Gebze Technical University
Department Of Computer Engineering
CSE 312 / CSE 504

Operating Systems

Homework #02
Due Date: Apr 20th 2018
System Calls and Threads

In this homework, we will use our 8080 Emulator with a more advanced OS that supports threads. Our OS now can run more than one thread using multitasking techniques that we saw in the lectures.

Our new OS will offer the following system calls in addition to the calls of HW1. Note that you should update your ASM files to include the definitions of the new system calls.

Call	**Params to pass**	**Function**	**How many cycles**
TCreate	Register A =11 The thread function pointer is at address BC.	It works like the pthread_create call of UNIX systems. It returns the thread ID in register B on success. On fail, returns 0. After this call, a new thread is created and it starts running the function passed as parameter.	80
TExit	Register A = 8, Register B holds the exit status	It works like the pthread_exit call of UNIX systems. It terminates the calling thread.	50
TJoin	Register A = 9 Register B holds the thread ID of the thread to wait for.	Blocks the calling thread until the thread with thread ID is terminated.	40
TYield	Register A = 10	Causes the calling thread to yield the CPU. The thread is placed at the end of the schedule queue.	40

Since you have more than one thread in the memory, you need to keep a thread table, the thread table will hold at least the following properties for each thread

- The saved register set of 8080 chip
- The thread ID
- Starting time of the thread (the cycle number of the CPU)
- How many cycles the thread has used so far
- The state of the thread (ready, blocked, running)
- The starting address of the memory location of the thread function

- Available empty stack space

Any other data structure can be added to the thread table as needed.

When your simulated computer starts running, it will first read a program file and start running the program.

For this homework, you will do the following

- Update your OS class with the system calls
- Write a robin robin scheduler (quantum time: 100 cycles) that keeps the thread table updated and does the thread switching
- Your new system should automatically choose stack location for the threads created.

You will write 5 ASM programs for your new system. These programs will use some variations of the following thread functions, which should be reentrant. Your functions should call common helper functions such as sort, printNum and search. All of these functions should end with a call to TExit.

- F1: A function that prints numbers from 0 to 50
- F2: A function that adds numbers from 1 to 1000 and prints the result
- F3: A function that prints numbers from 50 to 100
- F4: A function produces 50 random bytes, sorts them in increasing order and prints them on screen
- F5: A function that produces 50 random bytes and sorts them. Your function then searches the 1st, 10th, and 40th bytes in the sorted array using binary search and prints the returned indexes.

Write the following ASM programs that uses variations of the above functions.

1. A program that creates a thread with F1, waits until it is done, then creates another thread F2, waits until it is done, then creates another thread with F3, then ends the program.
2. A program that creates parallel threads with F1 and F3. Waits until all of them done, then terminates.
3. A program that creates 10 threads with F3 (10 parallel threads). Waits until all of them done, then terminates.
4. A program that creates two parallel threads with F4 and F5. Waits until all of them done, then terminates.
5. Modify F1 and F3 so that they call TYield at every loop iteration. Then write a program that creates parallel threads with F1 and F3. Waits until all of them done, then terminates.

You will provide the main simulation program as follows

- Write a simulation program that runs your systems with some command line parameters. There should be parameters for the program name and debug flag.
  - **`sim8080 exe.com 1`** : will read the program from exe.com. In debug mode 1, the status of the CPU will be printed to the screen after each instruction execution. At the end of the simulation, the whole memory will be saved to exe.mem as a text file of hexedecimal numbers. Each line of the file will start with the memory addres, then it will show 16 bytes of hexadecimal numbers separated with spaces.
  - In Debug mode 0, the program will be run and the contents of the memory will be saved as in Debug mode 1.
  - In Debug mode 2, every time a thread switch happens, the information is printed to the screen. The information printed will be
    - the thread IDs (both threads)
    - the total cycles spent for the blocked thread

- In Debug mode 3, information about each thread is printed on the screen when a thread switch occurs. This information will be very similar to "ps -ef" command in UNIX systems. It will include the thread ID and all of its thread table entries

We will provide the submission instructions in a separate document. You should strictly follow these instructions otherwise your submission may not get graded.
You will submit the following files for this homework

- main.cpp
- gtuos.cpp and gtuos.h
- 5 ASM programs with their functions as described above
- The sample ASM programs (print numbers, add numbers, etc.) that you write for HW1