

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

Course: High Performance Computing Lab

Practical No 1

Github Link : [Sem-7-Assign/HPC lab at main · parshwa913/Sem-7-Assign · GitHub](#)

PRN: 22510064

Name: Parshwa Herwade

Batch: B1

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.

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

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

Example:

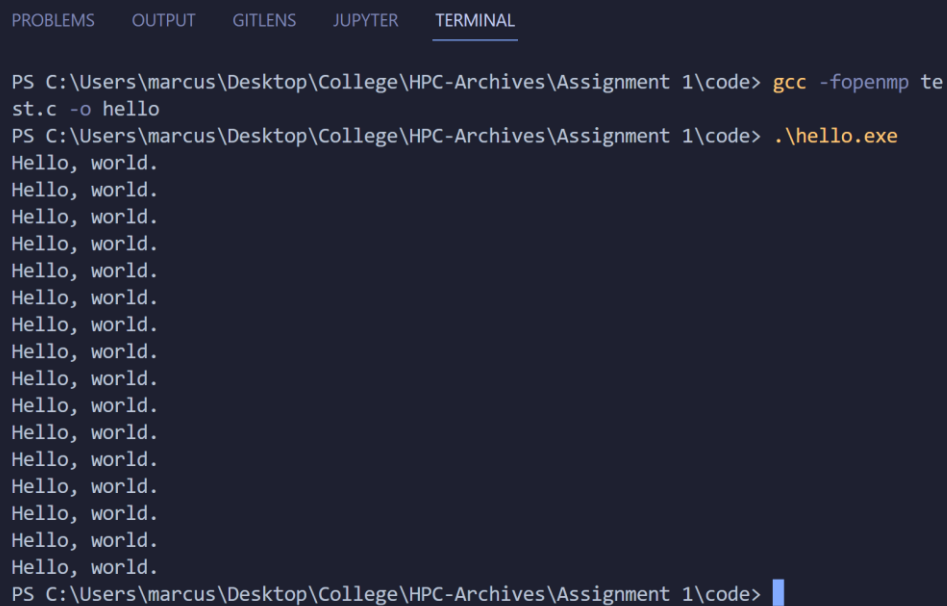
To run a basic Hello World,

```
#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
```



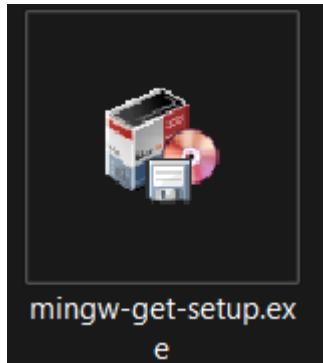
The screenshot shows a terminal window with the following content:

```
PROBLEMS  OUTPUT  GITLENS  JUPYTER  TERMINAL

PS C:\Users\marcus\Desktop\College\HPC-Archives\Assignment 1\code> gcc -fopenmp te
st.c -o hello
PS C:\Users\marcus\Desktop\College\HPC-Archives\Assignment 1\code> .\hello.exe
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.
Hello, world.
Hello, world.
Hello, world.
PS C:\Users\marcus\Desktop\College\HPC-Archives\Assignment 1\code> |
```

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

ANS.



The mingw setup has previously been done successfully by me for codes in c,cpp to run.

```
C:\Users\Parshwa>gcc --version
gcc (MinGW.org GCC-6.3.0-1) 6.3.0
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
C:\Users\Parshwa>mingw-get install mingw32-pthreads-w32
http://prdownloads.sourceforge.net/mingw/pthreads-GC-w32-2.10-mingw32-pre-20160821-1-dev.tar.xz?download
44.17 kB / 44.17 kB |=====| 100%
http://prdownloads.sourceforge.net/mingw/pthreads-w32-2.10-mingw32-pre-20160821-1-doc.tar.xz?download
38.54 kB / 38.54 kB |=====| 100%
http://prdownloads.sourceforge.net/mingw/pthreads-w32-2.10-mingw32-pre-20160821-1-lic.tar.xz?download
10.32 kB / 10.32 kB |=====| 100%
install: libpthreadgc-2.10-mingw32-pre-20160821-1-dev.tar.xz
installing libpthreadgc-2.10-mingw32-pre-20160821-1-dev.tar.xz
install: pthreads-w32-2.10-mingw32-pre-20160821-1-dev.tar.xz
installing pthreads-w32-2.10-mingw32-pre-20160821-1-dev.tar.xz
install: pthreads-w32-2.10-mingw32-pre-20160821-1-doc.tar.xz
installing pthreads-w32-2.10-mingw32-pre-20160821-1-doc.tar.xz
install: pthreads-w32-2.10-mingw32-pre-20160821-1-lic.tar.xz
installing pthreads-w32-2.10-mingw32-pre-20160821-1-lic.tar.xz
```

The pthreads package is installed too...

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

```
C test.c > main(void)
1 //the below code prints the message "Hello, world."
2 //for the number of threads specified by the number of actual threads available in the system
3 //my system has 8 cores,12 threads
4 //default answer will likely be 12 times message printed
5 #include <stdio.h>
6 #include <omp.h>
7
8 int main(void)
9 {
10     #pragma omp parallel
11     printf("Hello, world.\n");
12     return 0;
13 }
14
15 // #include <stdio.h>
16 // #include <omp.h>
17
18 // int main(void)
19 // {
20 //     omp_set_num_threads(4); // the code here sets the number of threads to 4
21
22 //     #pragma omp parallel
23 //     printf("Hello, world. %d\n", omp_get_thread_num());
24
25 //     return 0;
26 // }
```

OUTPUT:

```
PS C:\Users\Parshwa\Desktop\ASSIGN\HPC lab\22510064_HPC_A1> gcc -fopenmp test.c -o test.exe
PS C:\Users\Parshwa\Desktop\ASSIGN\HPC lab\22510064_HPC_A1> ./test.exe
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
PS C:\Users\Parshwa\Desktop\ASSIGN\HPC lab\22510064_HPC_A1>
```

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

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

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 hello_threads.c > main()
1  #include <stdio.h>
2  #include <omp.h>
3
4  int main() {
5      int n;
6      printf("Enter number of threads: ");
7      scanf("%d", &n);
8
9      printf("\nSequential Hello:\n");
10     for (int i = 0; i < n; i++) {
11         printf("Hello, World. %d\n", i);
12     }
13
14     printf("\nParallel Hello:\n");
15     omp_set_num_threads(9); // Set number of threads to 9
16     #pragma omp parallel
17     {
18         printf("Hello, World. %d\n", omp_get_thread_num());
19     }
20
21     return 0;
22 }
```

Output snapshot:

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

```
PS C:\Users\Parshwa\Desktop\ASSIGN\HPC lab\22510064_HPC_A1> gcc -fopenmp hello_threads.c -o hello_threads.exe
PS C:\Users\Parshwa\Desktop\ASSIGN\HPC lab\22510064_HPC_A1> ./hello_threads.exe
Enter number of threads: 9

Sequential Hello:
Hello, World. 0
Hello, World. 1
Hello, World. 2
Hello, World. 3
Hello, World. 4
Hello, World. 5
Hello, World. 6
Hello, World. 7
Hello, World. 8

Parallel Hello:
Hello, World. 5
Hello, World. 2
Hello, World. 0
Hello, World. 4
Hello, World. 6
Hello, World. 1
Hello, World. 3
Hello, World. 7
Hello, World. 8
PS C:\Users\Parshwa\Desktop\ASSIGN\HPC lab\22510064_HPC_A1>
```

Analysis:

In the sequential part of the program, the messages are printed one after another in order because everything runs on a single thread. It's predictable and always prints the same way.

In the parallel part, multiple threads print messages at the same time. That's why the output comes in a random order — it depends on how the system schedules the threads.

This shows how OpenMP allows different threads to work at the same time. It also helps us understand that when using parallel programming, output might not always be in order unless we manage the threads properly.

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

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

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

Elaborate the parameters and show calculation.

ANSWER:

To calculate the theoretical FLOPS of my system:

- Number of cores: 8
- Number of threads : 12
- Clock speed: 2.00 GHz = 2.00×10^9 Hz
- FLOPs per cycle (assumed): 8 (my CPU-i5 12450H)

Formula:

$$\begin{aligned}\text{FLOPS} &= \text{Number of Cores} \times \text{Clock Speed} \times \text{FLOPs per cycle} \\ &= 8 \times 2.00 \times 10^9 \times 8 \\ &= 128 \times 10^9 \text{ FLOPS} \\ &= \mathbf{128 \text{ GFLOPS}}\end{aligned}$$

So, the theoretical performance of my system is approximately **128 GFLOPS**.