CS 218
Homework, Asst. #12

Purpose: Become more familiar with operating system interaction and race conditions.
Due: Wednesday (6/28)
Points: 80 50 for program and 30 for write-up
Grading will include functionality, documentation, and coding style

Assignment:
Write an assembly language program to compute the provided formula LIMIT (defined constant) times. The program should calculate the formula either sequentially or in parallel (based on the command line argument). This is done by computing the formula in two threads (each thread performing LIMIT/2 computations).

I'M JUST OUTSIDE TOWN, SO I SHOULD BE THERE IN FIFTEEN MINUTES.
ACTUALLY, IT'S LOOKING MORE LIKE SIX DAYS.
NO, WAIT, THIRTY SECONDS.
THE AUTHOR OF THE WINDOWS FILE COPY DIALOG VISITS SOME FRIENDS.

The provided template performs some of the basic actions including declaring appropriate constants/variables/string and displaying some header messages (see example output).

Create a function, ***int2ternary***, to:

- Convert an integer into an ASCII ternary string (NULL terminated)

Create two thread functions, ***threadFunction0()***, and ***threadFunction1()***, where each perform the following actions:

- Display a simple start message (predefined)
- Compute the following formula LIMIT/2 times on a global variable:

$$\boldsymbol{myValue = \frac{myValue * A}{B} + C}$$

Note, do not simplify or alter the provided formula and treat all values as *unsigned*.

Update the provided main template to add the following functionality:

- Command line argument, with error checking.
- Sequential Computation (for *myValue*)
  - Start a thread for thread function 0, wait for thread function 0 to complete
  - Start a thread for thread function 1, wait for thread function 1 to complete
  - Display results (final value of *myValue*)
- Parallel (i.e., threaded) Computation (for *myValue*)
  - Start a thread for thread function 0
  - Start a thread for thread function 1
  - Wait for thread function 0 to complete
  - Wait for thread function 1 to complete
  - Display results (final value of *myValue*)

Review and explain the results for both the sequential and parallel operations. Change the LIMIT constant from 0x70000000 to 0x64 and re-execute the program. Include those results in the final explanation (which is submitted separately).

Submission:
When complete, submit:

- *A copy of the* ***source file*** *via the class web page by class time.*
  Assignments received after the due date/time will not be accepted. [50 points]
- Submit a write-up of the program results, including a copy of the program output and an explanation of the results. The program results can be captured using the provided timing script. The explanation must address the final value of ***myValue*** for the sequential and parallel operations (for both for 0x64 and 0x70000000). The explanation should be less than 150 words. Overly long explanations will be not be scored. [30 points]

Example Execution:
The following is an example execution for the non-threaded version:

```
ed-vm% ./ast12 -sq
*******************************************
Program Start

CPU Cores: 8
Final Counter Value should be: 11211221202112112111


--------------------------------------
Compute Formula -> Sequential
 ...Thread 0 starting...
 ...Thread 1 starting...
Sequential Counter -> Final Value: 11211221202112112111


Completed.
```

Note, the '**`ed-vm%`**' is the prompt and should not be typed. The actual number of cores will vary.

Thread Functions
The following are the function calls required for thread management (create thread and wait for thread to complete).

```
;   pthread_create(&pthreadID0, NULL, threadFunction0, NULL);
        mov     rdi, pthreadID0
        mov     rsi, NULL
        mov     rdx, threadFunction0
        mov     rcx, NULL
        call    pthread_create

;   pthread_join (pthreadID0, NULL);
        mov     rdi, qword [pthreadID0]
        mov     rsi, NULL
        call    pthread_join
```