Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Name : Saurabh Nitnaware	
Experiment No : 4	
Write a "Hello World" program using OpenMP library also display number of threads created during execution.	
Date of Performance : 15/02/24	
Date of Submission : 19/04/24	

Aim: Write a "Hello World" program using OpenMP library also display number of threads created during execution.

Objective: The objective of this task is to create a "Hello World" program using the OpenMP library in C, which will also display the number of threads created during execution.

Theory: The OpenMP API is a widely-used programming interface for parallel programming in shared-memory architectures, such as multicore CPUs. It provides a set of compiler directives, library routines, and environment variables that allow developers to create parallel programs in C, C++, and Fortran.

One of the simplest examples of a parallel program using OpenMP is a "Hello World" program that prints a message from multiple threads in parallel. In this program, we can use the #pragma omp parallel directive to create a parallel region, which is a block of code that will be executed by multiple threads in parallel. The omp_get_thread_num() function can then be used to determine the ID of the current thread within the parallel region, and the omp_get_num_threads() function can be used to determine the total number of threads that are created within the parallel region.

Here is a sample "Hello World" program using OpenMP in C that displays the number of threads created during execution:

Code:

```
#include <stdio.h>

#include <omp.h>

int main() {

    int num_threads = omp_get_max_threads();

    printf("Number of threads: %d\n", num_threads);

    #pragma omp parallel

    {

        int thread_id = omp_get_thread_num();
```

```
    printf("Hello, world from thread %d\n", thread_id);

  }

  return 0;

}
```

Conclusion: In conclusion, the "Hello World" program using OpenMP library in C provides a basic example of parallel programming using shared memory architecture. By using the #pragma omp parallel directive, we create a parallel region that executes a block of code in parallel. The omp_get_thread_num() and omp_get_num_threads() functions are used to get the ID of the current thread and the total number of threads created within the parallel region, respectively.