

Name: OG RAGAVI

Reg No: 20BCE1988

Lab: 8

Steps:

Initial set up

```
sudo apt-get update
sudo apt-get install libopenmpi-dev
sudo apt-get install openmpi-bin
sudo apt-get install mpich
```

1.Hello World Program

```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char** argv) {
    // Initialize the MPI environment
    MPI_Init(NULL, NULL);

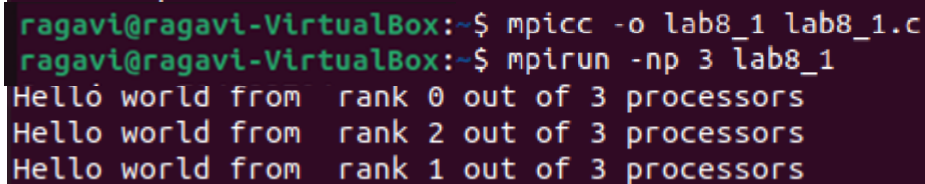
    // Get the number of processes
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Print off a hello world message
    printf("Hello world from rank %d out of %d processors\n", world_rank,
world_size);

    // Finalize the MPI environment.
    MPI_Finalize();
}
```

output:



```
ragavi@ragavi-VirtualBox:~$ mpicc -o lab8_1 lab8_1.c
ragavi@ragavi-VirtualBox:~$ mpirun -np 3 lab8_1
Hello world from rank 0 out of 3 processors
Hello world from rank 2 out of 3 processors
Hello world from rank 1 out of 3 processors
```

2. Sum of Array Program

CODE:

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
// size of array
#define n 10
int a[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
// Temporary array for slave process
int a2[1000];
int main(int argc, char* argv[])
{
    int pid, np, elements_per_process, n_elements_recieved;
    // np -> no. of processes
    // pid -> process id
    MPI_Status status;

    // Creation of parallel processes
    MPI_Init(&argc, &argv);

    // find out process ID,
    // and how many processes were started
    MPI_Comm_size(MPI_COMM_WORLD, &np);
    MPI_Comm_rank(MPI_COMM_WORLD, &pid);

    // master process
    if (pid == 0) {
        int index, i;
        elements_per_process = n/np;

        // check if more than 1 processes are run
        if (np > 1) {
            // distributes the portion of array
            // to child processes to calculate
            // their partial sums
            for (i = 1; i < np - 1; i++) {
                index = i * elements_per_process;
                MPI_Send(&elements_per_process, 1, MPI_INT, i, 0, MPI_COMM_WORLD);
```

```

MPI_Send(&a[index], elements_per_process, MPI_INT, i, 0,
MPI_COMM_WORLD);
}

// last process adds remaining elements
index = i * elements_per_process;
int elements_left = n - index;

MPI_Send(&elements_left, 1, MPI_INT, i, 0, MPI_COMM_WORLD);

MPI_Send(&a[index], elements_left, MPI_INT, i, 0, MPI_COMM_WORLD);
}

// master process add its own sub array
int sum = 0;
for (i = 0; i < elements_per_process; i++)
sum += a[i];

// collects partial sums from other processes
int tmp;
for (i = 1; i < np; i++) {
MPI_Recv(&tmp, 1, MPI_INT, MPI_ANY_SOURCE,
0, MPI_COMM_WORLD, &status);
int sender = status.MPI_SOURCE;
sum += tmp;
}

// prints the final sum of array
printf("Sum of array is : %d\n", sum);
}

// slave processes
else {
MPI_Recv(&n_elements_recieved, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
&status);

// stores the received array segment
// in local array a2
MPI_Recv(&a2, n_elements_recieved, MPI_INT, 0, 0, MPI_COMM_WORLD,
&status);

// calculates its partial sum
int partial_sum = 0;
for (int i = 0; i < n_elements_recieved; i++)

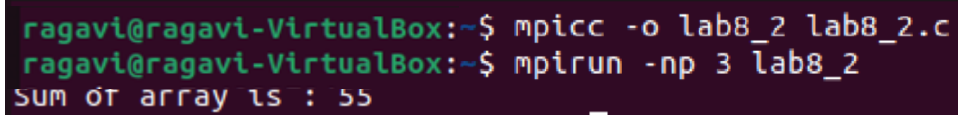
```

```
partial_sum += a2[i];

// sends the partial sum to the root process
MPI_Send(&partial_sum, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
}
// cleans up all MPI state before exit of process
MPI_Finalize();

return 0;
}
```

output:



```
ragavi@ragavi-VirtualBox:~$ mpicc -o lab8_2 lab8_2.c
ragavi@ragavi-VirtualBox:~$ mpirun -np 3 lab8_2
sum of array is : 55
```