AY 2025-26 Odd Sem

Course: Cutting Edge Technologies Lab

Course Code: 7CS352

Practical No 1

PRN: 23510119

Name: Niharika Pasnur

Batch: T1

Title: Introduction to parallel programming, OpenMP installation, simple “Hello World” programs.

Problem Statement 1: Illustrate different types of Parallel Programming Models(Shared memory, distributed memory, accelerated computing)

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;

}

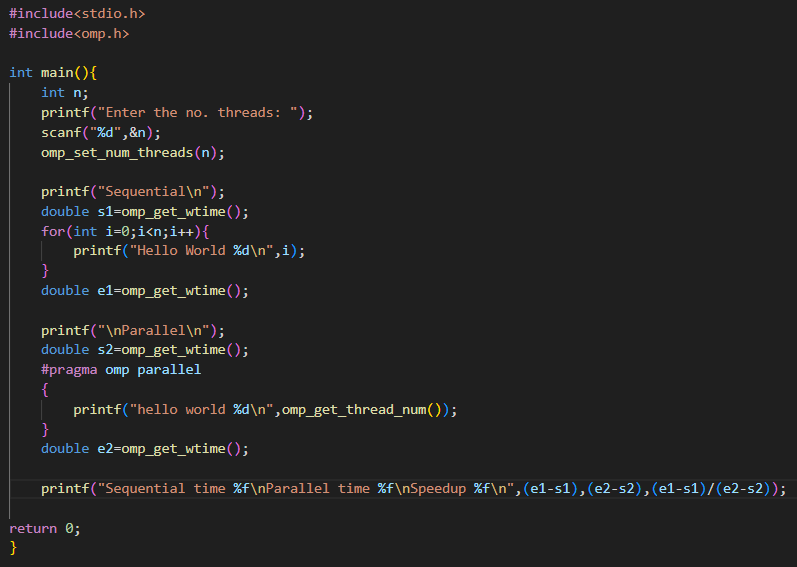
gcc -fopenmp test.c -o hello

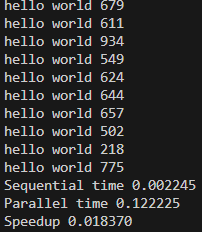
.\hello.exe



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.





Analysis:

Sequential time is less than parallel time. The speedup is 0.018.

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

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) = no. of cores\*clock frequency\* no. of FLOPs per cycle per core

Number of cores: Total cores of CPU = 8

Clock frequency: CPU frequency in Hz = 2.0 GHz

FLOPs per cycle per core: Using AVX2 = 8 FLOPs per cycle per core

FLOPS= 8 \* 2 \* 109 \* 8 = 128 x 109 FLOPS

* Theoretical peak performance: 128 GFLOPS.
* Real performance depends on memory bandwidth, instruction mix, and parallel efficiency.
* Helps estimate upper limit for floating-point computations in OpenMP programs