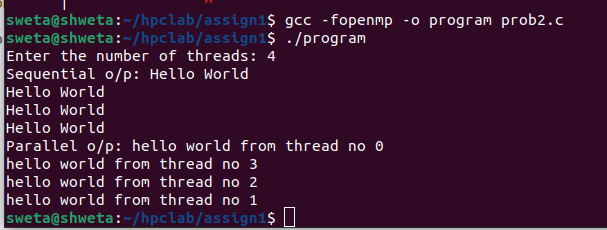
Problem Statement 2 – Print ‘Hello, World’ in Sequential and Parallel in OpenMP

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:

* In the sequential part, "Hello, World" is printed a specific number of times, corresponding to the number of threads requested. Each print statement is executed one after the other.
* In the parallel part, each thread prints "Hello, World from thread X," where X is the thread number. The order of these print statements may vary because threads run concurrently and are scheduled by the operating system, making the output potentially non-deterministic.

GitHub Link: make a public repository upload code of an assignment and paste its link here.

https://github.com/shweta29k/HPC\_LAB

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

Elaborate the parameters and show calculation.

=>

1.Number of CPU cores

2.Number of floating-point operations per cycle (usually determined by the architecture)

3.Base clock speed (in GHz)

4.Number of threads per core (usually 2 for modern CPUs with Hyper-Threading or SMT)

FLOPS = Number of cores × Number of threads per core × Base clock speed (in Hz) × Number of floating-point operations per cycle

My device details:

Processor: Intel Core i5 – 11320H 3.2GHz (supports AVX 2 instruction set)

1. No. of Cores = 4
2. Total No. Of threads = 8
3. Clock Speed = 3.2GHz = 3.2 x 10 ^ 9 Hz
4. FLOPS (AVX 2 supports) = 16 single precision(32 bit) and 8 double precision (64 bit) floating point operations per cycle per core.

**Single Precision**

FLOPS = Number of Cores×Clock Speed(in Hz)×FLOP per Cycle

= 4 × 3.2 × 10^9 ×16

= 204.8 × 10 ^9

= 204.8 GFLOPS

**Double Precision**

FLOPS = Number of Cores×Clock Speed(in Hz) ×FLOP per cycle

= 4 × 3.2 × 10^9 × 8

= 102.4 × 10^9

= 102.4 GFLOPS