Course: High Performance Computing Lab

Practical No 1

PRN: 22510046

Name: Tanmay Milind Chavan

Batch: B3

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");

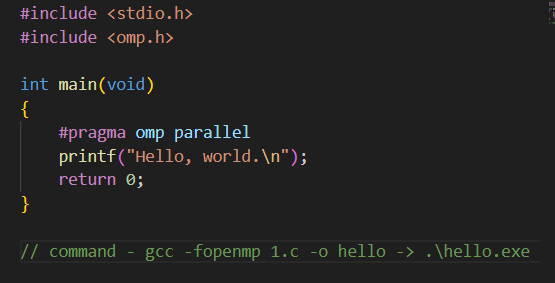
*return* 0;

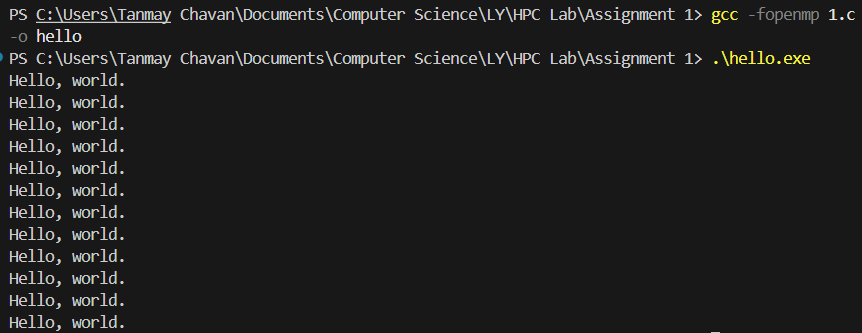
}

gcc -fopenmp test.c -o hello

.\hello.exe



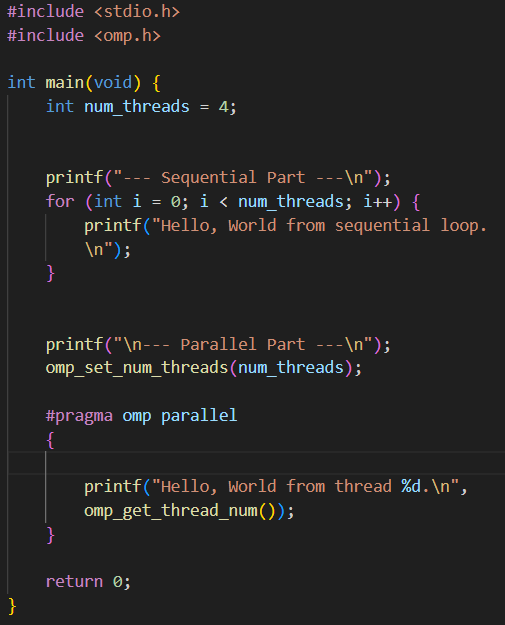




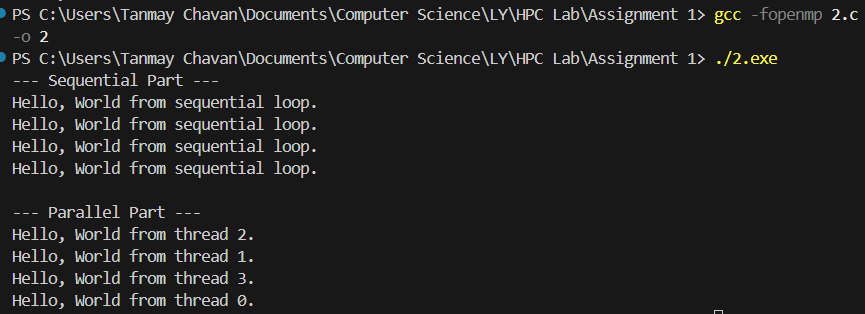
Problem Statement 2 – Print ‘Hello, World’ in Sequential and Parallel in OpenMP

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:

**Analysis:** The sequential part will print the message num\_threads times in a predictable order. The parallel part will also print the message num\_threads times, but the order of the output will be non-deterministic (it can change each time you run it). This is because the threads are racing to execute the printf statement.

GitHub Link: https://github.com/Kodar11/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.

**Elaborate the parameters and show calculation.**

1. **Parameters:**
   * **CPU Clock Speed (F):** State the value you found (3.19 GHz) and explain that this is the frequency at which the CPU's internal clock operates.
   * **Number of Cores (C):** State the value you found (8) and explain that this is the number of independent processing units on the chip.
   * **Floating-Point Operations per Cycle (I):** State the assumed value (e.g., 8 for single-precision) and briefly explain that this is due to modern instruction sets like AVX2, which allow the CPU to perform multiple floating-point operations in a single clock cycle.
2. **Calculation:**
   * Write down the formula: Practical FLOPS=F×C×I.
   * Substitute the values you found: Practical FLOPS=(3.19×109)×(8)×(16).
   * Show the result: Practical FLOPS=408.32 ×109 or 408.32 GFLOPS.

**Based on your base speed (2.00 GHz):**

* **Formula:** Theoretical FLOPS=F×C×I
* **Parameters:**
  + F (Frequency): 2.00×109 Hz
  + C (Cores): 8
  + I (Double-Precision Ops/Cycle): 16
* **Result:** Theoretical FLOPS=(2.00×109)×8×16=256 GFLOPS

**2. Based on your max turbo speed (4.40 GHz):**

* **Formula:** Theoretical FLOPS=F×C×I
* **Parameters:**
  + F (Frequency): 4.40×109 Hz
  + C (Cores): 8
  + I (Double-Precision Ops/Cycle): 16
* **Result:** Theoretical FLOPS=(4.40×109)×8×16=563.2 GFLOPS

1. **Conclusion:** Briefly state that this is the theoretical peak performance and that actual performance will be lower due to various system limitations.