Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

Class: Final Year B.Tech(Computer Science and Engineering)

Year: 2025-26 **Semester:** 1

Course: High Performance Computing Lab

PRN: 22510021 **Batch:** B7

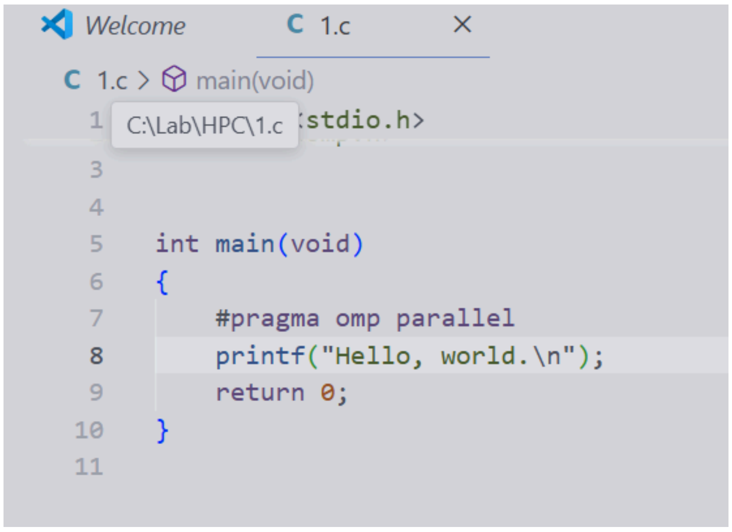
Practical No. 1

Title of practical: Introduction to OpenMP

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

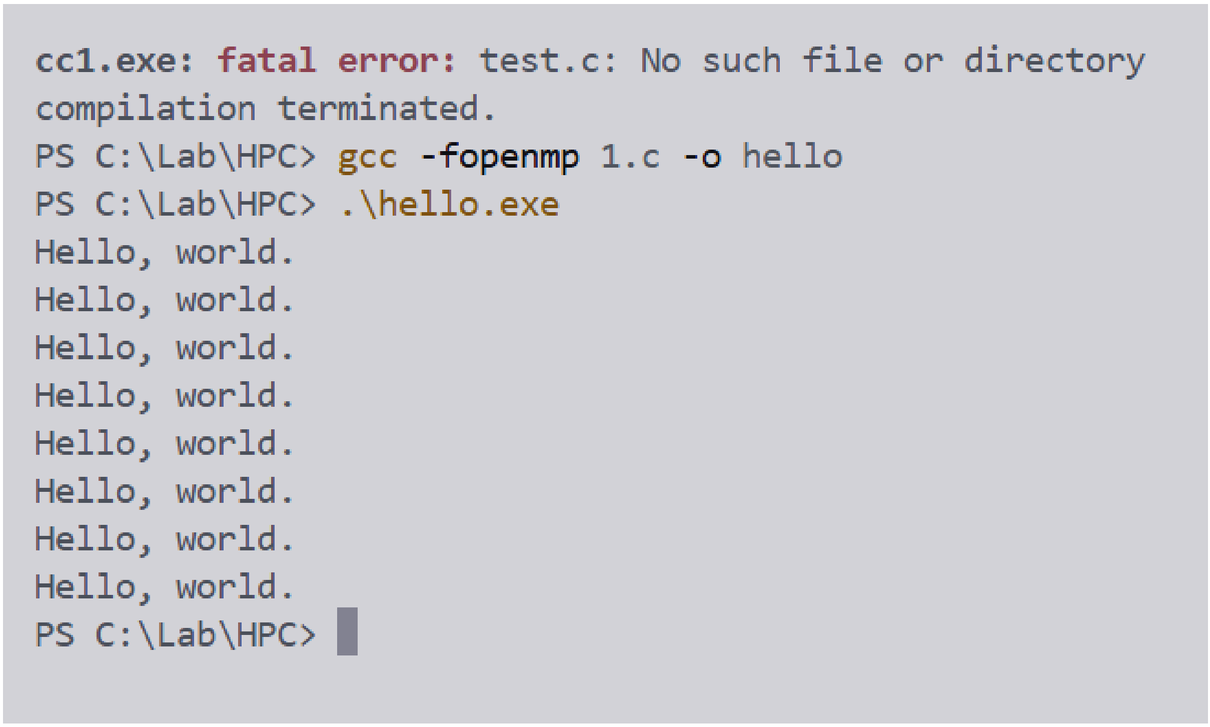
Example:

- To run a basic Hello World,

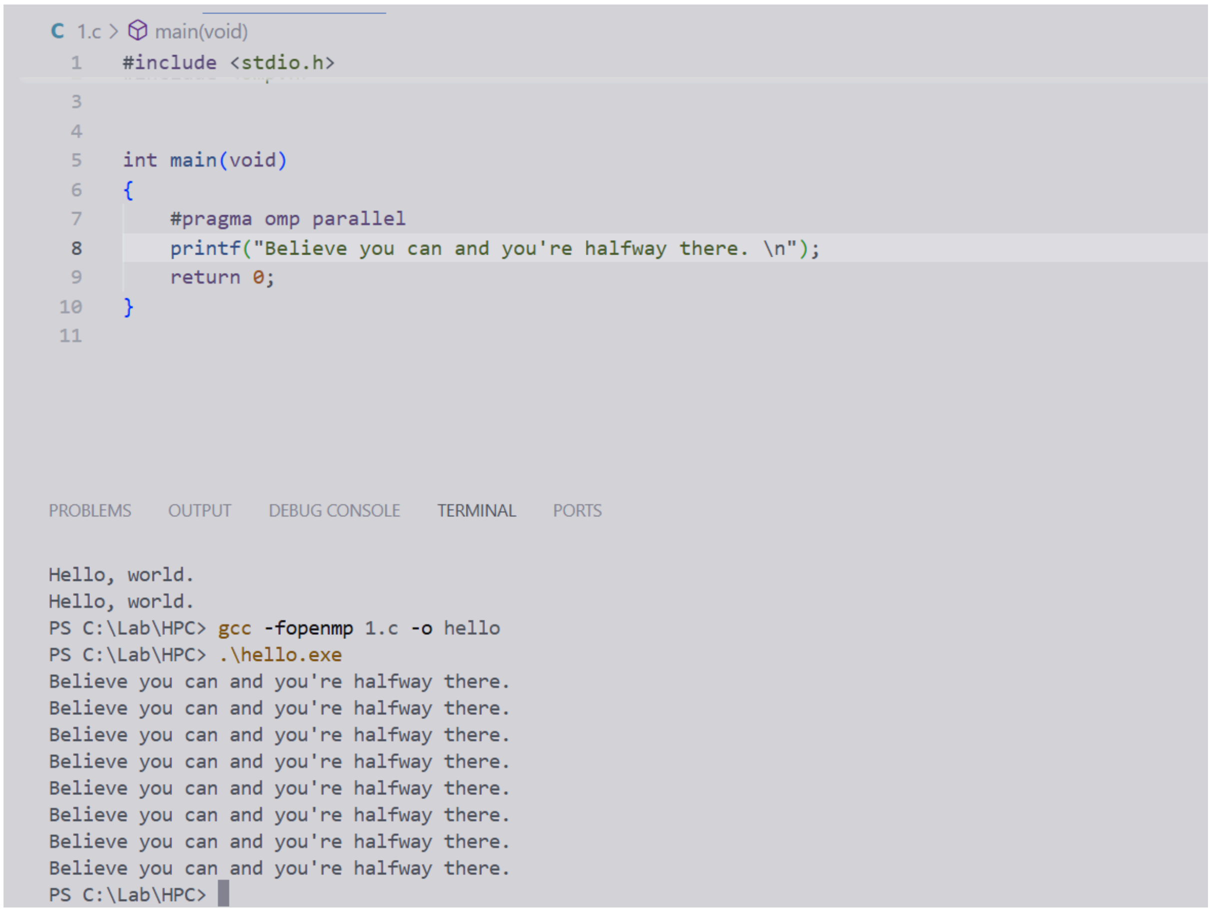


```
:stdio.h>


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

```
gcc -fopenmp test.c -o hello
.\hello.exe
```



```
cc1.exe: fatal error: test.c: No such file or directory
compilation terminated.
PS C:\Lab\HPC> gcc -fopenmp 1.c -o hello
PS C:\Lab\HPC> .\hello.exe
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
PS C:\Lab\HPC>
```

- 

```c
#include <stdio.h>



int main(void)
{
    #pragma omp parallel
    printf("Believe you can and you're halfway there. \n");
    return 0;
}

```

```
Hello, world.
Hello, world.
PS C:\Lab\HPC> gcc -fopenmp 1.c -o hello
PS C:\Lab\HPC> .\hello.exe
Believe you can and you're halfway there.
Believe you can and you're halfway there.
Believe you can and you're halfway there.
Believe you can and you're halfway there.
Believe you can and you're halfway there.
Believe you can and you're halfway there.
Believe you can and you're halfway there.
Believe you can and you're halfway there.
PS C:\Lab\HPC>
```

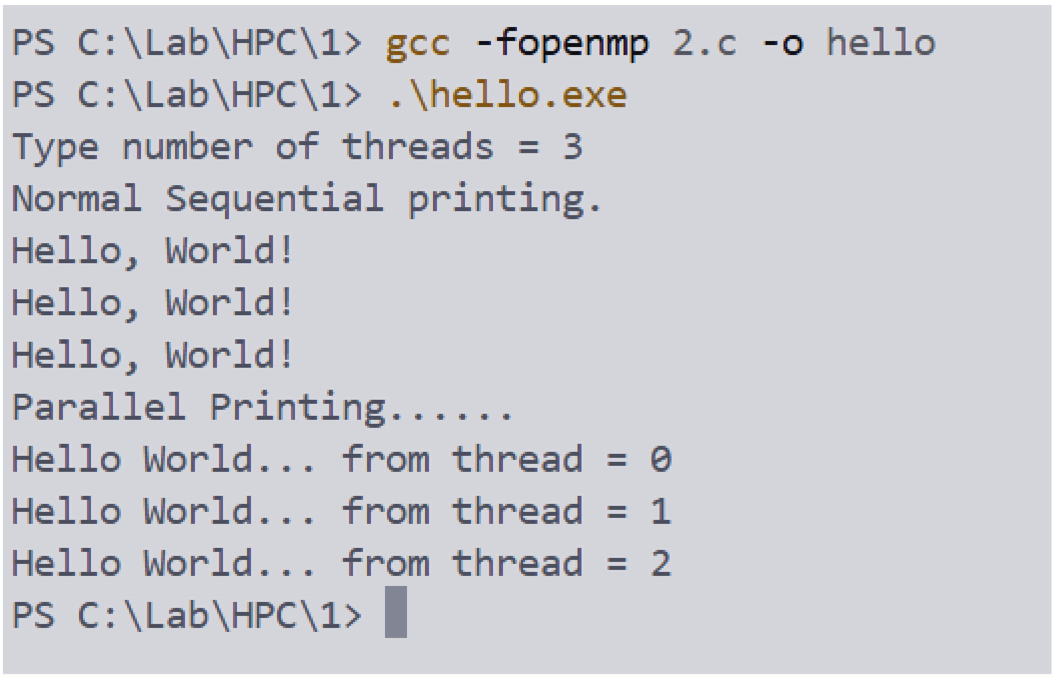
Problem Statement 2 – Print ‘Hello, World’ in Sequential and Parallel in OpenMP

We first ask the user for a number of threads – OpenMP allows us 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(){

    int n;

    printf("Type number of threads = ");

    scanf("%d", &n);

    omp_set_num_threads(n);

    printf("Normal Sequential printing. \n");

    for(int i=0; i<n; i++){

        printf("Hello, World!\n");

    }

    printf("Parallel Printing......\n");

    #pragma omp parallel

    {

        printf("Hello World... from thread = %d\n",

            omp_get_thread_num());

    }

    return 0;

}
```

Output snapshot:



```
PS C:\Lab\HPC\1> gcc -fopenmp 2.c -o hello
PS C:\Lab\HPC\1> .\hello.exe
Type number of threads = 3
Normal Sequential printing.
Hello, World!
Hello, World!
Hello, World!
Parallel Printing......
Hello World... from thread = 0
Hello World... from thread = 1
Hello World... from thread = 2
PS C:\Lab\HPC\1>
```

Analysis:

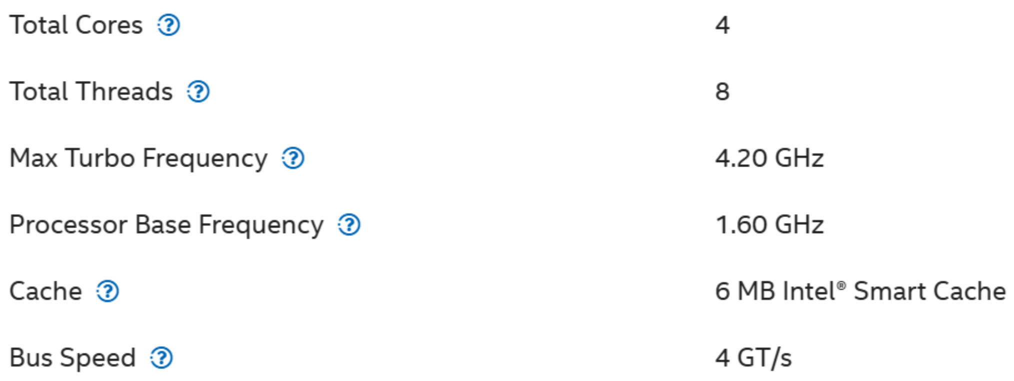
By using the omp set function we can set number of threads desired for out program

- omp_get_num_threds = sets how many threads OpenMP should create.
- #pragma omp parallel = initiates the parallel region.
- omp_get_num_threads = provides the thread ID during parallel execution.

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.

Intel(R) Core(TM) i5-10210u



Total Cores	4
Total Threads	8
Max Turbo Frequency	4.20 GHz
Processor Base Frequency	1.60 GHz
Cache	6 MB Intel® Smart Cache
Bus Speed	4 GT/s

Elaborate the parameters and show calculation.

Below is the formula to calculate FLOPS :
MAX FLOPS = (# Number of cores) * (Clock Frequency (cycles/sec)) * (# FLOPS / cycle)

Number of cores = 4

Clock Frequency = 1.60 GHz (base freq is taken here)

FLOPS/cycle = 8

MAX FLOPS = (# Number of cores) * (Clock Frequency (cycles/sec)) * (# FLOPS / cycle)

1. AT BASE

   CLOCK MAX FLOPS = 4 * 1.60 GHz * 8

   51.2 GFLOPS

2. AT BASE TURBO

   MAX FLOPS = 4 * 4.20 GHz * 8

   134.4 GFLOPS