BCSE307P – Compiler Design Lab

Winter Semester 2023-24

Assessment 12

OpenMP Programming

Name: Sujay Ghosh

Reg. No: 21BLC1607

Slot: L7 + L8

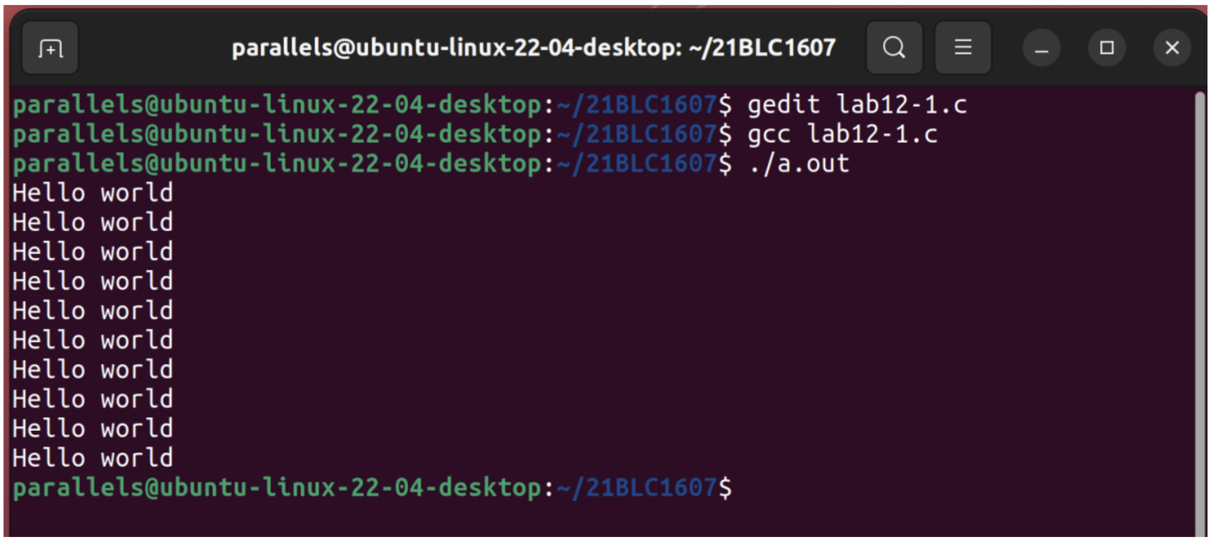
Faculty: Dr. Rathna

Pgm 1:

Code:

```
#include <stdio.h>

#include <omp.h>


void main() {

	#pragma omp parallel for

	for (int i=0; i<10; i++) {

		printf("Hello world\n");

	}

}
```

Output:



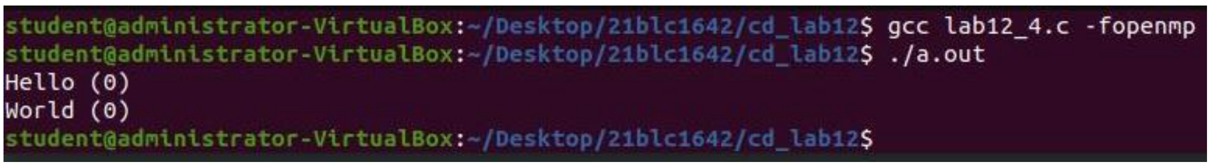
```
parallels@ubuntu-linux-22-04-desktop:~/21BLC1607$ gedit lab12-1.c
parallels@ubuntu-linux-22-04-desktop:~/21BLC1607$ gcc lab12-1.c
parallels@ubuntu-linux-22-04-desktop:~/21BLC1607$ ./a.out
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
parallels@ubuntu-linux-22-04-desktop:~/21BLC1607$
```

Pgm 2:

Code:

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



int main() {
    #pragma omp parallel
    {
        int ID = omp_get_thread_num();
        printf("Hello (%d)\n", ID);
        printf("World (%d)\n", ID);
    }
}
```

Output:



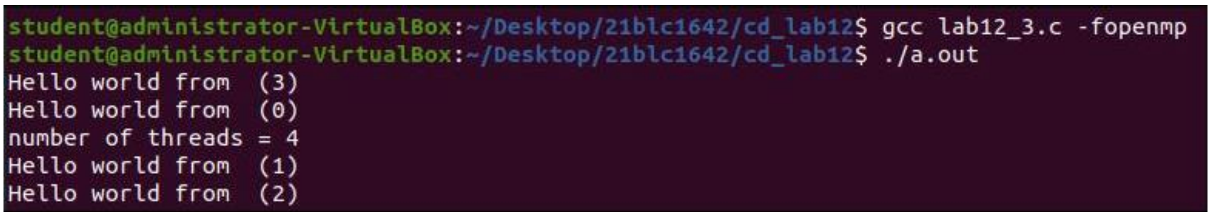
```
student@administrator-VirtualBox:~/Desktop/21blc1642/cd_lab12$ gcc lab12_4.c -fopenmp
student@administrator-VirtualBox:~/Desktop/21blc1642/cd_lab12$ ./a.out
Hello (0)
World (0)
student@administrator-VirtualBox:~/Desktop/21blc1642/cd_lab12$
```

Pgm 3:

Code:

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



int main() {
    int nthreads, tid;
    #pragma omp parallel num_threads(4) private(tid)
    {
        tid = omp_get_thread_num();
        printf("Hello world from (%d)\n", tid);
        if (tid == 0) {
            nthreads = omp_get_num_threads();
            printf("number of threads = %d\n", nthreads);
        }
    }
}
```

Output:



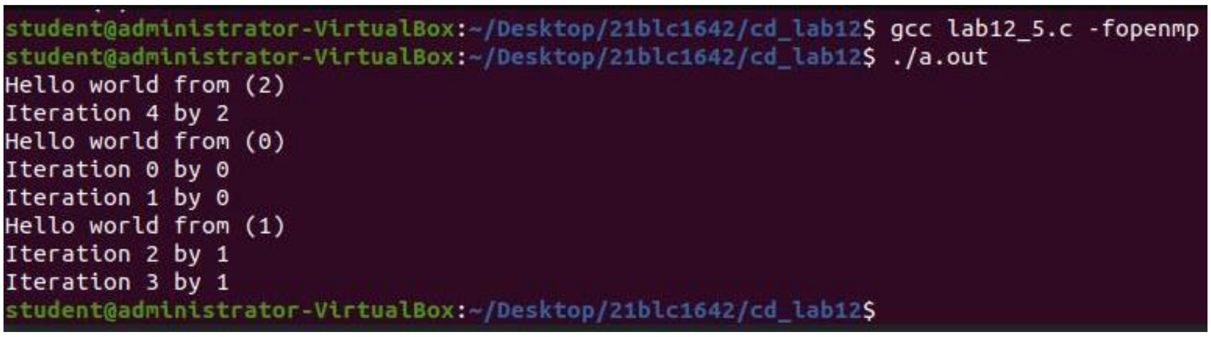
```
student@administrator-VirtualBox:~/Desktop/21blc1642/cd_lab12$ gcc lab12_3.c -fopenmp
student@administrator-VirtualBox:~/Desktop/21blc1642/cd_lab12$ ./a.out
Hello world from  (3)
Hello world from  (0)
number of threads = 4
Hello world from  (1)
Hello world from  (2)
```

Pgm 4:

Code:

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



int main() {
   int nthreds, tid;
   omp_set_num_threads(3);

   #pragma omp parallel private(tid)
   {
      int i;
      tid = omp_get_thread_num();
      printf("Hello world from (%d)\n", tid);
      #pragma omp for
      for (i=0; i<=4; i++) {
         printf("Iteration %d by %d\n", i, tid);
      }
   }
}
```

Output:



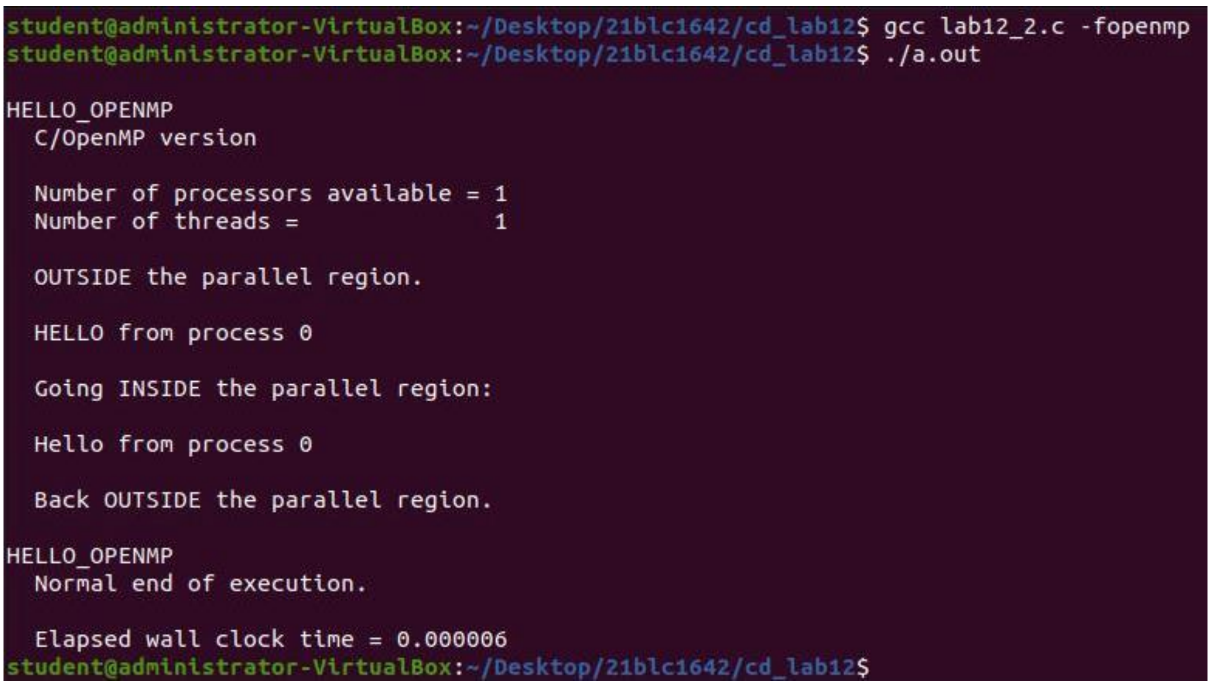
```
student@administrator-VirtualBox:~/Desktop/21blc1642/cd_lab12$ gcc lab12_5.c -fopenmp
student@administrator-VirtualBox:~/Desktop/21blc1642/cd_lab12$ ./a.out
Hello world from (2)
Iteration 4 by 2
Hello world from (0)
Iteration 0 by 0
Iteration 1 by 0
Hello world from (1)
Iteration 2 by 1
Iteration 3 by 1
student@administrator-VirtualBox:~/Desktop/21blc1642/cd_lab12$
```

Pgm 5:

Code:

```
#include<stdio.h>
#include<omp.h>
void main()
{
int id;
  double wtime;



  printf ( "\n" );
  printf ( "HELLO_OPENMP\n" );
  printf ( "  C/OpenMP version\n" );



  printf ( "\n" );
  printf ( "  Number of processors available = %d\n",
omp_get_num_procs ( ) );
  printf ( "  Number of threads =              %d\n",
omp_get_max_threads ( ) );



  wtime = omp_get_wtime ( );



  printf ( "\n" );
  printf ( "  OUTSIDE the parallel region.\n" );
  printf ( "\n" );



  id = omp_get_thread_num ( );
  printf ( "  HELLO from process %d\n", id ) ;



  printf ( "\n" );
  printf ( "  Going INSIDE the parallel region:\n" );
  printf ( "\n" );
/*
  INSIDE THE PARALLEL REGION, have each thread say hello.
*/
# pragma omp parallel \
  private ( id )
  {
    id = omp_get_thread_num ( );
    printf ("  Hello from process %d\n", id );
  }
/*
  Finish up by measuring the elapsed time.
*/
  wtime = omp_get_wtime ( ) - wtime;



  printf ( "\n" );
  printf ( "  Back OUTSIDE the parallel region.\n" );
/*
  Terminate.
*/
  printf ( "\n" );
  printf ( "HELLO_OPENMP\n" );
  printf ( "  Normal end of execution.\n" );



  printf ( "\n" );
  printf ( "  Elapsed wall clock time = %f\n", wtime );



}
```

<u>Output:</u>



```
student@administrator-VirtualBox:~/Desktop/21blc1642/cd_lab12$ gcc lab12_2.c -fopenmp
student@administrator-VirtualBox:~/Desktop/21blc1642/cd_lab12$ ./a.out

HELLO_OPENMP
  C/OpenMP version

  Number of processors available = 1
  Number of threads =              1

  OUTSIDE the parallel region.

  HELLO from process 0

  Going INSIDE the parallel region:

  Hello from process 0

  Back OUTSIDE the parallel region.

HELLO_OPENMP
  Normal end of execution.

  Elapsed wall clock time = 0.000006
student@administrator-VirtualBox:~/Desktop/21blc1642/cd_lab12$
```

Result:

Thus, the experiment has been successfully executed and verified.