National University of Computer and Emerging Sciences

FAST School of Computing **Fall-2023** **Islamabad Campus**

CS-2006: Operating Systems SOL

Tuesday, 26th September, 2023

Course Instructors

Khwaja Bilal Hassan

Serial No:

Sessional Exam-I

Total Time: 1 Hour

Total Marks: 60

_____

Signature of Invigilator

Student Name	Roll No.	Course Section	Student Signature
_____	_____	_____	_____

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Attempt on question paper. Attempt all of them. Read the question carefully, understand the question, and then attempt it.
2. No additional sheet will be provided for rough work. Use the back of the last page for rough work.
3. If you need more space, write on the back side of the paper and clearly mark question and part number etc.
4. After asked to commence the exam, please verify that you have **Nine** different printed pages including this title page. There are total of **Seven** questions.
5. Calculator sharing is strictly prohibited.
6. Use permanent ink pens only. Any part done using soft pencil will not be marked and cannot be claimed for rechecking.

	Q-1	Q-2	Q-3	Q-4	Q-5	Q-6	Q-7	Total
Marks Obtained								
Total Marks	**10**	**12**	**08**	**08**	**06**	**10**	**06**	**60**

Question 1 [10 Marks]

Question: Select the appropriate answer in the following by encircling them.

1. Which of the following best describes the typical architecture of a CPU?
   a) Thousands of simple cores optimized for parallel processing
   b) A few complex cores optimized for single-threaded tasks
   c) Specialized cores designed for graphics rendering
   d) None of the above
2. The instruction fetched from the main memory is kept in _____?
   a) PC
   b) AC
   c) IR
   d) None of the above
3. Which of the following is not a characteristic of a real-time operating system?
   a) Predictable response times
   b) Prioritization of tasks
   c) Support for complex graphics rendering
   d) Meeting strict deadlines
4. What is the main advantage of a microkernel-based operating system structure?
   a) High performance
   b) Simplicity and modularity
   c) Wide support for legacy applications
   d) Rich graphical user interfaces
5. Which structure allows multiple operating systems to run on a single computer simultaneously?
   a) Monolithic structure
   b) Virtualization-based structure
   c) Modular structure
   d) Exokernel structure
6. A variety of technologies are used to implement memory systems, and across this spectrum of technologies, the following relationships does not hold:
   a) Faster access time, greater cost per bit.
   b) Greater capacity, smaller cost per bit.
   c) Greater capacity, slower access" speed
   d) Slower access time, smaller capacity
7. Which of the following does not affect the cache design:
   a) Number of Cache levels
   b) Cache Size
   c) Memory Speed
   d) Replacement Algorithm
8. Boot Program performs which of the following tasks
   a) Connect I/O devices
   b) Call restore function
   c) Examine/check machine configuration
   d) Clear memory
9. _____ process is assigned as the parent process to an orphan process
   a) Zombie
   b) Init

c) Parent
d) Main

10. No preemptive processor scheduling algorithms
    a) Rely on interrupts to multiplex the CPU
    b) Rely on cooperative sharing mechanisms to multiplex the CPU
    c) Are the preferred class of algorithms to support real-time computing
    d) Are most widely used because of their simplicity
    e) None of the above

Question 2 [2+2+2+2+2+2=12 Marks]

Briefly Explain the following your answer should not exceed three lines.

a) Multi-Programming VS Multiprocessing

Multi-Programming involves running multiple tasks on a single CPU by switching between them, while Multiprocessing involves running multiple tasks on multiple CPUs simultaneously for true parallel processing.

b) PCB VS Process Address Space

A PCB (Process Control Block) contains information about a specific process, including its state and resource utilization, while a Process Address Space refers to the memory allocated to a process, containing its code, data, and stack.

c) JVM VS Virtual Machine

JVM (Java Virtual Machine) is a specific virtual machine designed for running Java programs, while a "virtual machine" is a more general term that refers to software or hardware emulation of a computer system, allowing the execution of code in a controlled environment.

d) Tags Vs Address

Tags in the context of computer memory typically refer to metadata used in memory management, like cache tags, which help identify the location of data in cache. An address, on the other hand, is a numerical value used to specify the location of data in memory or storage.

e) What is starvation in CPU scheduling? How starvation affect the response time in SRTF scheduling algorithm.

Starvation in CPU scheduling is when lower-priority tasks are consistently delayed in favor of higher-priority ones. In SRTF scheduling, this can lead to longer response times for lower-priority tasks as they continuously wait for the CPU to become available, causing potential performance imbalances.

f) Hard Real-Time Systems VS Soft Real-Time System

Hard real-time systems have strict and non-negotiable timing requirements, where missing a deadline can result in a catastrophic failure. Soft real-time systems have timing requirements that are important but not as critical, allowing some flexibility in meeting deadlines without catastrophic consequences.

Question 3 [4+4 = 8 Marks]

a) Explain general methods used to pass parameters to the OS system call. Which method have no limit on number and length of parameters

There are typically three general methods for passing parameters to an OS system call:

Register Method: In this approach, some system call parameters are passed via CPU registers. Registers are typically fast and limited in number, so this method may not support a large number of parameters.

Stack Method: In this approach, parameters are pushed onto the stack before the system call is invoked. The system call then accesses these parameters from the stack. This method allows for a more flexible and extensive parameter passing mechanism since the stack can hold a variable number of parameters.

Block Method: Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register.

b) We have a three level memory with hit ration between level 1 and level 2 is 95 while the hit ratio between level 2 and level 3 is 80. The access speed from level 1 is 0.5 while that for level 2 is 2.5 and from level 3 it is 5 micro seconds. What will be the average access time?

To calculate the average access time in a three-level memory hierarchy, you can use the following formula:

Average Access Time = Access Time Level 1 + (Hit Ratio Level 1 to Level 2 * (Access Time Level 2 + (Hit Ratio Level 2 to Level 3 * Access Time Level 3)))

Using the given values:

Access Time Level 1 = 0.5 microseconds
Hit Ratio Level 1 to Level 2 = 95% or 0.95
Access Time Level 2 = 2.5 microseconds
Hit Ratio Level 2 to Level 3 = 80% or 0.80
Access Time Level 3 = 5 microseconds
Plug these values into the nested formula:

Average Access Time = 0.5 + (0.95 * (2.5 + (0.80 * 5)))

Average Access Time = 0.5 + (0.95 * (2.5 + 4))

Average Access Time = 0.5 + (0.95 * 6.5)

Average Access Time = 0.5 + 6.175

Average Access Time = 6.675 microseconds

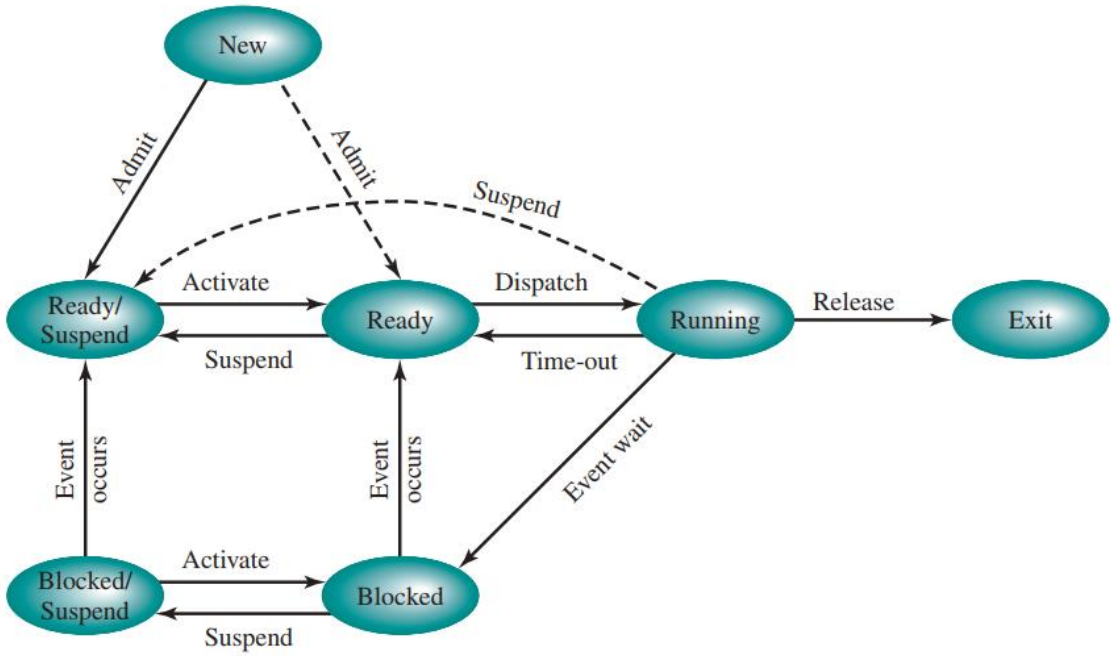
Question 4 [5+1+1+1=8 Marks]

Suppose there is a system with max memory of 4 words. The initial state of main memory and swap area is as follow. Here P1 size =2 words, P2 size = 2 words, P3 size = 1 word.

Main Memory
P1
P2

Swap Area
P3

a) Draw a process state diagram for the above.



(b) With two Suspend states

Figure 3.9 Process State Transition Diagram with Suspend States

b) If there is no process currently in the Running state, which process will be scheduled according to FCFS?

P1

c) What will be the status of the main memory and swap area after the previous step

P2 is already in main memory and **P3** will be shifted from ready suspended/swap area to ready state.

d) Can a process P2 be moved to blocked state? Explain the steps if possible.

In order to move P2 from ready to blocked state it need to first go to **running state** first and then it can go for **blocked state**.

Question 5 [6 Marks]

Calculate TAT and AWT for the following Ready Queue scenario by using following algorithms.
Note: The processor time start from 0 and there can be time where CPU may be Idle.

a) SJF (Preemptive)

Process	CPU Burst	Priority	Arrival Time
P1	4	5	1
P2	1	3	2

P3	3	0	3
P4	2	4	3
P5	4	0	12

Idle	P1	P2	P4	P3	P1	Idle	P5

0 1 2 3 5 8 11 12 16

Process no.	TAT	WT
P1	10	6
P2	1	0
P3	5	2
P4	2	0
P5	4	0

WAT = 8/5 = 1.6

Question 6 [10 Marks]

Question: Create a C program for the following scenario.

1. You have a dataset/array of size 10 that needs to be processed.
2. You want to divide the dataset into smaller chunks, and each chunk need to processing by a separate child process.
3. The child processes will run and compute the square of each number in your array and print it out each child will do the same on their respective data chunks simultaneously. While parent wait for all the child to end their execution

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

#define ARRAY_SIZE 10

int data[ARRAY_SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

void calculateAndPrintSquare(int index) {
   int result = data[index] * data[index];
   printf("Child %d: %d^2 = %d\n", getpid(), data[index], result);
}

int main() {
   pid_t child_pid;
```

```
    for (int i = 0; i < ARRAY_SIZE; i++) {
        child_pid = fork();

        if (child_pid == 0) {
            // Child process
            calculateAndPrintSquare(i);
            exit(0);
        } else if (child_pid < 0) {
            perror("Fork failed");
            exit(1);
        }
    }

    // Parent waits for all child processes to complete
    for (int i = 0; i < ARRAY_SIZE; i++) {
        wait(NULL);
    }

    printf("All child processes have finished.\n");

    return 0;
}
```

Question 7 [6 Marks]

Create a process tree and write the output of the following code:

```
int main() {
    int n = 3; // Number of child processes to create
    for (int i = 0; i < n; i++) {
        pid_t child_pid = fork();
        if (child_pid == 0) {
            printf("Child %d: Hello from child %d!\n", i + 1, i + 1);
            break;
        } else if (child_pid < 0) {
            perror("Fork failed");
            return 1;
        }
    }
    printf("Parent: Hello from the parent!\n");
    return 0;
}
```

```
Child 1: Hello from child 1!
Child 2: Hello from child 2!
Child 3: Hello from child 3!
Parent: Hello from the parent!
```

```
Parent: Hello from the parent!
Parent: Hello from the parent!
Parent: Hello from the parent!
```