CS 3113 Intro to Operating Systems

Name and ID ___________________________

Homework #3

Instructions:
1) To complete assignment one, you need to read Chapters 1, 3 and 4 of the textbook.
2) HW must be submitted on typed pdf or word document.
<u>You must do your work on your own.</u>

Q1. Using Amdahl's Law, calculate the speedup gain of an application that has a 60 percent parallel component for (a) two processing cores and (b) four processing cores. (15 points)

a) Speed up= 1 / (0.4+(0.6/2)) =1.429

b) Speed up= 1/(0.4+0.6/4)=1.818

Q2. A system with two dual-core processors has four processors available for scheduling. A CPU-intensive application is running on this system. All input is performed at program start-up, when a single file must be opened. Similarly, all output is performed just before the program terminates, when the program results must be written to a single file. Between start-up and termination, the program is entirely CPU-bound. Your task is to improve the performance of this application by multithreading it. The application runs on a system that uses the one-to-one threading model (each user thread maps to a kernel thread). (20 points)

a. How many threads will you create to perform the input and output? Explain.
b. How many threads will you create for the CPU-intensive portion of the application? Explain.

a) one thread for input and one thread for output. These operations are sequential and cannot happen at the same time and would not benefit from parallelism so they need one thread each. The input thread will read the input file, this is an operation that does not require many threads so one thread is sufficient, and the output thread will write the results to the file and one thread is enough for output. These two operations cannot happen at the same time so two threads can perform this operation with each operation getting a dedicated thread.

b) the system has a total of four cores available and since the function is CPU-intensive, it is better to use all of them to increase efficiency. Since we have four cores, we can create 4 threads so that we use all the cores available. Each core can handle one thread, so for each core we can create one thread and increase efficiency of the program.

Q3. Consider the following code segment: (15 points)

```
pid t pid;

pid = fork();
if (pid == 0) { /* child process */
          fork();
          thread create( . . .);  /* for the purpose of this problem, you can ignore the lack
of arguments to the function */
}
fork();
```

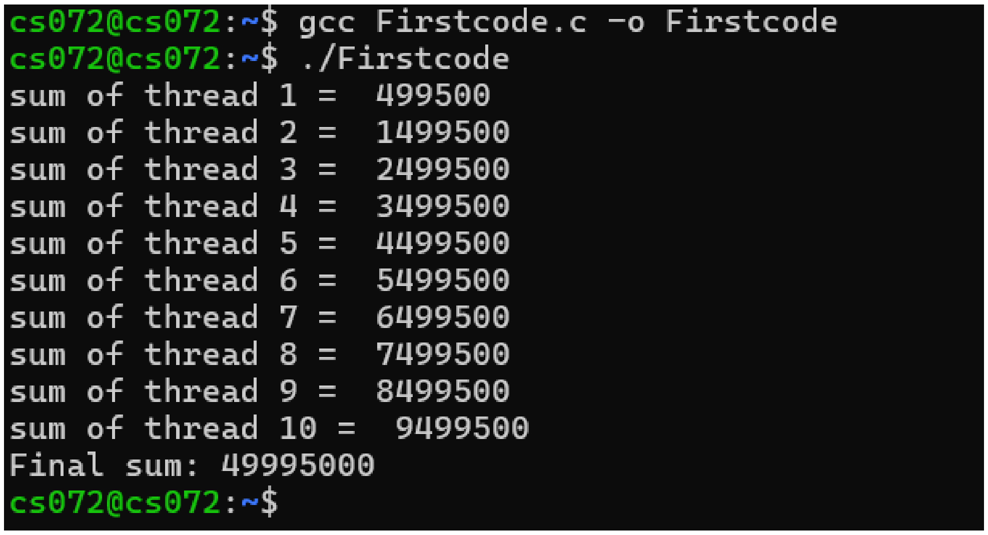
a. How many unique processes are created?
b. How many unique threads are created?

a) 5 processes are created including the parent process, the main function, 2 processes after the first fork, 1 process when the child forks and 2 more with the last fork bringing the total up to five.

b) The main function has its own thread but the child process calls the thread_create() function explicitly so there are two threads but only the child process explicitly creates a thread using the function so 1 unique thread is created.

Q4. Pthread programming: writing a program to join on ten threads for calculating 0-9999. Each thread calculates the sum of 1000 numbers. Please attach screenshots of your execution results below. You also need to submit your code (along with a readme file) separately. (50 points)

All files (MUST INCLUDE: source codes, a readme file, and homework 3) should be zipped together.



```
cs072@cs072:~$ gcc Firstcode.c -o Firstcode
cs072@cs072:~$ ./Firstcode
sum of thread 1 =  499500
sum of thread 2 =  1499500
sum of thread 3 =  2499500
sum of thread 4 =  3499500
sum of thread 5 =  4499500
sum of thread 6 =  5499500
sum of thread 7 =  6499500
sum of thread 8 =  7499500
sum of thread 9 =  8499500
sum of thread 10 =  9499500
Final sum: 49995000
cs072@cs072:~$
```