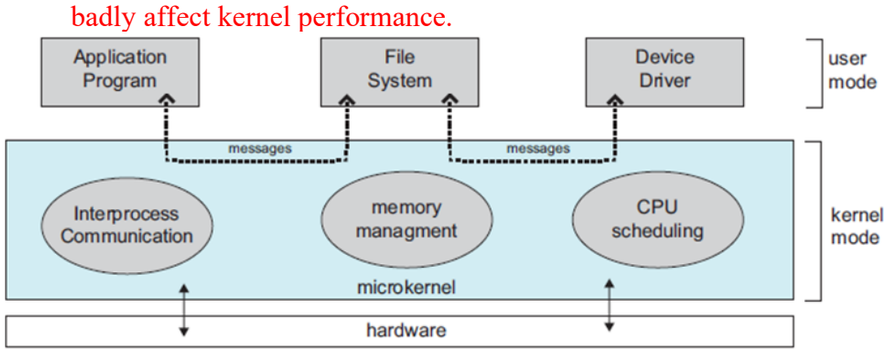
Assumption University
Vincent Mary School of Engineering, Science and Technology
CSX3008 Quiz1 KEY 2/2024 Total 50 points

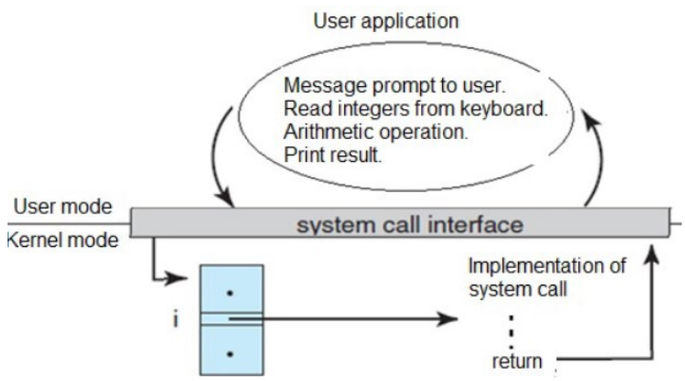
Instructions: Answer all questions in the spaces provided on the examination paper (*please do not use pencils for your answer*). Students are allowed to use an electronic calculator. This is a closed-book examination.

1. (3 points) Check whether the following operations use a *system call*. Briefly describe your reason(s).
   1.1) (1 point) Program reads a file from USB memory. A system call is needed to get I/O service from the kernel.
   1.2) (1 point) Executing a loop statement (either a *for loop* or a *while loop*). A CPU operation does not need a system call.
   1.3) (1 point) The program waits for keyboard values. A system call is needed to get I/O service from the kernel.
2. (5 points) Answer the following based on the OS architecture shown in the **Figure** below:
   2.1) (1 point) Identify the OS architecture. Microkernel
   2.2) (2 points) Briefly explain how the core OS functions, such as the file system and device driver, support user applications. Through the message passing facility, the user application gets connected to the file system and device driver, and these functions will be executed in user mode.
   2.3) (2 points) Describe this architecture's advantages and disadvantages. There is no system call burden is an advantage, and the disadvantage is too many messages passing may badly affect kernel performance.

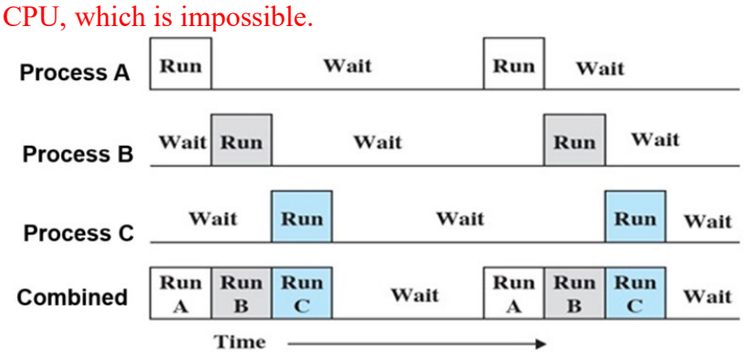


3. (5 points) Answer the following based on the **Figure** below:
   3.1) (1 point) Identify the operations on the user application that cause a system call. Message prompt, keyboard read, and print result
   3.2) (1 point) Why do those operations need the system call? Because those operations are I/O in nature.
   3.3) (2 points) Briefly describe how the CPU behaves during the system call. During a system call, the CPU stops its current user application execution and executes the system call instructions (kernel mode), which are loaded into the main memory by the kernel (call interrupt service routine) for a particular system call, and once it is over, the CPU resumes it interrupted job (back to user mode)

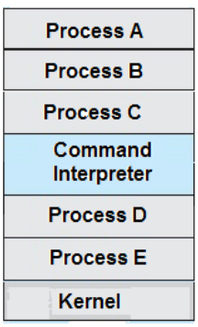
3.4) (1 point) What is the significance of user and kernel modes? These modes distinguish a CPU's execution into the user application execution phase (user mode) and the system call instruction execution phase (kernel mode). This separation of execution modes will protect the kernel from the user-level instructions.



4. (1 point) Check whether the following statement is **true** or **false**: An interrupt service routine (ISR) is an OS-predefined program that deals with interrupts. True
5. (1 point) Check whether the following statement is **true** or **false**: The message-passing Inter-process Communication (IPC) model is useful for exchanging a large amount of data and requires system calls to implement. False
6. (2 points) To execute a program, its disk location address (assume, 0xAF065B81) is mapped to the main memory address. Why is this mapping relevant? This address mapping scheme is a part of *process* creation. The CPU cannot access the executable file of a user application from a secondary drive (disk drive), so it must be transferred from its disk location to the main memory address location. This address mapping scheme is called the address binding operation and is done by OS.
7. (2 points) Show how a **Mutex lock** (mutual exclusion lock) can support the **critical section** (CS) in a process synchronization problem. Describe one of the main disadvantages of the **Mutex lock.** The critical section (CS) is an eminent part of processes under synchronization. A process waiting for its CS must gain access to the mutex lock (**acquire-lock()**); otherwise, it cannot enter its CS. Once its CS is over, it must release the mutex lock (**release-lock()**) for the following waiting process. The main disadvantage of mutex-lock in CS problems is that the CPU must check the availability of the mutex lock in each clock cycle (called spin-lock), eventually leading to a busy-waiting CPU situation.
8. (2 points) Assume an **eight-core** system runs an application program. It has been noticed that 75% of threads run in parallel. Calculate the application's speedup gain.
   Speedup = 1/((S + (1-S)/N); S = 0.25, and N = 8, therefore Speedup =2.91 times
9. (6 points) A three-process execution paradigm is shown in the **Figure** below. Answer the following based on the figure:
   - 9.1) (2 points) Is this multiprocessing or multiprogramming? Why? It is multiprogramming because a single CPU interleaves the processes execution
   - 9.2) (2 points) Can the processes execute without utilizing the 'wait' state? Briefly describe your reasons. Yes, it is possible to perform interleaved process execution by using a time quantum base Round-Robin algorithm.
   - 9.3) (2 points) Is there any preemption involved in this process's execution? Describe. Yes, preemption is involved; without preemption, there is no interleaved process execution on a single CPU, which is impossible.



10. (2 points) Consider the memory of a multitasking OS shown in the **Figure** below. Assume that all the processes are available in memory simultaneously and no process priority details are available. The context-switching facility of the processes is implemented in the OS. Briefly describe a suitable algorithm for scheduling these processes in a multiprogramming system. Applying either Round-Robin or Shortest-Job-first preemptive algorithm for the process scheduling is possible here.

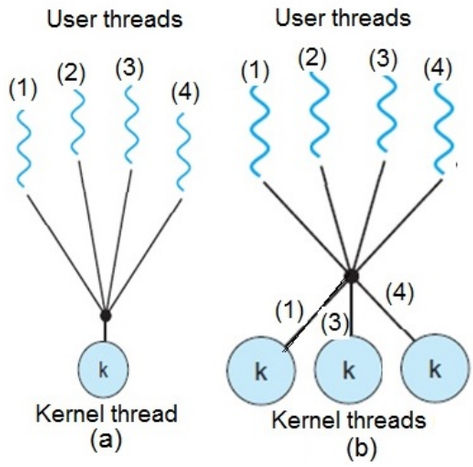


11. (2 points) Identify the inter-process communication (IPC) process and its IPC model.

```
item next_consumed;
while (true) {
        if (counter == 0)
          { /* wait until counter!= 0 */}
        next_consumed = buffer[out];
        out = (out + 1) % BUFFER_SIZE;
         counter--; }
```

The process is a consumer process, and the IPC model is a shared memory model (common buffer)

12. (4 points) Answer the following based on the two user-level to kernel-level threads mapping scenarios shown in the **Figure** below:
    12.1) (1 point) Why is thread mapping significant? The user-level threads cannot be identified and executed without the user-level to kernel-level thread mapping
    12.2) (1 point) Identify the thread mapping scheme suitable for multiprogramming. Figure(a)
    12.3) (1 point) Describe the impact on the process when the **user thread (1)** in (**a**) causes a **system call**. The entire threads will be blocked
    12.4) (1 point) Describe the impact on the process when the **user thread (3)** in (**b**) causes a system call. Only the thread 3 is blocked



13. (4 points) The execution sequence of producer and consumer processes in a multiprogramming system is shown in the **Figure** below (where the ***counter*** is a shared variable for both the producer and consumer processes). Based on the figure, answer the following:
    13.1) (2.5 points) Check whether **race condition**s occur in the interleaved execution of processes. Describe its reason. There is a race condition at time T5. The consumer process failed to update its counter due to the producer takeover.
    13.2) (1.5 points) Is this a process synchronization problem? Why? Yes, it is a process synchronization problem because the consumer and producer share a common variable, '*counter*', along with their shared *buffer*.

Time	Process	Register-counter Status	Value
T_0	producer	$register_1$ = counter	$register_1 = 5$
T_1	producer	$register_1$ += 1	$register_1 = 6$
T_2	producer	counter = $register_1$	counter = 6
T_3	consumer	$register_2$ = counter	$register_2 = 6$
T_4	consumer	$register_2$ –= 1	$register_2 = 5$
T_5	producer	$register_1$ = counter	$register_1$ = 6
T_6	consumer	counter = $register_2$	counter = 5

14. (3 points) What is a critical section (CS)? Describe briefly how the CS of each synchronized process is accomplished with a mutual exclusion lock (Mutex Lock).
The critical section (CS) is an essential part of processes under synchronization. A process waiting for its CS must gain access to the mutex lock (acquire-lock()); otherwise, it cannot enter its CS. Once its CS is over, it must release the mutex lock (release-lock()) for the following waiting process. The mutex lock system ensures that only one process at a time enters its CS to maintain the integrity of the shared resource.
15. (8 points) Five processes with their *arrival time*, *burst time*, and *priority* are shown in the **Table** below**.** Based on the table, estimate the *average waiting time* of the processes by the following scheduling algorithms by using their Gantt Chart (Note: you should show all the individual waiting times of the processes before calculating their average waiting time):

Process	Arrival Time (ms)	CPU Burst (ms)	Priority
P1	0	7	3
P2	1	9	1
P3	2	7	2
P4	3	3	5
P5	4	10	4

15.1) (2 points) First-Come First-Served (FCFS).
Wt. P1 = 0, P2 = 6, P3 =14, P4 = 20, P5 = 22, Avg.Wt = 12.4 ms

15.2) (2 points) Preemptive Shortest-Job-First (SJF).
Wt. P1 = 3, P2 = 16, P3 =8, P4 = 0, P5 = 22, Avg.Wt = 9.8 ms

15.3) (2 points) Preemptive Priority.
Wt. P1 = 16, P2 = 0, P3 =8, P4 = 30, P5 = 19, Avg.Wt = 14.6 ms

15.4) (2 points) Round-Robin (*time quantum* = 7).
Wt. P1 = 0, P2 = 23, P3 =12, P4 = 18, P5 = 22, Avg.Wt = 15 ms

End of Examination Questions