***Lab-1 Screenshots***

Compiling and running the given sum-pthd.c program to check the output.

A picture containing chart

Description automatically generated

**Question-1)**

Graphical user interface, text

Description automatically generated

Compiling and running the sum-oo.cpp program to check the output: -

Text

Description automatically generated

**Question-2)** Text

Description automatically generated

Compiling and running the Sum.java program to check the output.

Text

Description automatically generated

**Question-3)**

Text

Description automatically generated

When the program is run with the script with 2000 as an argument, we get the following result: - (NOTE: - In the result, only the lines which produce wrong output are printed for better readability of the output).

This is because as the number of processes (java Sum) increases, the number of threads increases. Although each thread might have their own local variables, the global variables are shared. This leads to threads producing wrong results due to “race condition”.

**Question-4)**

Now when synchronized keyword is applied to provide a locking mechanism for the global variable sum, then the following results are produced with the same automate script and same input of 2000.

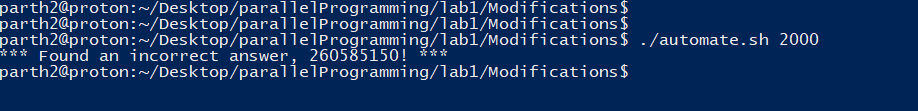
Here also, only the incorrect sum will be printed (if incorrect sum is calculated).

Text

Description automatically generated

As we can see that the program runs perfectly.

When we remove the static keyword from the declaration of the lock (lck) object, then, there is an incorrect sum found.



This is because when static keyword is removed, then all the threads can change the value of lock (lck) even when in execution. This will defeat the purpose of declaring a lock and hence race condition is not removed for the global variable sum.

**Question-5)**

To print the thread name, we will use the Thread.currentThread().getName() is used. The output for the same is given below: -

A picture containing diagram

Description automatically generated

**Question-6)**

Compiling and running the sum-omp program, we get the following output: -

**Graphical user interface, text

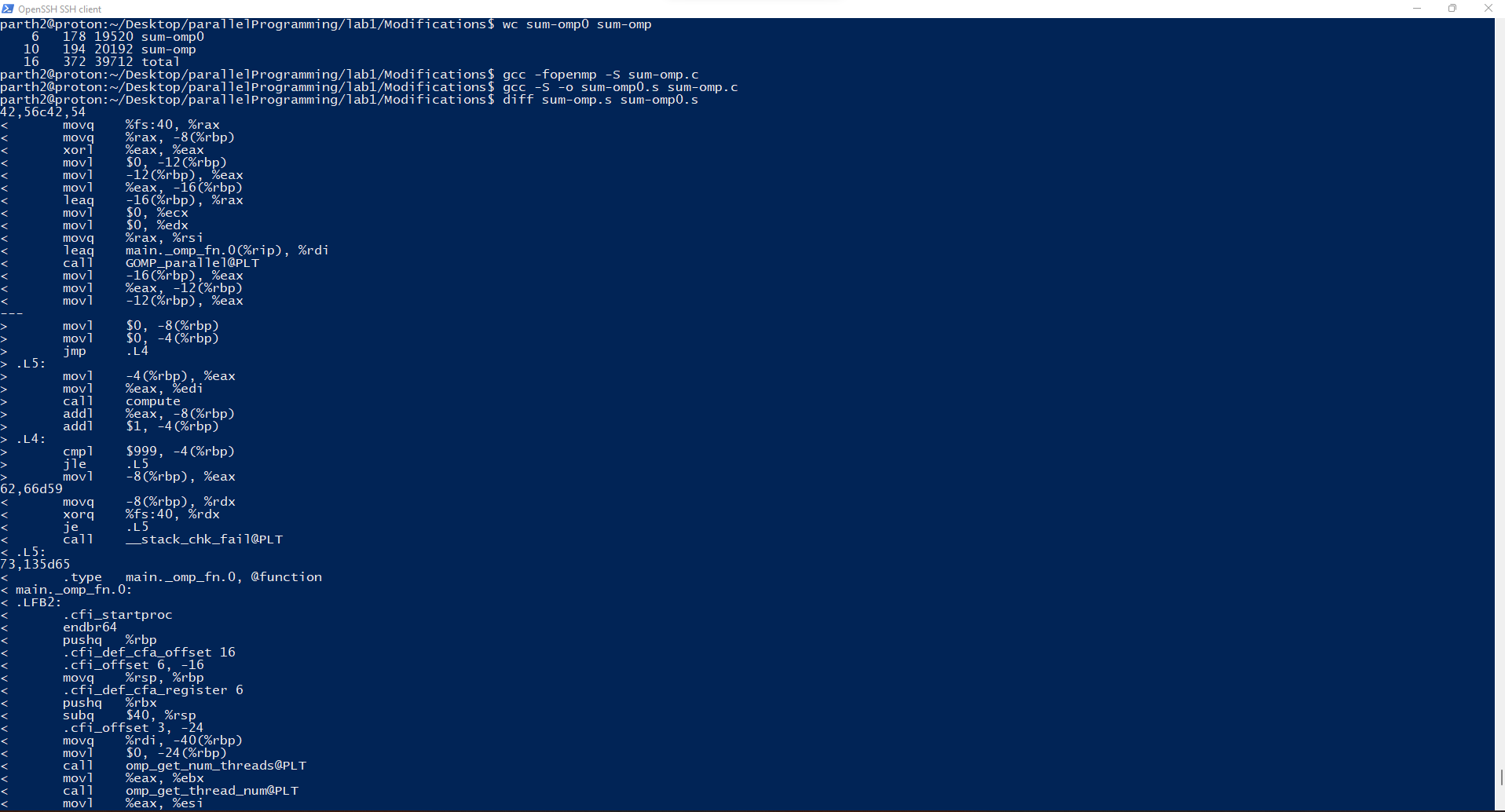
Description automatically generated**

Now, removing the -fopenmp flag and then comparing the wc, we get,

Text

Description automatically generated

The next step is to compare the assembly code: -



**Question-7)**

Inserting the required print statement in the for loop, we get the following output which contains the thread ids for the threads accessing the shared variable on each iteration. This gives us the full picture of how OpenMp functions

Shape, rectangle

Description automatically generated

Shape

Description automatically generated

**Question-8)**

After adding the required exports in the .bash\_profile and .bashrc files, we will compile and run the sum-mpi.c and get the following output: -

Text

Description automatically generated

On adding the required statements for printing the rank and size, we have: -

Text

Description automatically generated with medium confidence

**Question-9)**

After making the necessary changes in the .bash\_profile for running the chapel programs, we will compile the sum1.chpl and sum2.chpl programs.

Text

Description automatically generated

Then, we will run the Sum1.chpl with one locale as given below:-

Text

Description automatically generated

Then, we will run the sum2 program with the number of locales as 4: -

Text

Description automatically generated

Now, after adding the writeln statement in sum2.chpl, we get the following output: -

A picture containing text

Description automatically generated

A picture containing background pattern

Description automatically generated

Text

Description automatically generated

Also, when you close the connection to the server, you could see the connections being closed on all servers. It will ask to close 4 times

Text

Description automatically generated