Name: Khushi Nitinkumar Patel
PRN: 2020BTECS00037
Batch: B3

HPCL Practical - I

1. Introduction to Parallel Programming.

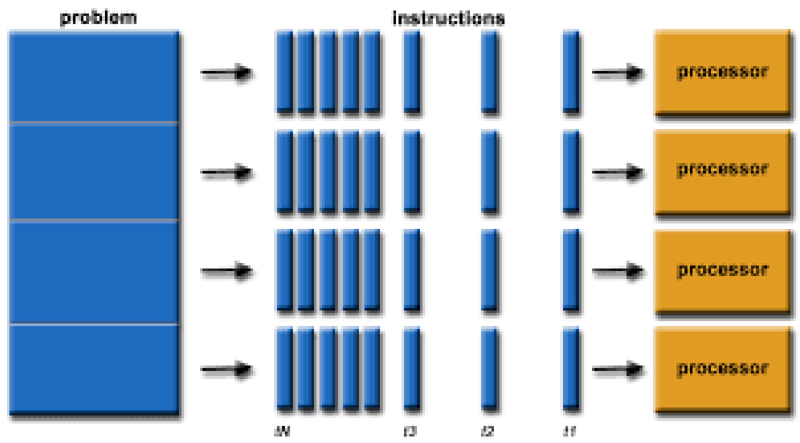
1. **What is Parallel Programming?**

   Definition: Parallel programming, in simple terms, is the process of decomposing a problem into smaller tasks that can be executed at the same time using multiple compute resources.

   Significance: In parallel programming, tasks are parallelized so that they can be run at the same time by using multiple computers or multiple cores within a CPU. Parallel programming is critical for large scale projects in which speed and accuracy are needed. It is a complex task, but allows developers, researchers, and users to accomplish research and analysis quicker than with a program that can only process one task at a time.

   Working: Parallel programming works by assigning tasks to different nodes or cores. In High Performance Computing (HPC) systems, a node is a self-contained unit of a computer system containing memory and processors running an operating system. Processors, such as central processing units (CPUs) and graphics processing units (GPUs), are chips that contain a set of cores. Cores are the units executing commands; there can be multiple cores in a processor and multiple processors in a node.

   Parallel processing techniques can be utilized on devices ranging from embedded, mobile, laptops, and workstations to the world's largest supercomputers. Different computer languages provide various technologies to enable parallelism. For C, C++ and Fortran, OpenMP, open multi-processing, provides a cross-platform API for developing parallel applications that enable running parallel tasks across cores of a CPU. When processes need to communicate between different computers or nodes, a technology such as MPI, message passing interface, is typically used.



2. **Why is Parallel programming important?**

- The whole real-world runs in dynamic nature i.e. many things happen at a certain time but at different places concurrently. This data is extensively huge to manage.

- Real-world data needs more dynamic simulation and modeling, and for achieving the same, parallel computing is the key.

- Parallel computing provides concurrency and saves time and money.

- Complex, large datasets, and their management can be organized only and only using parallel computing's approach.

- Ensures the effective utilization of the resources. The hardware is guaranteed to be used effectively whereas in serial computation only some part of the hardware was used and the rest rendered idle.

- Also, it is impractical to implement real-time systems using serial computing.

Advantages of Parallel Computing over Serial Computing are as follows:

1. It saves time and money as many resources working together will reduce the time and cut potential costs.
2. It can be impractical to solve larger problems on Serial Computing.
3. It can take advantage of non-local resources when the local resources are finite.
4. Serial Computing 'wastes' the potential computing power, thus Parallel Computing makes better work of the hardware.

Significance: In parallel programming, tasks are parallelized so that they can be run at the same time by using multiple computers or multiple cores within a CPU. Parallel programming is critical for large scale projects in which speed and accuracy are needed. It is a complex task, but allows developers, researchers, and users to accomplish research and analysis quicker than with a program that can only process one task at a time.

3. Introduction to OpenMP and its features.

OpenMP is an Application Program Interface (API) that may be used to explicitly direct multi-threaded, shared memory parallelism in C/C++ programs. It is not intrusive on the original serial code in that the OpenMP instructions are made in pragmas interpreted by the compiler.

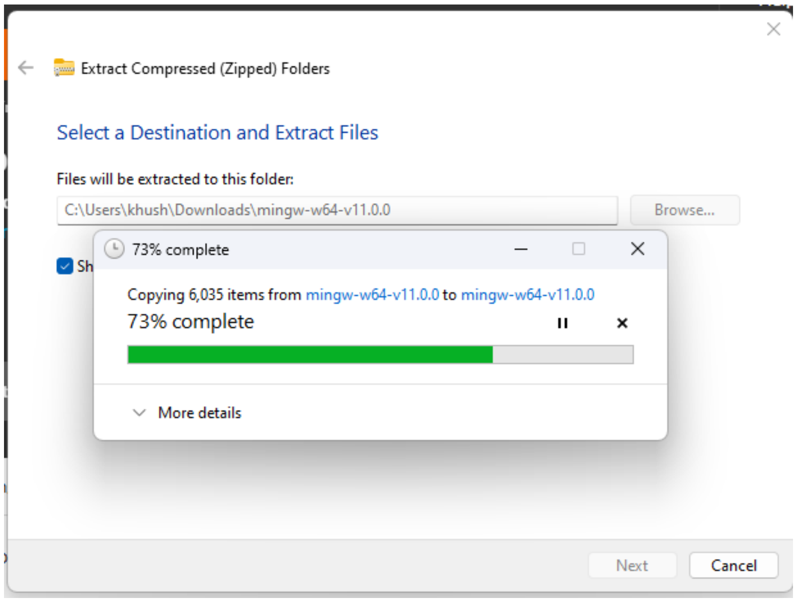
OpenMP uses the fork-join model of parallel execution. All OpenMP programs begin with a single master thread which executes sequentially until a parallel region is encountered, when it creates a team of parallel threads (FORK). When the team threads complete the parallel region, they synchronize and terminate, leaving only the master thread that executes sequentially (JOIN).

Key features of OpenMp are:

1. **Directive-based approach:** OpenMP uses compiler directives, which are special annotations in the source code, to indicate the regions of code that should be executed in parallel.
2. **Shared-memory model:** OpenMP is designed for shared-memory architectures, where multiple threads can access the same memory space.
3. **Parallel loops**: The most common use of OpenMP is for parallelizing loops. By using loop constructs like #pragma omp parallel for, iterations of a loop can be distributed among multiple threads, enabling parallel execution.
4. **Parallel regions**: OpenMP allows you to define parallel regions using #pragma omp parallel, where the enclosed code will be executed by multiple threads.
5. **Thread synchronization**: OpenMP provides mechanisms to synchronize threads, such as #pragma omp barrier, which ensures that all threads reach a specific point in the code before continuing.
6. **Work-sharing constructs**: In addition to parallel loops, OpenMP supports work-sharing constructs like #pragma omp sections and #pragma omp single, which allow different threads to handle specific tasks.
7. **Data scope attributes:** OpenMP provides ways to specify how variables are shared or private to threads using data scope attributes.
8. **Nested parallelism:** OpenMP allows you to create nested parallel regions, meaning you can have parallel regions inside other parallel regions.
9. **Environment variables:** OpenMP defines environment variables that can control the number of threads, thread affinity, and other runtime behaviors.
10. **Runtime library routines**: OpenMP includes a set of runtime library routines that can be used in the code to query or set parameters related to parallelism.

2. Setting Up the Development Environment.

1. **Installing OpenMp libraries and compilers.**



Extract Compressed (Zipped) Folders

Select a Destination and Extract Files

Files will be extracted to this folder:

C:\Users\khush\Downloads\mingw-w64-v11.0.0

73% complete

Copying 6,035 items from mingw-w64-v11.0.0 to mingw-w64-v11.0.0

73% complete

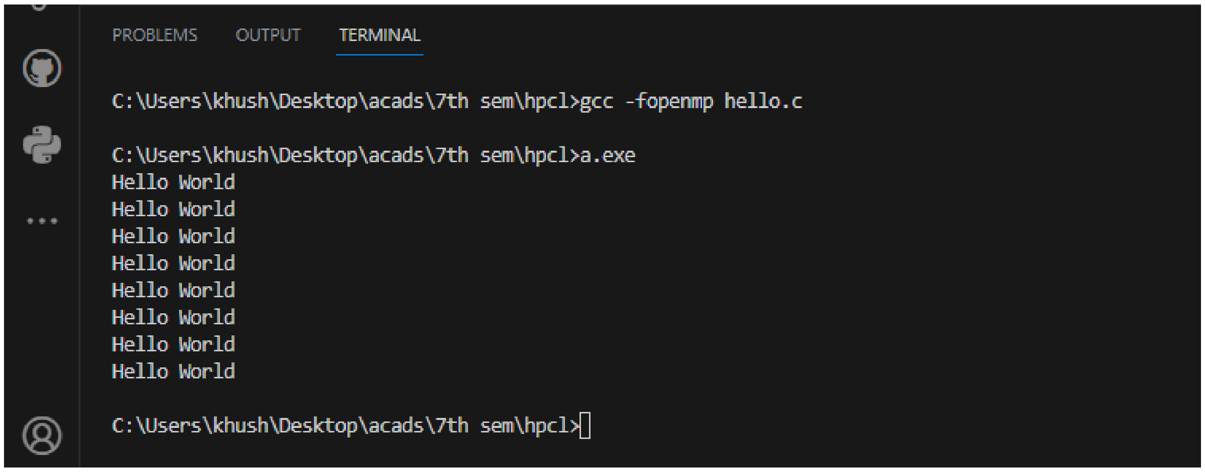
3. Basic OpenMP Constructs.

1. **Hello World program with OpenMP. Using #pragma omp parallel.**

CODE:

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main(int argc, char *argv[]){
    #pragma omp parallel
    printf("%s\n", "Hello World");

    return 0;
}
```

OUTPUT:



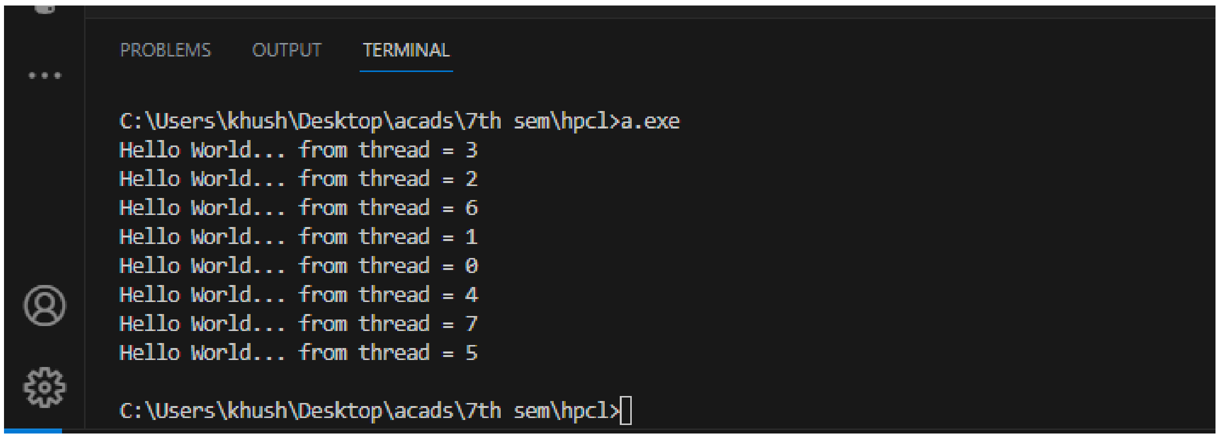
```
C:\Users\khush\Desktop\acads\7th sem\hpcl>gcc -fopenmp hello.c

C:\Users\khush\Desktop\acads\7th sem\hpcl>a.exe
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World

C:\Users\khush\Desktop\acads\7th sem\hpcl>
```

2. **Thread identification using omp_get_thread_num().**

CODE:

```c
// OpenMP program to print Hello World
// using C language

// OpenMP header
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{

    // Beginning of parallel region
#pragma omp parallel
    {

        printf("Hello World... from thread = %d\n",
            omp_get_thread_num());
    }
    // Ending of parallel region
}
```

OUTPUT:



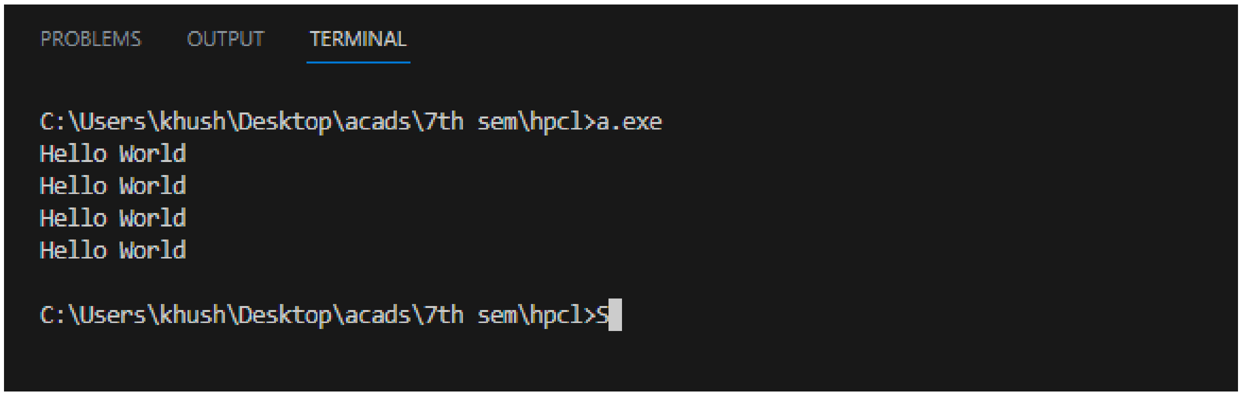
```
C:\Users\khush\Desktop\acads\7th sem\hpcl>a.exe
Hello World... from thread = 3
Hello World... from thread = 2
Hello World... from thread = 6
Hello World... from thread = 1
Hello World... from thread = 0
Hello World... from thread = 4
Hello World... from thread = 7
Hello World... from thread = 5

C:\Users\khush\Desktop\acads\7th sem\hpcl>
```

3. Using omp_get_num_threads().

CODE:

```c
//%compiler: clang
//%cflags: -fopenmp

#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main(int argc, char *argv[]){
    #pragma omp parallel num_threads(4)
    printf("%s\n", "Hello World");

    return 0;
}
```

OUTPUT:



```
C:\Users\khush\Desktop\acads\7th sem\hpcl>a.exe
Hello World
Hello World
Hello World
Hello World

C:\Users\khush\Desktop\acads\7th sem\hpcl>S
```