Assignment 1

Instructions

1. This assignment consists of **nine** questions.
2. **Due Date: Monday, October 9th, 2023**. Submit your answers to Canvas.
3. Before you begin, please take the time to review the course policy on academic integrity at: https://www.cityu.edu.hk/ah/ . Please write your own answer. All submitted answer will be scanned by anti-plagiarism software.

Please use the answer sheet provided to write all your answers.

1. Describe how you could obtain a statistical profile of the amount of time spent by a program executing different sections of its code. Discuss the importance of obtaining such a statistical profile.

2. Describe the operating system's two modes of operation.

3. Describe the relationship between the API, the system-call interface, and the operating system. Why use APIs rather than system calls?

4. What are the differences between user-level threads and kernel-support threads? Name and describe the four different multithreading models.

5. Considering the following four algorithms: FCFS, non-preemptive SJF, preemptive SJF and RR. Please answer the following questions. (Each question may have more than one answer).

   a. Which algorithm(s) has/have the maximum CPU utilization (suppose the context switch overhead can be ignored)?

   b. Which algorithm(s) has/have the minimum average waiting time?

   c. Which algorithms(s) is/are the fairest?

6. Consider the following process A, B, C, D with arrival and processing times as given in the table:

Process Name	Arrival Time	Processing Time
A	0	7
B	2	4
C	4	4
D	9	3

Compute the finish time, response time, and turnaround time for each process for the following CPU Scheduling. Insert the values into the table in the answer sheet:

1) First-Come-First-Serve (FCFS)
2) Shortest-Job-First (SJF) (preemptive)
3) Round-Robin (RR) (time quantum = 1)

Note: For the answer of RR, we assume that the new coming job will be put into the end of the waiting queue.

		A	B	C	D
FCFS	Finish				
	Response				
	Turnaround				
SJF	Finish				
	Response				
	Turnaround				
RR	Finish				
	Response				
	Turnaround				

Draw the schedule for each scheduling scheme by filling in the name of the current running process. Each square represents one time unit.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
FCFS																			
SJF																			
RR																			

7. Consider the following two threads in the same process executing on the same core. The value at memory 2000 is initialized to 0. We also assume that Thread 1 is scheduled to execute first.

Thread_1

```
.main
mov $3,%ax
```

```
.top
mov 2000,%dx
add $1,%dx
mov %dx,2000
sub $1,%ax
test $0,%ax
jgt .top
halt
```

Thread_2

```
.main
mov $5,%ax
.top
mov 2000,%dx
add $2,%dx
mov %dx,2000
sub $1,%ax
test $0,%ax
jgt .top
halt
```

Note: the above-used assembly instructions are the same as the x86 assembly introduced in tutorial 3.

(a) What is the final value at the memory 2000 if the interrupt is disabled?

(b) If the interrupt occurs in each instruction, can we get the same value at the shared memory 2000? If yes, please explain the reason; if not, what part of instructions in the code block of Thread 1 and Thread 2 needs to be protected?

The following two questions are based on running the same `x86.py` simulator provided in tutorial 3.

Copy the `x86.py` simulator and program `wait-for-me.s`, which are in /public/cs3103/assignment1/ directory, to your working directory on gateway server.

8. Run: `./x86.py -p wait-for-me.s -a ax=1,ax=2 -R ax -M 2000`. This sets the `%ax` register to 1 for thread 1, and 2 for thread 2, and watches `%ax` and memory location 2000. How should the code behave? How is the value at location 2000 being used by the threads? What will its final value be?

9. Now switch the inputs: `./x86.py -p wait-for-me.s -a ax=2,ax=1 -R ax -M 2000`. How do the threads behave? What is thread 0 doing? How would changing the interrupt interval (e.g., `-i 1000`, or perhaps to use random intervals) change the trace outcome? Is the program efficiently using the CPU?