

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

Course: High Performance Computing Lab

Practical No 1

PRN: 22520007

Name: Manish Namdev Barage

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.

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

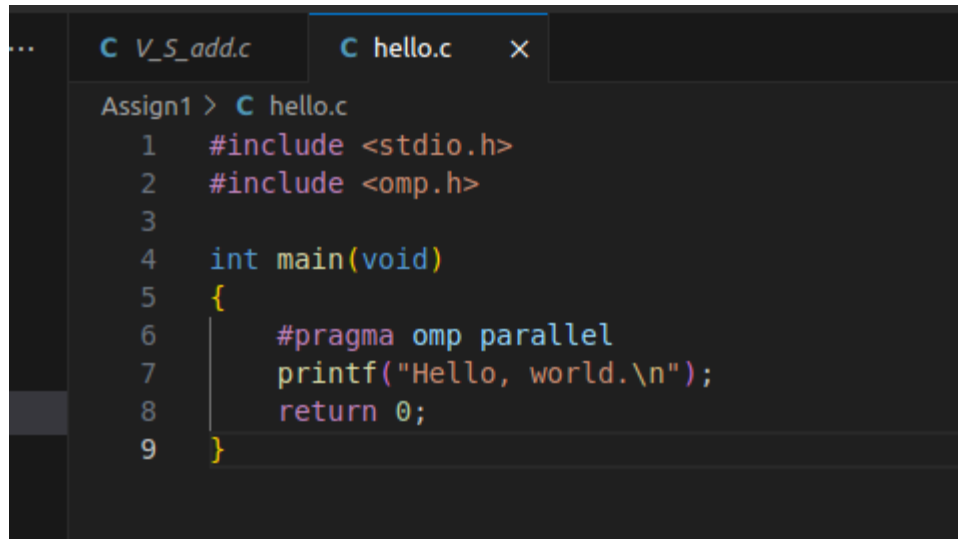
Then, to run a program in OpenMP, we have to pass a flag `-fopenmp`.

Example:

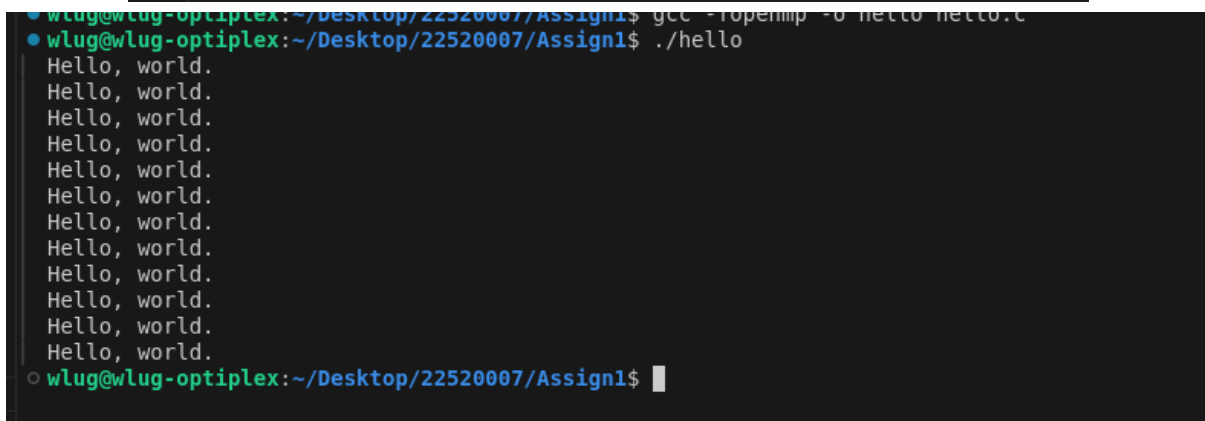
To run a basic Hello World,

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

Program: Hello world basic program in C.



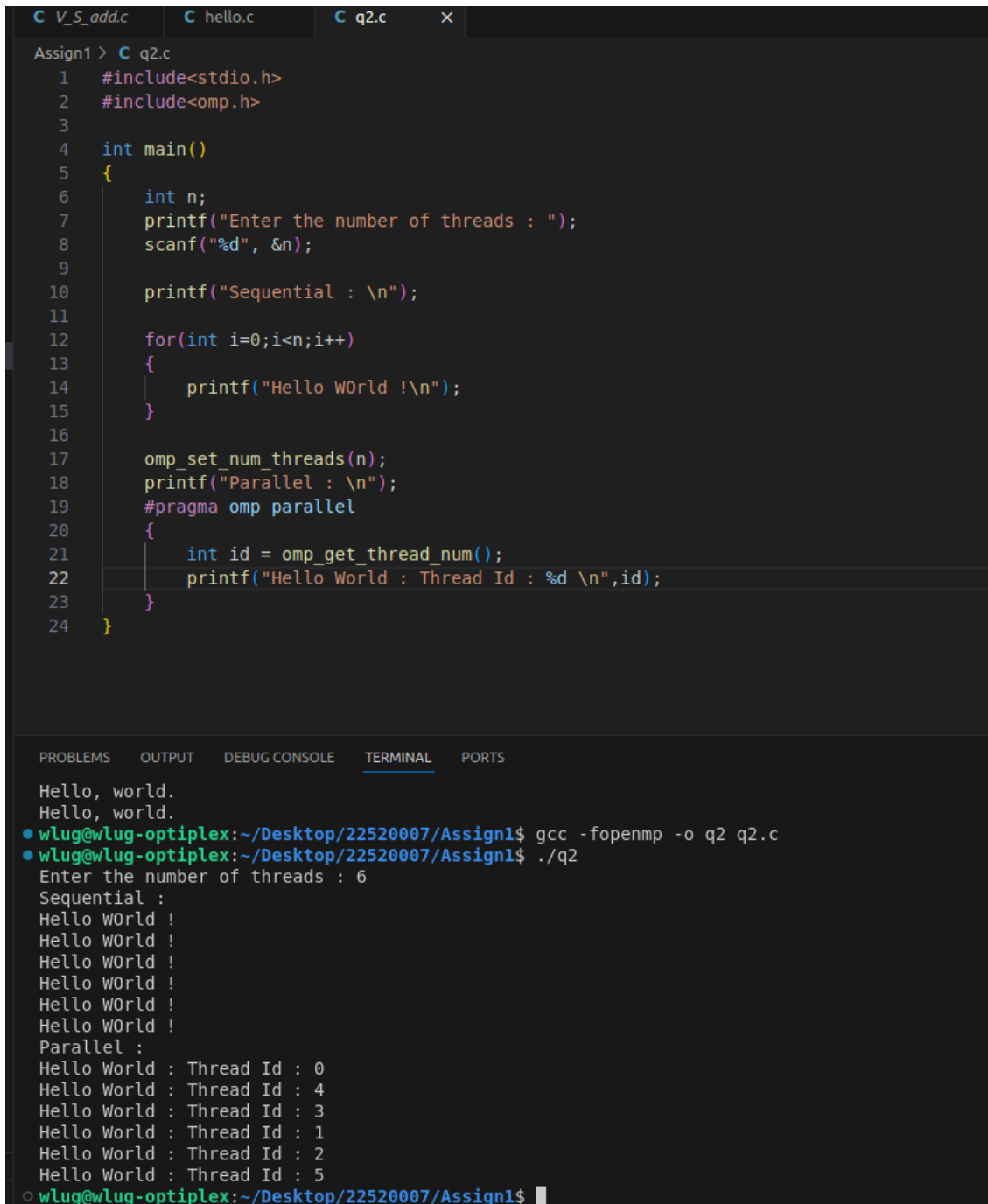
```
...  C V_S_add.c  C hello.c  x
Assign1 > C hello.c
1  #include <stdio.h>
2  #include <omp.h>
3
4  int main(void)
5  {
6      #pragma omp parallel
7      printf("Hello, world.\n");
8      return 0;
9  }
```



```
wlug@wlug-optiplex:~/Desktop/22520007/Assign1$ gcc -fopenmp -o hello hello.c
wlug@wlug-optiplex:~/Desktop/22520007/Assign1$ ./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.
wlug@wlug-optiplex:~/Desktop/22520007/Assign1$
```

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.



```
C V_S_add.c C hello.c C q2.c X
Assign1 > C q2.c
1  #include<stdio.h>
2  #include<omp.h>
3
4  int main()
5  {
6      int n;
7      printf("Enter the number of threads : ");
8      scanf("%d", &n);
9
10     printf("Sequential : \n");
11
12     for(int i=0;i<n;i++)
13     {
14         printf("Hello WOrld !\n");
15     }
16
17     omp_set_num_threads(n);
18     printf("Parallel : \n");
19     #pragma omp parallel
20     {
21         int id = omp_get_thread_num();
22         printf("Hello World : Thread Id : %d \n",id);
23     }
24 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Hello, world.
Hello, world.
• wlug@wlug-optiplex:~/Desktop/22520007/Assign1$ gcc -fopenmp -o q2 q2.c
• wlug@wlug-optiplex:~/Desktop/22520007/Assign1$ ./q2
Enter the number of threads : 6
Sequential :
Hello World !
Hello World !
Hello World !
Hello World !
Hello World !
Hello World !
Parallel :
Hello World : Thread Id : 0
Hello World : Thread Id : 4
Hello World : Thread Id : 3
Hello World : Thread Id : 1
Hello World : Thread Id : 2
Hello World : Thread Id : 5
• wlug@wlug-optiplex:~/Desktop/22520007/Assign1$
```

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

Analysis: We can provide how many threads we want to execute a task. Every thread runs the task parallelly. The sequence of threads is random and any thread is selected for the execution randomly.

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

Code snapshot:

```
Assign1 > C flops.c
1  #include <stdio.h>
2
3  int main() {
4
5      double clock_speed_ghz = 3.2;
6      int num_cores = 6;
7      int ipc = 4;
8      int flop_per_instruction = 2;
9
10
11     double clock_speed_hz = clock_speed_ghz * 1e9;
12
13
14     double flops = clock_speed_hz * num_cores * ipc * flop_per_instruction;
15
16
17     printf("Theoretical FLOPS: %.2e\n", flops);
18
19     return 0;
20 }
21
```

Output:

```
Theoretical FLOPS: 1.54e+11
wlug@wlug-optiplex:~/Desktop/22520007/Assign1$
```

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

Parameters:

1. FLOPS: Floating point operations per seconds, which means how many floating point arithmetic operations such as addition, subtraction, multiplication or division can be executed by CPU in one second.

$$\text{FLOPS} = (\text{Time needed for calculation}) / \text{clocks per second}$$

2. CPU clock speed: 2.4 Ghz

Can be found using command: *lscpu | grep "Ghz"*

This shows the number of clock cycles performed by the CPU