

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

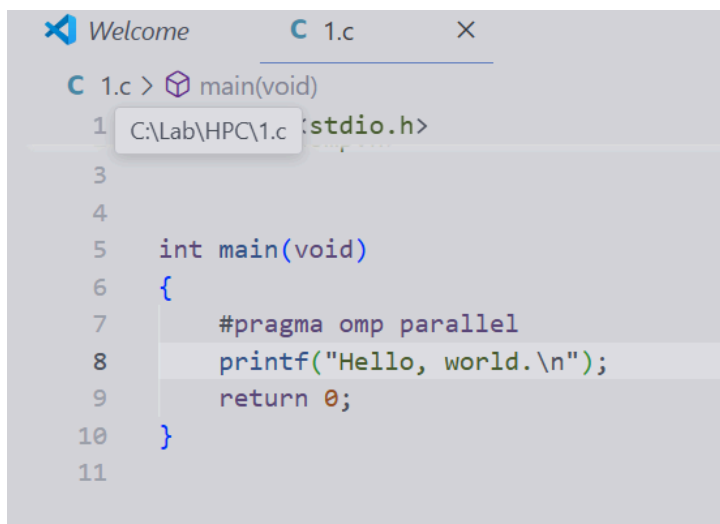
**Practical No. 1**

**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,

A screenshot of a code editor window titled 'Welcome' with a tab for '1.c'. The code is in C and includes a comment 'C 1.c > main(void)'. The code starts with a preprocessor directive for stdio.h, followed by an empty line, then the main function signature 'int main(void)', an opening curly brace, a pragma directive for OpenMP parallel, a printf statement to print 'Hello, world.\n', a return statement with 0, and a closing curly brace. Line numbers 1 through 11 are visible on the left margin.

```
C 1.c > main(void)
1  C:\Lab\HPC\1.c :stdio.h>
3
4
5  int main(void)
6  {
7      #pragma omp parallel
8      printf("Hello, world.\n");
9      return 0;
10 }
11
```

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

```
.\hello.exe
```

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

```
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 1.c > main(void)
1  #include <stdio.h>
2
3
4
5  int main(void)
6  {
7      #pragma omp parallel
8      printf("Believe you can and you're halfway there. \n");
9      return 0;
10 }
11
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
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>
```

# Walchand College of Engineering, Sangli

## Department of Computer Science and Engineering

### 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:

```
#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;
}
```

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

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 our program

- `omp_get_num_threads` = 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

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

|                            |                         |
|----------------------------|-------------------------|
| 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 :**

$$\text{MAX FLOPS} = (\# \text{ Number of cores}) * (\text{Clock Frequency (cycles/sec)}) * (\# \text{ FLOPS / cycle})$$

Number of cores = 4

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

FLOPS/cycle = 8

$$\text{MAX FLOPS} = (\# \text{ Number of cores}) * (\text{Clock Frequency (cycles/sec)}) * (\# \text{ FLOPS / cycle})$$

1. AT BASE

$$\text{CLOCK MAX FLOPS} = 4 * 1.60 \text{ GHz} * 8$$

**51.2 GFLOPS**

2. AT BASE TURBO

$$\text{MAX FLOPS} = 4 * 4.20 \text{ GHz} * 8$$

**134.4 GFLOPS**