Purpose: This assignment will explore the concept of concurrency and multi-threaded programming.

You will not need to write a program for this assignment, download the code provided and follow the instructions below.

Answer the questions below in a document file (.docx/.odt).

Instructions

We will begin by examining the state of the virtual machine/hardware you are using. Check under the settings of the virtual machine. Enable as many cores as you can for the virtual machine, the virtual machine may need to be shut down in order to change this setting.

Question 1. **How many CPU cores are being used on the virtual machine?**

Question 2. **How many CPU cores exist on the physical computer?**

Download and assemble the provided code for this assignment. You will need to use the following commands:

```
yasm -g dwarf2 -f elf64 assignment10_complete.asm

gcc -no-pie assignment10_complete.o -pthread

time ./a.out
```

Note the addition of -pthread in the call to the c compiler.

Question 3. **How much real time does it take for the program to execute?**

Open the code in a source code editor. Change the constant "THREAD_COUNT" to be equal to 1. Reassemble and run the program again.

Question 4. **How much real time does it take for the program to execute with only one thread?**

Question 5. **What is the speedup factor of using 6 threads versus 1? Speedup = Time for 1 Thread / Time for 6 Threads**

Question 6. **Assuming the entire program could be run in parallel, what is the theoretical maximum speedup you would expect from using six threads versus 1?**

Set the "THREAD_COUNT" constant back to 6. In the "findAbundantNumbers" function, comment out the "call obtainLock" instruction.

Reassemble and run the program a few times.

Question 7. **How does the number of abundant values found compare to when the instruction was not commented out?**

Question 8. **Briefly explain what might be causing the difference in results between calling obtainLock and not.**

Uncomment the "call obtainLock" instruction.

Comment out the "lock inc qword[abundantCount]" in the "findAbundantNumbers" function and uncomment the "inc qword[abundantCount]" instruction.

Reassemble and run the program several times.

Question 9. **How does the number of abundant values found compare to the original?**

Question 10. **Again, briefly explain what is causing the difference in results when the increment is not locked. Set the "THREAD_COUNT" constant to 100. Reassemble and run the program again.**

Question 11. **How does the program performance compare between 100 and 6 threads? Explain why it does not have the same increase in performance compared to 1 and 6 threads.**

Submission
Upload the document (.docx, .odt) file with your answers to the class website.