Walchand College of Engineering, Sangli

Department of Computer Science and Engineering

Course: High Performance Computing Lab

Practical No 1

PRN: 22510019

Name: Soham Patil

Batch: B6

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)

Note: Also install `mingw32-pthreads-w32` package.

Then, to run a program in OpenMP, we have to pass a flag `-fopenmp`.

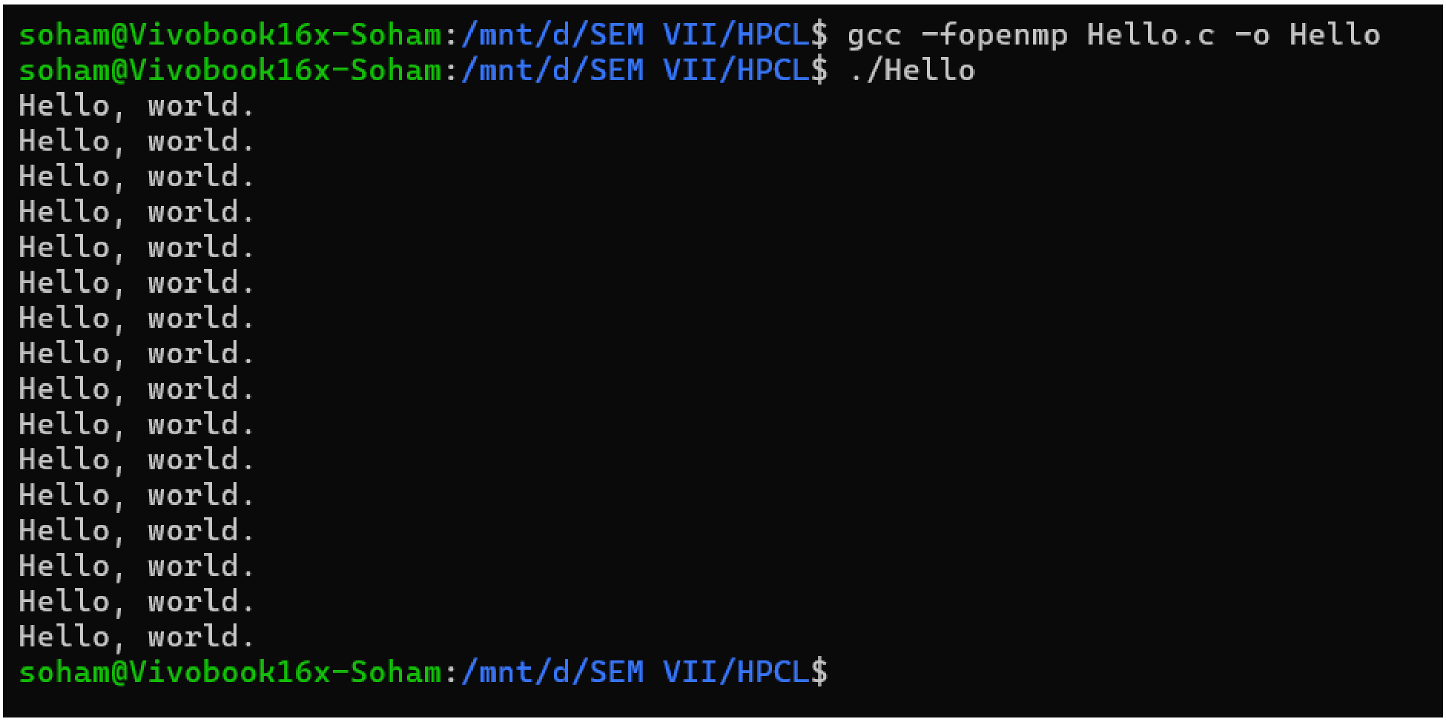
Example:

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

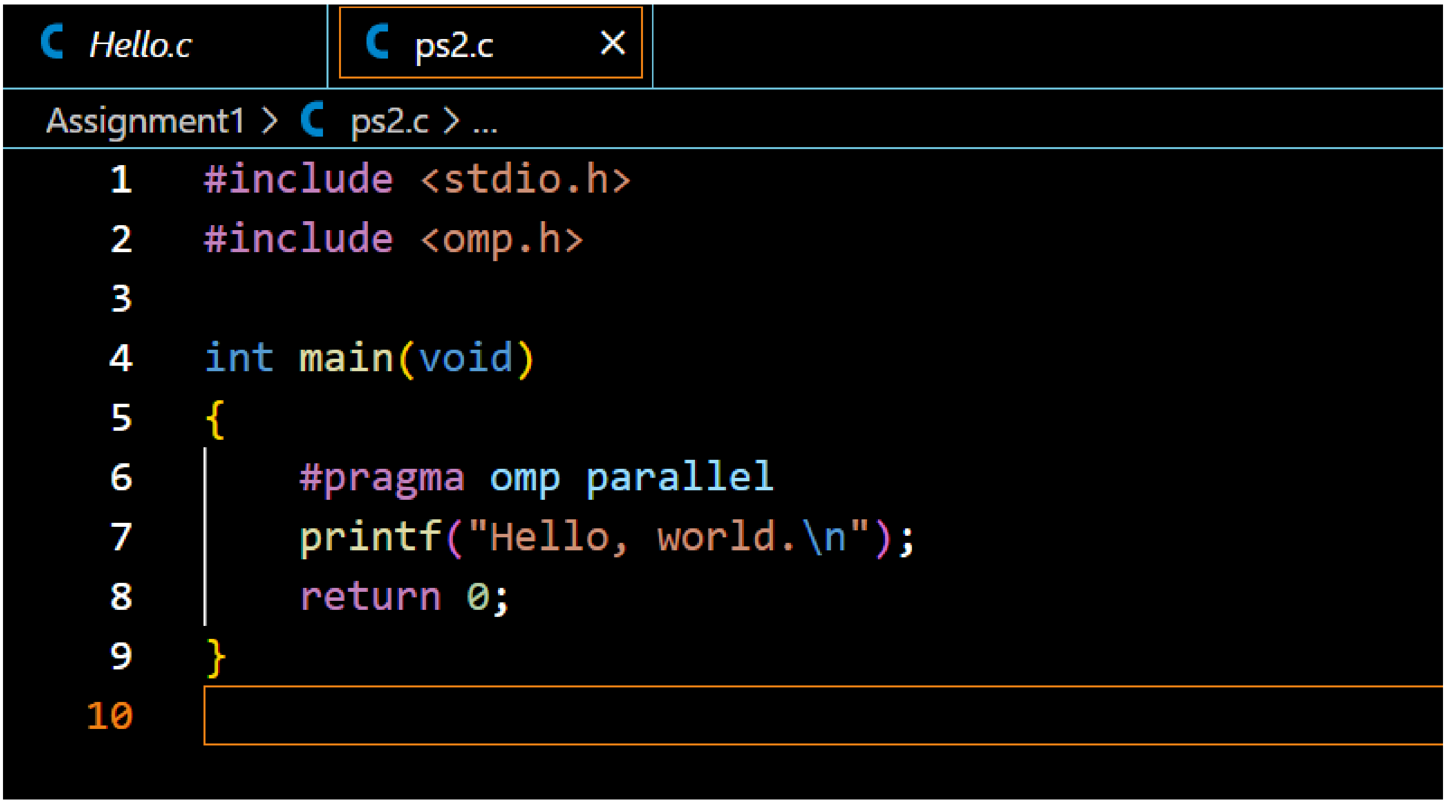


```
soham@Vivobook16x-Soham:/mnt/d/SEM VII/HPCL$ gcc -fopenmp Hello.c -o Hello
soham@Vivobook16x-Soham:/mnt/d/SEM VII/HPCL$ ./Hello
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
soham@Vivobook16x-Soham:/mnt/d/SEM VII/HPCL$
```

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

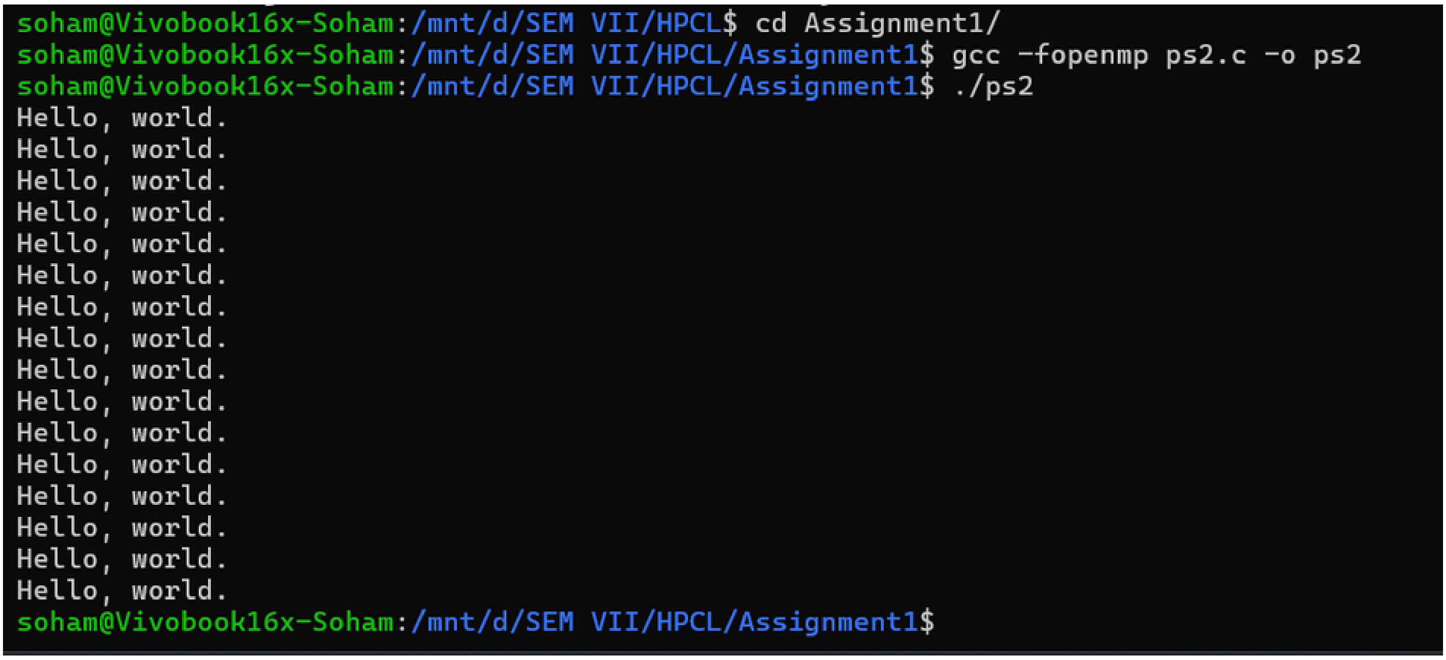
We first ask the user for number of threads – OpenMP allows to set the threads at runtime. Then, we print the Hello, World in sequential – number of times of threads count and then run the code in parallel in each thread.

Code snapshot:

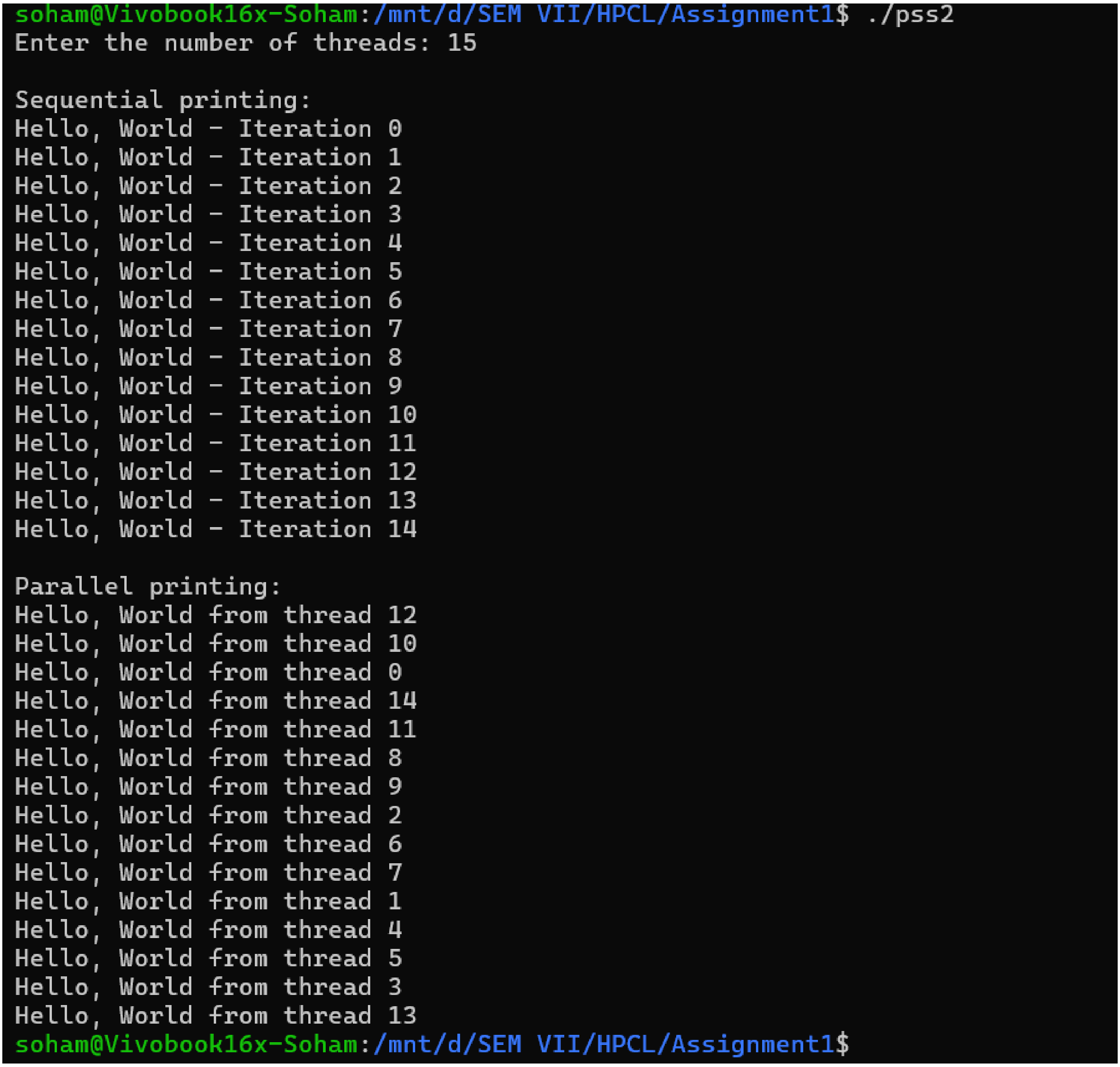


```c
#include <stdio.h>
#include <omp.h>

int main(void)
{
    #pragma omp parallel
    printf("Hello, world.\n");
    return 0;
}

```

Output snapshot:



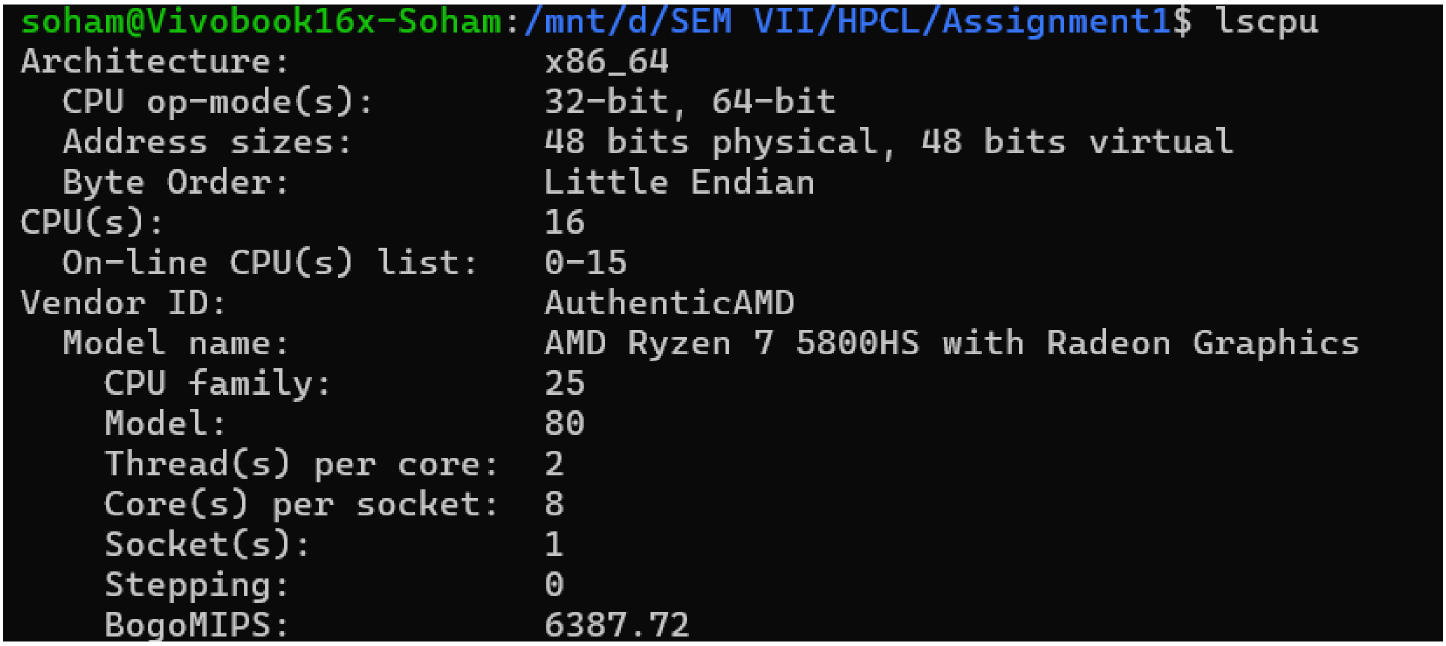
```
soham@Vivobook16x-Soham:/mnt/d/SEM VII/HPCL$ cd Assignment1/
soham@Vivobook16x-Soham:/mnt/d/SEM VII/HPCL/Assignment1$ gcc -fopenmp ps2.c -o ps2
soham@Vivobook16x-Soham:/mnt/d/SEM VII/HPCL/Assignment1$ ./ps2
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
soham@Vivobook16x-Soham:/mnt/d/SEM VII/HPCL/Assignment1$
```



```
soham@Vivobook16x-Soham:/mnt/d/SEM VII/HPCL/Assignment1$ ./pss2
Enter the number of threads: 15

Sequential printing:
Hello, World - Iteration 0
Hello, World - Iteration 1
Hello, World - Iteration 2
Hello, World - Iteration 3
Hello, World - Iteration 4
Hello, World - Iteration 5
Hello, World - Iteration 6
Hello, World - Iteration 7
Hello, World - Iteration 8
Hello, World - Iteration 9
Hello, World - Iteration 10
Hello, World - Iteration 11
Hello, World - Iteration 12
Hello, World - Iteration 13
Hello, World - Iteration 14

Parallel printing:
Hello, World from thread 12
Hello, World from thread 10
Hello, World from thread 0
Hello, World from thread 14
Hello, World from thread 11
Hello, World from thread 8
Hello, World from thread 9
Hello, World from thread 2
Hello, World from thread 6
Hello, World from thread 7
Hello, World from thread 1
Hello, World from thread 4
Hello, World from thread 5
Hello, World from thread 3
Hello, World from thread 13
soham@Vivobook16x-Soham:/mnt/d/SEM VII/HPCL/Assignment1$
```

GitHub Link: 🌐 HPCL/Assignment1 at main · Somzee5/HPCL

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



```
soham@Vivobook16x-Soham:/mnt/d/SEM VII/HPCL/Assignment1$ lscpu
Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          48 bits physical, 48 bits virtual
  Byte Order:             Little Endian
CPU(s):                   16
  On-line CPU(s) list:    0-15
Vendor ID:                AuthenticAMD
  Model name:             AMD Ryzen 7 5800HS with Radeon Graphics
    CPU family:           25
    Model:                80
    Thread(s) per core:   2
    Core(s) per socket:   8
    Socket(s):            1
    Stepping:             0
    BogoMIPS:             6387.72
```

8 ops (256-bit AVX2) × 2 (FMA) = 16 FLOPs per cycle

FLOPs per Second per Core

= Clock speed × FLOPs per cycle per core

= 4.1 GHz × 16 FLOPs/cycle

= $4.1 \times 10^9 \times 16$

= 65.6 GFLOPS/core

FLOPS = $8 \times 4.1 \times 10^9 \times 16$

= **524.8 GFLOPS**

Class: Final Year (CSE)   Year: 2025-26 Semester: 1