1. OS Design (5 points)

(a) (3 points) Describe three expected advantages of micro kernels compared to monolithic kernels.

- Modularity: clear user-kernel interface.
- Safety: Server’s malfunction can be as isolated as any other user programs.
- Flexibility: For example, different file systems and APIs may coexist.

(b) (2 points) Some micro kernels such as L4 and Exokernel are hardware (i.e., processor) dependent unlike the original hardware-independent micro kernels. Describe two reasons for this design decision.

Better performance can be achieved by taking advantage of specific hardware support and avoiding potential problems due to the ignorance of specific hardware peculiarities.

2. Multithreading and Synchronization (5 points)

(a) (2 points) Briefly describe one advantage and one disadvantage of kernel-level threads.

Advantages:
- The kernel can schedule another thread when if one thread performs a blocking system call, e.g., to do I/O.
- Multiprocessing is easier because the kernel can directly schedule threads.

Disadvantage:
- The kernel is unaware of the user-level application parallelism. Even if multiple processors are given, the kernel may not fully utilize them.
- The kernel should be modified to support different a thread package.

(b) (2 points) Briefly describe how to combine the advantages of user- and kernel-level threads. (Recall the “scheduler activation” paper.)

A user-level thread manager schedules user-level threads using virtual processors mapped to physical processors managed by the kernel. In this way, user applications can request or yield one or more virtual processors, if necessary, to meet the user level parallelism requirements. The kernel informs the user-level thread manager of important kernel events such as a blockage of a thread or availability of physical processors using upcalls and scheduler activations.
(c) (1 point) To solve a critical section problem, mutual exclusion, progress, and bounded waiting conditions should be met. Explain why the following tentative solution violates the progress requirement.

/* There are two processes \( P_0 \) and \( P_1 \) and \( j = 1-i \) where \( i = 0 \) or \( 1 \). */

```java
boolean flag[2];

while(1)
{
    flag[i] = true;
    while (flag[j]);

    Critical Section

    flag[i] = false;

    Remainder Section
}
```

The above algorithm violates the progress requirement, because \( P_0 \) and \( P_1 \) can wait for each other forever if they set \( \text{flag}[i] = \text{true} \) simultaneously. (For more details, refer to Section 7.2.1.2 of the Silberschatz book.)

3. CPU Scheduling (5 points)

(a) (1 point) Describe how to determine the time slice size in the round robin scheduling algorithm.

The time slice should be larger than the context switch time. However, it should not be too large. Otherwise, it may degrade to FCFS (First Come First Serve) when the time slice is larger than most jobs.

(b) (1 point) Describe how the multilevel feedback queue scheduling handles the potential starvation problem in the multilevel queue scheduling.

If a process consumes too much CPU time, it is moved to a lower priority queue. On the other hand, a process is moved to a higher priority queue if it has waited too long.

(c) (1 point) Give a simple example in which fast processing can actually result in a deadline miss.

Assume task A is optimized and takes less execution time than it used to. As a result, it enters a critical section earlier blocking task B that has an urgent deadline. As a result, task B misses its deadline.

(d) (2 points) EDF (Earliest Deadline First) is an optimal dynamic scheduling algorithm and its utilization bound is 100\%. Describe two problems applying EDF to practical systems.
• It may suffer the domino effect in the presence of a transient overload.
• It requires an infinite number of priority levels.

4. Memory Management (5 points)

(a) (1 point) Compute the average memory access time when a TLB (translation lookaside buffer) is used as follows:
(i) TLB hit ratio = 95%, (ii) TLB access time = 20ns, (iii) memory access time = 100ns (direct memory access without TLB access).
\[0.95 \times (20 + 100)ns + 0.05 \times (20 + 100 + 100)ns = 125ns\]

(b) (1 point) Briefly describe one approach to alleviate thrashing due to memory contention in a multiprogramming environment. (There could be several solutions, but you only have to give one solution.)
If the page fault rate is too high, the OS should suspend (a) process(s) and distribute its page frames to other processes. (Alternatively, the working set model can be applied. For more details, refer to Sections 10.6.2 and 10.6.3 of the Silberschatz book.)

(c) (1 points) Briefly describe why forward-mapped page tables can suffer poor performance for 64 bit address spaces.
It will require seven levels of pages tables. Thus, one page fault requires seven memory accesses just to convert a virtual address to physical address. (As long as an answer discusses the several levels of page table lookups and the corresponding performance penalty, the answer is considered correct.)

(d) (2 points) Briefly describe two reasons for supporting superpages in modern operating systems.
• The TLB size has not increased as much as the memory size has increased over the last decade.
• Some applications, e.g., database and multimedia applications, may prefer big pages.

5. True/False Questions (5 points)

(a) A micro kernel provides the minimal, indispensable operating system primitives including basic scheduling, address spaces, device drivers, and interprocess communication. [T/F]
False. (Device drivers do not have to be implemented inside a micro kernel.)

(b) In a monolithic kernel, most operating system components, e.g., memory management, IPC (interprocess communication), and basic synchronization modules, execute outside the kernel. [T/F]
False.
(c) Each thread of one process has to maintain a separate program counter, stack, and registers. [T/F]
   True.

(d) Segmentation has no external fragmentation problem. [T/F]
   False.

(e) Cycle stealing of the DMA (Direct Memory Access) may slow down the CPU. [T/F]
   True.