Question Sheet for **Introduction to Operating Systems**@CS_NYCU, Fall 2022
**Provide sufficiently clear explanation in your answers to get full credits

1. [8 pts] There are two kinds of interrupts, namely, interrupts from external devices (IRQ) and interrupts triggered by software execution (trap/exception). Which one(s) of the following are better described as a trap? Why?
   a) Illegal memory reference
   b) Timer expiration
   c) Divided by zero
   d) I/O device status change

2. [2 pts] I/O scheduling determines which one among a set of pending I/O requests should be serviced next. Now, consider the case for disk I/O scheduling. Among disk requests, the scheduler often gives a higher priority to read requests than write requests. Explain why.

3. [9 pts] Modern operating systems divide the execution mode into a user mode and a kernel mode. Answer the following questions:
   a) What is the benefit of this user-kernel mode separation?
   b) Does the user-kernel mode separation require hardware (CPU) support? Why?
   c) Embedded devices often do not support installation of new programs, and all software runs on the device have been well-tested before shipping. To embedded operating systems, is the separation of user mode and kernel mode necessary? Why?

4. [8 pts] In virtual machine implementation, let native execution refer to that the guest and the host share the same instruction set architecture (ISA). If the host and the guest use different ISA, then the virtual machine is non-native execution. Answer the following:
   a) Given a real-world example of native-execution virtual machines, and also give an example of non-native-execution ones.
   b) Discuss the pros and cons of these two approaches.

5. [9 pts] Show a scenario that triggers each of the following process-state changes:
   a) Running to ready
   b) Running to waiting
   c) Waiting to running

6. [6 pts] Answer the following questions regarding process management in UNIX:
   a) What is a zombie process?
   b) How do you avoid to have zombie processes in your first programming assignment?
   c) What is an orphaned process?

7. [6 pts] A context switch operation from process P1 to process P2 will involve the following steps. Now, give a correct sequence of these steps during the switch:
   a) Push the state register to P1's stack
   b) Restore the state register from P2's stack
   c) Push all general-purpose registers to P1's stack
   d) Restore all general-purpose registers from P2's stack
   e) Save the stack pointer register to P1's process control block
   f) Restore the stack pointer register from P2's process control block

8. [6 pts] Let a program be perfectly parallelizable. Consider that you have prepared a multi-threaded (parallelized) version and a single-threaded version of the program and then run these two on a

machine. However, unexpectedly, you observed that the multi-threaded version does not perform better than the single-threaded version. Discuss possible reasons. (Hint: discuss in terms of the threading model and the multiprocessing architecture.)

9. [4 pts] Chrome browser creates a process rather than a thread for each tab. This seems counter-intuitive, because threads are more economic than processes in terms of resource requirement. Explain the rationale behind the Chrome's design choice.

10. [4 pts] Is Shortest-Job-First a good choice for process scheduling with deadlines? Justify your answer.

11. [12 pts] Consider the multilevel feedback queue scheduling algorithm. Let the scheduler has three queues, Q1, Q2, and Q3. Let processes in Q1 be serviced first and those in Q3 be serviced the last. Processes in the same queue are scheduled by RR, and the time slice sizes for Q1, Q2, and Q3 are 3, 9, and 15 units of time, respectively. If a process releases the CPU before using up its time slice, then it is upgraded to the next queue of a smaller time slice. If a process runs out of its time slice, then it is preempted and downgraded to the next queue of a larger time slice. Let there be two processes: P1 repetitively runs on the CPU for 1ms before calling a blocking I/O operation; P2 repetitively calls a blocking I/O operation after executing on the CPU for 10ms.
    a) Which one of P1 and P2 is better described as an I/O-bound process?
    b) After a long period of time, in which queues P1 and P2 will be staying in respectively? Why?

12. [6 pts] Answer the following questions regarding process scheduling on multiple CPUs:
    a) [4 pts] Why process migration is necessary for load balancing? Should it be conducted very frequent?
    b) [2 pts] Linux kernel takes a hierarchical approach to load balancing by introducing different levels of domains. For example, all NUMA nodes of the system form a domain, and all cores of the same NUMA node form another domain. Load balancing is conducted in a layer-by-layer manner, beginning from the topmost domain. The question is: why not just treat all cores in the system equally using a single, flat model? Provide your explanation.

13. [8 pts] Interrupt disabling and atomic test-and-set are both hardware-oriented solutions to the critical-section problem. Now, consider that you are implementing critical sections in the kernel for an embedded operating system on a single-core CPU. Between the two solutions, which one is more appropriate? Justify your answer.

14. [12 pts] Consider the following code template:

```
thread()
{
    ...
    X
    Y
    ...
}
```

Write pseudo code for the following questions:

a) [8 pts] Consider two concurrent threads, both executing the function thread(). Add proper semaphore operations such that no threads execute Y until both of the two threads have executed X. Threads can use different semaphore operations. (Hint: the Jack-Rose problem.)
b) [4 pts] Now generalize the solution to arbitrary numbers of threads (say, 1000). Add proper semaphore operations and *necessary logic* such that no threads execute Y until all threads have executed X. Note: *The total number of semaphores you use must be* ***independent*** *of the total number of threads. Specifically, two semaphores would be enough.*