Course: High Performance Computing Lab

Practical No 1

PRN: 21510068

Name:Aaditya Diak Jawanjal

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*

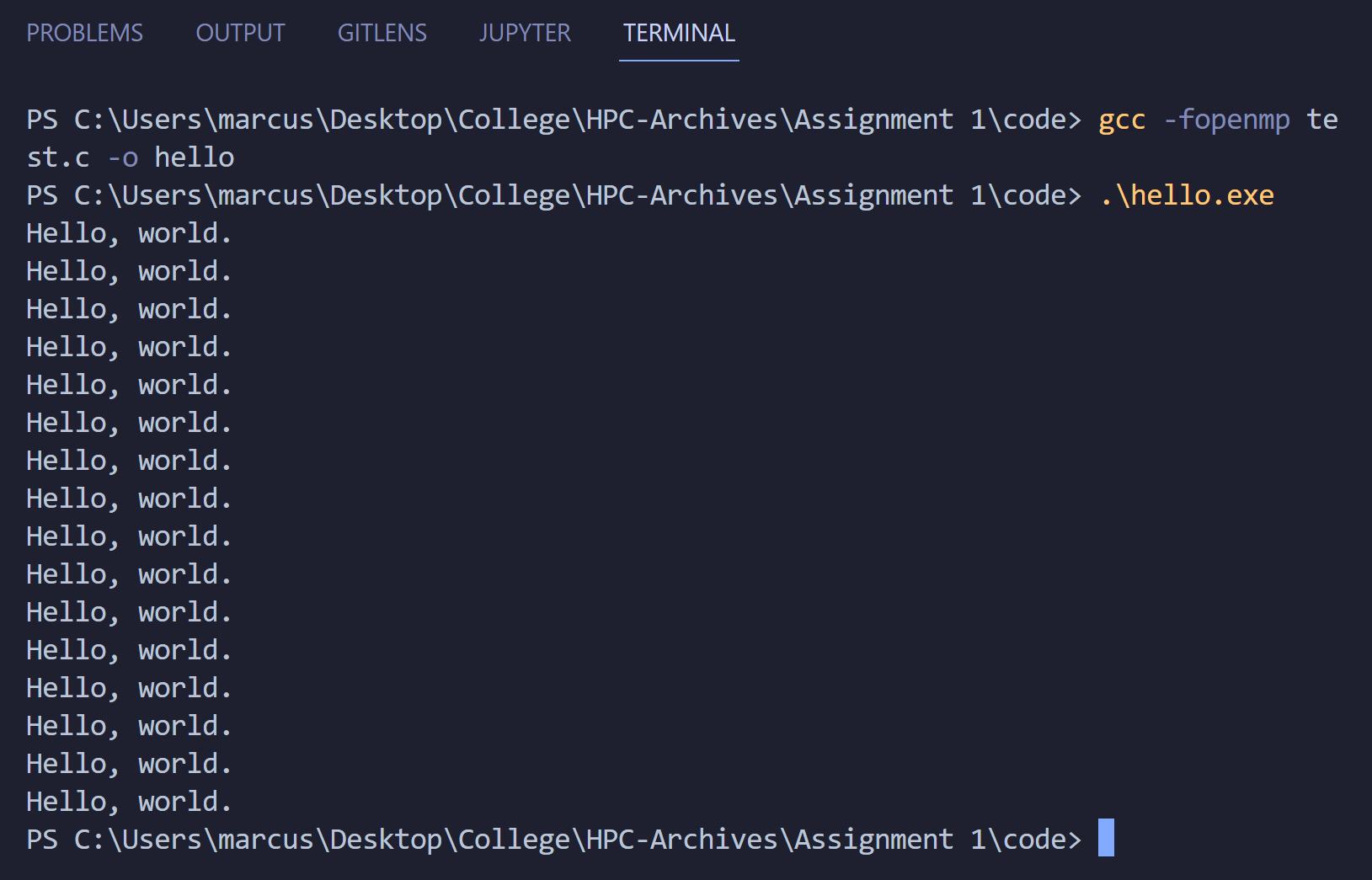
    printf("Hello, world.\n");

*return* 0;

}

gcc -fopenmp test.c -o hello

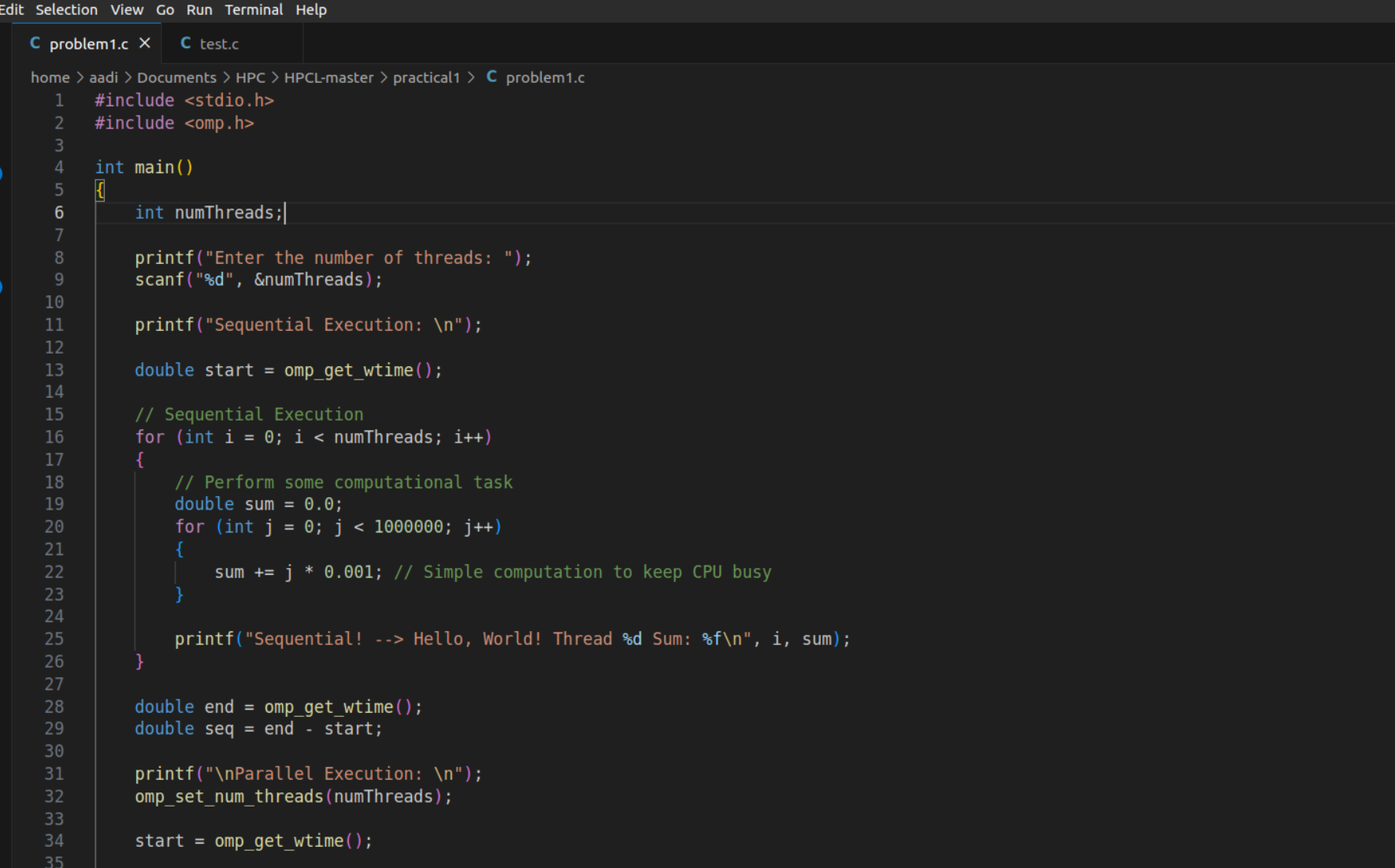
.\hello.exe

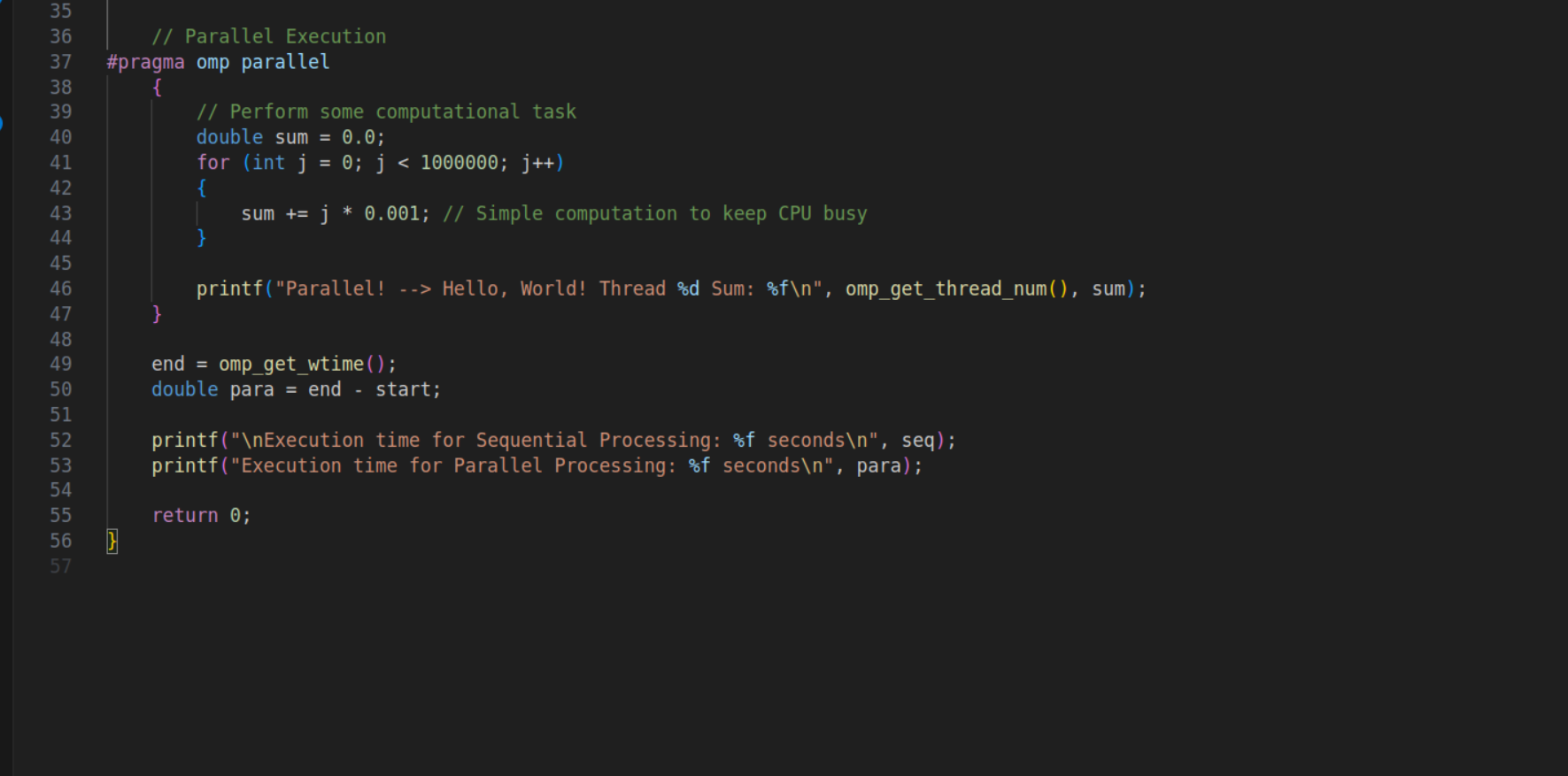


Problem Statement 2 – Print ‘Hello, World’ in Sequential and Parallel in OpenMP

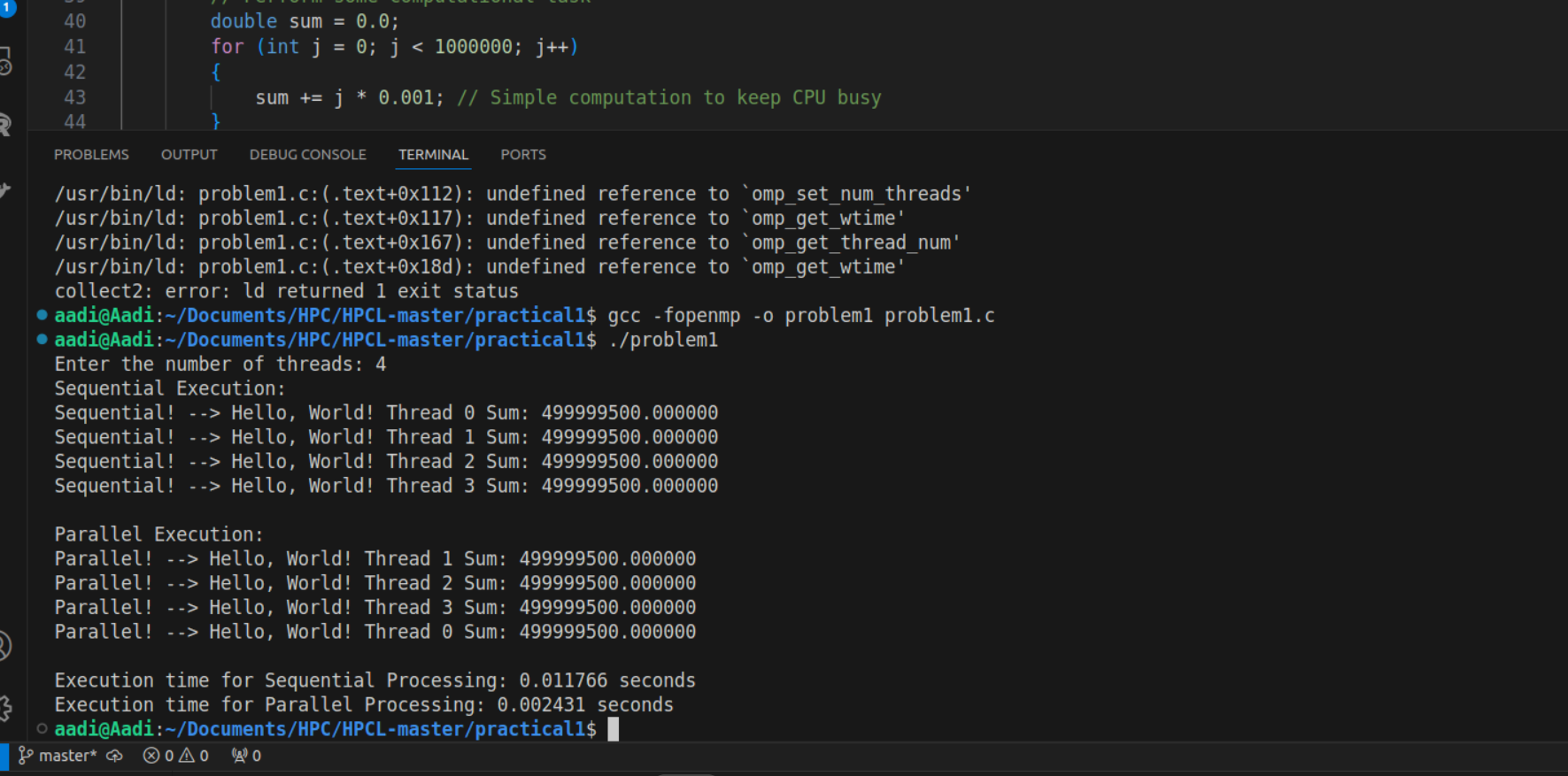
We first ask the user for number of threads – OpenMP allows to set the threads at runtime. Then, we print the Hello, World in sequential – number of times of threads count and then run the code in parallel in each thread.

Code snapshot:





Output snapshot:



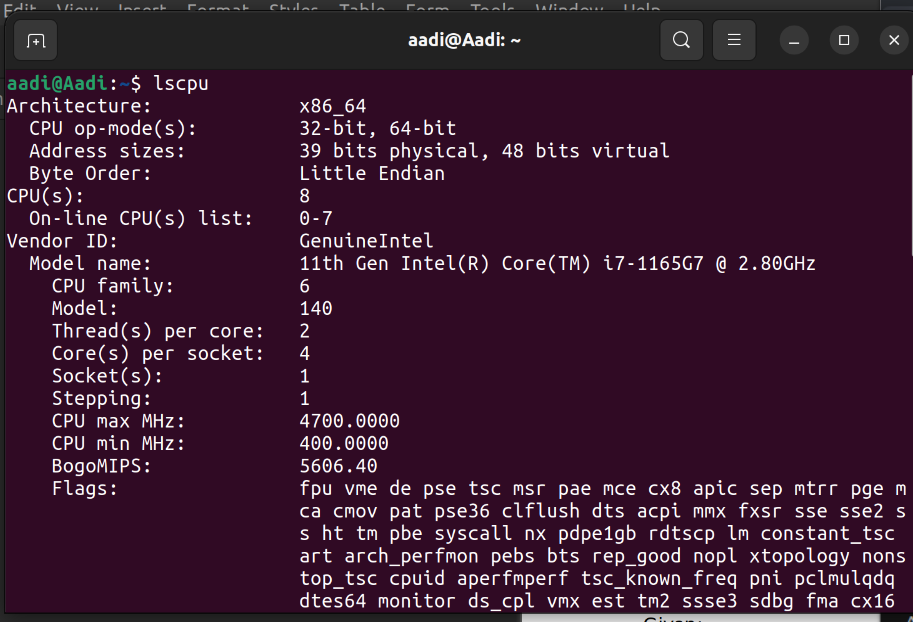
Analysis:

Sequential Execution: The code sequentially iterates over the number of threads specified Since the tasks are executed one after another, the total time taken is directly proportional to the number of iterations (numThreads).

Parallel Execution: In the parallel section, the same computational task is divided among multiple threads using OpenMP. Each thread performs the task simultaneously, reducing the total execution time compared to the sequential approach, especially if the system has multiple cores.

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

Elaborate the parameters and show calculation.



FLOPS=Number of Cores×Clock Speed (GHz)×IPC×10^9

Given:

* Number of Cores = 8
* Clock Speed = 2.8 GHz
* IPC = 16

### Calculation:

FLOPS=8×2.8×16×10^9  GFLOPS

FLOPS=2x179.2×109=358.4 GFLOPS

So, the theoretical FLOPS for your system is **358.4** GFLOPS (GigaFLOPS) with an IPC of 16.

**GitHub Link:** [**https://github.com/aadityajawanjal34/HPC\_Assignments**](https://github.com/aadityajawanjal34/HPC_Assignments)