Programming Assignment II

(Due date: See course's website)

In this project, you will be implementing a per-thread cycle accurate timer as a new system call in Windows CE .Net. This is slightly different than the existing system call *SC_GetThreadTimes* because the existing system call returns time in 100 nanoSeconds increment whereas the new system call returns the number of cycles. The new system call will work exclusively with Intel Performance Counter available in x686 or higher (PII and up or K6 and up). This special instruction is available as part of the Simulator provided by the platform builder in CE. You will need to revise Windows CE source code. You will also need to rely on *OEMIoControl* to create the new System Call, which is similar to the process in lab 5. Please refer to the manual of lab 5 on how to create a system call. The system call is specified as follows:

- The system call, *SC_GetThreadTickCount* is used to obtain the number of cycles that each process has been executing. This execution time **must not** include the time that a thread is block state. The interface can be as follows:

  ```
  SC_GetThreadTickCount(HANDLE hThread, __int64 *myTickCount)
  ```

- User *KernelIoControl* as an interface of the system call, `KernelIoControl(-3366, (LPVOID)hid, 0, (LPVOID)(&count), 0, 0)`, where -3366 is a user defined number as an entry in *OEMIoControl*, hid (HANDLE) is the thread handler of the thread, count (unsigned __int64) is used to store the number of cycles that this thread has been executed.

You will need to use *rdtsc* in assembly language to obtain the number of cycles. More information about *rdtsc* instruction is provided as a separate document. You will need to add two additional fields in the thread structure: one for recording the thread *starting_time* and the other for recording the *accumulated_time*. By time, we mean the time in terms of the number of cycles. After you create these two fields, you will need to:

- Assign zero to these two fields when initializing the thread structure.
- Add a function in *schedule.c* which returns the total number of cycles that this thread has been used in its running state.
- Sum up the accumulated number of cycles when the thread is swapped out from running state. This requires a careful reading and analysis of the source code (schedule.c).
- You also need to make certain that this function should return the accumulated time (up to the last time the thread was swapped out) and the number of cycles in the current turn on the processor.

Some hints

- Data type: You need to use unsigned __int64 as the storage of the number of cycles.
- Assembly in C language: You need to use assembly in C function to obtain the number of rdtsc cycles. The basic format is:

```
asm {    assembly instruction }
```

Submission Procedure:

- Lab demo (will take place on the due date).
  1. We will provide a sample test program.
  2. Before the end of execution of thread, call the system call to get the thread execution cycles and print it out
- Your documentation should include.
  - Source files
    1. All modified source files and header files
  - Project report that include:
    1. Difficulties encountered
    2. Workload distribution
    3. Your approach to the problem
    4. Number of hours spent on the project
    5. All the necessary steps to complet the project

NOTICE: I will not take late submission.