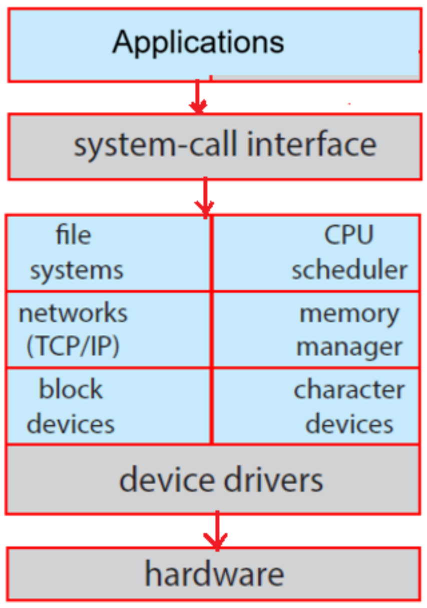
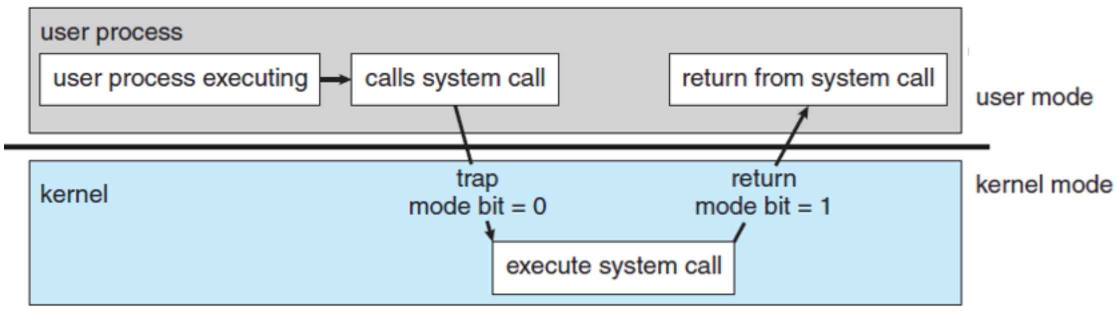
Assumption University
Vincent Mary School of Engineering, Science and Technology
CSX3008 Quiz1 2/2025 Total **50** points **SOLUTION**

Instructions: Read the questions carefully and answer all questions in the spaces provided on the examination paper (*please do not use pencils for your answer*). Students may use a non-programmable scientific calculator. This is a closed-book examination.

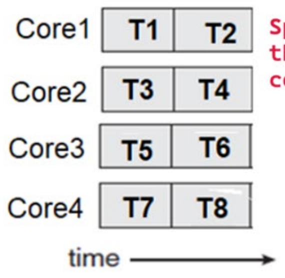
1. (3 points) Check whether the following operations use a *system call*. Briefly describe your reason(s).
   1.1) (1 point) Program reads a file from USB memory. **Yes, the I/O operation requires a system call.**
   1.2) (1 point) Executing a loop statement (either a *for loop* or a *while loop*). **A CPU-bound operation does not need a system call.**
   1.3) (1 point) The program waits for keyboard values. **Yes, the I/O operation requires a system call.**
2. (5 points) Answer the following based on the OS architecture shown in the **Figure** below:
   2.1) (1 point) Identify the OS architecture. **Monolithic architecture**
   2.2) (2 points) Briefly describe the importance of the system-call interface. **The system call interface provides kernel support for the system calls generated by user I/O functions.**
   2.3) (2 points) Describe this architecture's *advantages* and *disadvantages*. **Easy to implement, especially since there is no system call burden for the kernel. But it is not easy to add additional kernel functions or modify the current kernel.**



3. (6 points) Answer the following based on the user mode to kernel mode transition shown below:
   3.1) (1 point) What is a system call? **A software interrupt, primarily generated by user source code with an I/O command, is used to request services from the kernel.**
   3.2) (1 point) Why does the user process cause a system call? **Because those user instructions are I/O-oriented.**
   3.3) (2 points) Describe how the OS supports the CPU to execute the system call. **During a system call, the CPU stops its current user application execution and executes the system call instructions (in kernel mode), which are loaded into the main memory by the kernel (this execution routine is called an interrupt service routine or ISR), and once it is over, the CPU resumes its interrupted job (back to user mode)**
   3.4) (2 points) Describe the significance of *user and kernel modes* in terms of the system call. **These two modes distinguish a CPU's execution mode: the execution of CPU-bound instructions in a user application (user mode) and the execution of predefined system-call instructions of the kernel (kernel mode). This separation of execution modes will protect the kernel from the user-level instructions.**

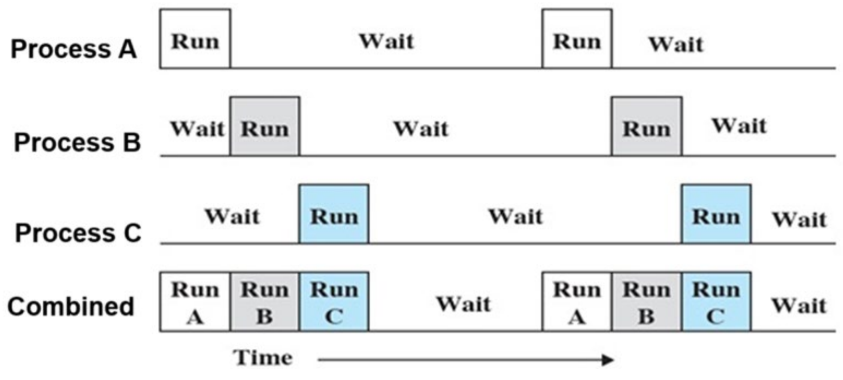


4. (1 point) An **interrupt** from either the hardware or the software usually signals the occurrence of an event in a computer system.
5. (1 point) The process of starting a computer by loading the OS kernel is known as **booting** the system.
6. (1 point) Check whether the following statement is **true** or **false**: Most of the commands in the UNIX command line interpreter are implemented through system programs and not based on system calls. **True**
7. (1 point) The bootstrap program, which is loaded in an electrically erasable programmable read-only memory (EEPROM), is known as **firmware**.
8. (2 points) A **program** is a ***passive*** entity, and a **process** is an ***active*** entity. Briefly describe.
A program, especially when stored on disk as an executable file, is a passive entity because a CPU cannot execute its instructions directly from the disk. It will be executed only when it is loaded into main memory at runtime. Once the executable file is in the main memory, it becomes a process and is managed by the CPU scheduler for execution. Therefore, the process is an active entity.
9. (3 points) Calculate the speedup gain of an application running on a four-core machine as shown in the **Figure** below, where T1, T2, T3, T4, T5, T6, T7, and T8 are the threads of an application- hints: There are *four* serial executions and *four* parallel executions.

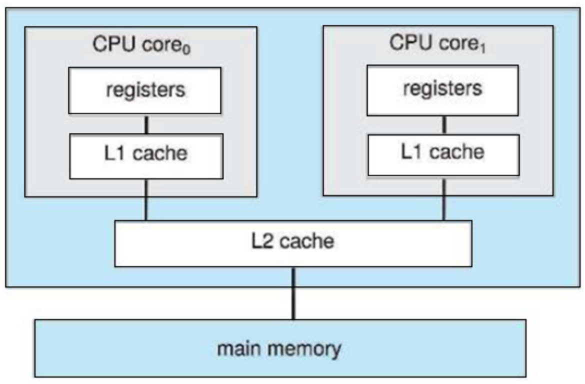


Speedup = 1/(S + (1-S)/N), where S = 0.5 (50% of the threads are running in serial), and N= 4 (no. of cores). Speedup = 1/(0.5 + (1-0.5)/4 = 1.6 times.

10. (5 points) A three-process execution paradigm is shown in the **Figure** below. Answer the following based on the figure:
    10.1) (2 points) Is this multiprocessing or multiprogramming? Why? **It is multiprogramming because a single CPU interleaves the processes' execution.**
    10.2) (1.5 points) Can the processes execute without utilizing the 'wait' state? Briefly describe your reasons. **Yes, it is possible to perform interleaved process execution using a time quantum-based Round-Robin algorithm.**
    10.3) (1.5 points) Is there any preemption involved in this process's execution? Describe. **Yes, preemption is involved; without preemption, it is not possible to execute an interleaved process execution on a single CPU.**



11. (2 points) Identify the multiprocessing architecture shown in the following figure (symmetric or asymmetric). Describe why: **It is a Symmetric multiprocessing because both CPU cores are sharing a common main memory and its related bus systems.**



12. (5 points) What is a **semaphore**? **A semaphore is one of the most advanced process synchronization tools used by an OS. It protects access to shared data via wait(P) and signal(V) functions.** Show how the usage of semaphores in solving the **producer-consumer** synchronization problem (assume that the *size* of the buffer is ***n***, ***mutex*** is a semaphore initialized to **1**, ***full*** is a semaphore initialized to **0**, and ***empty*** is a semaphore initialized to ***n***).

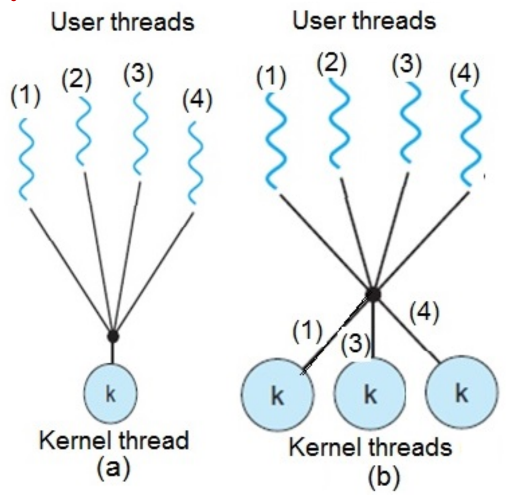
The structure of the **producer process**

```
do   { //   produce an item
     wait (empty); // consumer should wait  because the buffer is empty
     wait (mutex); // consumer wait, mutex  semaphore  is with producer
     add the item to the  buffer; // fill buffer (CS)
     signal (mutex); // signal to consumer that 'mutex' is free and get it
     signal (full); // signal to consumer that buffer is full
     }while (TRUE);
```

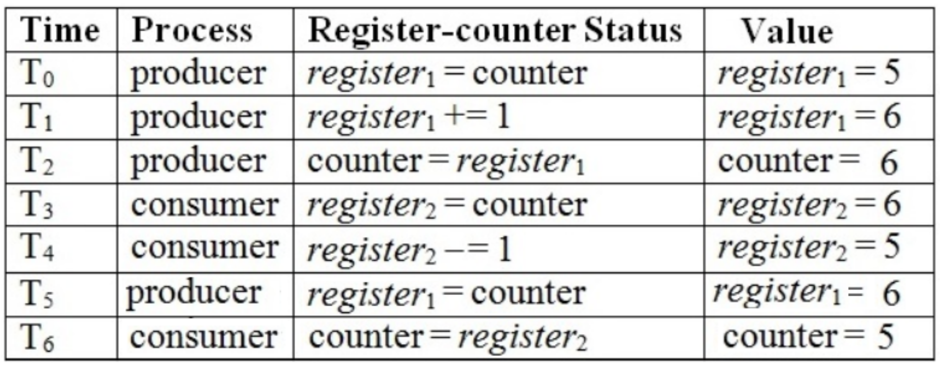
The structure of the **consumer process**

```
do { wait (full); // producer wait because buffer is full
     wait (mutex); // producer wait 'mutex'  is with consumer
     remove an item from  buffer; // consume buffer (CS)
     signal (mutex); // signal to producer that mutex is free and get it
     signal (empty); // signal to producer that buffer is  empty
     } while (TRUE);
```

13. (4 points) Answer the following based on the two user-level to kernel-level threads mapping scenarios shown in the **Figure** below:
  13.1) (1 point) Identify the thread mapping scheme that is suitable for **multiprogramming**. **(a)**
  13.2) (1 point) Identify the thread mapping scheme that is suitable for **multiprocessing**. **(b)**
  13.3) (1 point) Describe the impact on the process when the **user thread (1)** in (**a**) causes a **system call**. **The entire thread (process) will be blocked.**
  13.4) (1 point) Describe the impact on the process when the **user thread (3)** in (**b**) causes a system call. **Only thread 3 is blocked.**



14. (3 points) The execution sequence of the producer and consumer processes in a multiprogramming system is shown in the Figure below (where the counter is a shared variable for both processes). Based on the figure, determine whether a **race condition** occurs during the interleaved execution of processes. Describe its reason. **There is a race condition at time T5. The consumer process failed to update its counter to 5 because the producer took over**.



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

15. (8 points) Five processes in a concurrent system, along with their *arrival times*, *burst times*, and *priorities*, 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
P_1	0	10	5
P_2	1	12	4
P_3	2	7	3
P_4	3	5	1
P_5	4	3	2

15.1) (2 points) First-Come First-Served (FCFS). **(Waiting time: P_1 = 0, P_2 = 9, P_3 = 20, P_4 = 26, P_5 = 30; the Average Waiting Time is 17 ms)**

15.2) (2 points) Preemptive Shortest-Job-First (SJF). **(Waiting time: P_1 = 15, P_2 = 24, P_3 = 8, P_4 = 3, P_5 = 0; the Average Waiting Time is 10 ms)**

15.3) (2 points) Preemptive Priority. **(Waiting time: P_1 = 27, P_2 = 15, P_3 = 8, P_4 = 0, P_5 = 4; the Average Waiting Time is 10.8 ms)**

15.4) (2 points) Round-Robin (*time quantum* = 5). **(Waiting time: P_1 = 18, P_2 = 24, P_3 = 26, P_4 = 12, P_5 = 16; the Average Waiting Time is 19.2 ms)**

End of Examination Questions