A picture containing text

Description automatically generatedText

Description automatically generated with medium confidence**KING’S OWN INSTITUTE\***

**Success in Higher Education**

# INDIVIDUAL ASSIGNMENT COVERSHEET

Family Name: XXXX Given Name: XXXXX

Student Number: 121XXXXX Lecturer’s Name: Ali Noori

Subject Name: ICT201

Assignment Title: Assessment 4 (Individual)

**Declaration**

(This declaration must be completed by the student, or the assignment will not be marked.)

I certify the following:

* I have read and understood the *Student Academic Misconduct Policy*
* This assignment is my own work based on my personal study and or research.
* I have acknowledged all material and sources used in the preparation of this assignment including any material generated in the course of my employment.
* The assignment has not previously been submitted for assessment.
* I have not copied in part or in whole or otherwise plagiarised the work of other students.
* I have read and I understand the criteria used for assessment.
* The assignment is within the word and page limits specified in the unit outline.
* The use of any material in this assignment does not infringe the intellectual property / copyright of a third party.
* I understand that this assignment may undergo electronic detection for plagiarism, and an anonymous copy of the assignment may be retained on the database and used to make comparisons with other assignments in future.
* By completing this coversheet in full and submitting this assignment electronically, I am bound by the conditions of the KOI’s *Student Academic Misconduct Policy* and the declaration on this coversheet.

09 / 05 / 2023

Signature Date

Table of Contents

[Executive Summary 2](#_Toc134481792)

[Introduction 2](#_Toc134481793)

[OS Design Considerations depends on: 3](#_Toc134481794)

[Thread 3](#_Toc134481795)

[Synchronization 4](#_Toc134481796)

[Scheduling 4](#_Toc134481797)

[Memory management 5](#_Toc134481798)

[Reliability and fault tolerance 5](#_Toc134481799)

[Other OS design Considerations 6](#_Toc134481800)

[Power Management 6](#_Toc134481801)

[Security and Reliability 6](#_Toc134481802)

[Performance Monitoring and Optimization 6](#_Toc134481803)

[Challenges and Future Treads 7](#_Toc134481804)

[Conclusion 7](#_Toc134481805)

[Reference Lists 8](#_Toc134481806)

# Executive Summary

This report talks about how operating systems (OS) should be designed to work well with multiprocessor and multicore systems. These types of systems are becoming more common, so it's important for OS to be able to use them efficiently. The report focuses on five important things to think about when designing OS for these systems: thread, synchronization, memory management, scheduling, and reliability and fault tolerance.

The report explains some of the challenges and solutions for designing OS that achieve these design considerations, like how to make sure OS can perform thread and how to use different synchronization mechanisms. The report also talks about things like memory management, scheduling, and reliability and fault tolerance. Overall, the report recommends that designers must carefully consider these design considerations when creating OS to make the most of multiprocessor and multicore systems.

# Introduction

A multiprocessor is a type of computer system that has more than one CPU or processor working together to allow the system to execute multiple instructions at the same time, which makes it work faster (Javapoint, n.d.). Additionally, if one CPU fails, the other processors can keep working without any issues, which makes multiprocessors more dependable. In a distributed memory multiprocessor, each CPU has its own private memory and can utilize local data to perform computational tasks, while also being able to communicate with other processors or access the main memory through the bus if remote data is necessary (Javapoint, n.d.).

A multicore processor is a computing component with multiple independent processing units, known as cores, in a single CPU system, each core can read and run computer instructions, making the system appear to have several processors, even though they are cores (Javapoint, n.d.). These cores can perform typical processor functions such as adding, moving data, and branching. The performance of a multicore system is dependent on the software techniques used to implement its cores, which is why there is a heightened emphasis on developing software capable of parallel execution across multiple cores (Javapoint, n.d.).

It is essential to take a number of design considerations into account to make sure an operating system operates at its best. These elements can significantly affect a system's overall performance and aid in avoiding performance-related bottlenecks. Hardware specs, concurrent user count, and application kinds must all be taken into account for the system to operate at its best performance. It is feasible to optimise the performance of the operating system and prevent potential problems that may otherwise impair it by taking these elements into consideration during the design phase. Some critical design considerations that can impact operating system performance include virtual memory usage, process priority, resource allocation, and system architecture.

# OS Design Considerations depends on:

Operating system design is a complex process that requires careful consideration of various factors to ensure the efficient and reliable operation of the system. Some of the critical design considerations for operating systems include power management, security and reliability, and performance monitoring and optimization. Power management is essential in modern computing systems as it helps to conserve energy, increase battery life, and reduce operational costs. Security and reliability are crucial in operating systems to prevent unauthorized access, data breaches, and system failures. Finally, performance monitoring and optimization are necessary to ensure efficient resource utilization and system responsiveness. By implementing various techniques such as task scheduling, memory management, and optimization, operating systems can monitor system performance and optimize it for optimal performance. Overall, careful consideration of these design considerations is crucial in developing robust and reliable operating systems. Following are the factors to consider:

## Thread

Threads are important in modern computing systems as they allow multiple tasks to be executed simultaneously, improving the overall performance and efficiency of the system. According to Silberschatz et al. (2018), threads are "lightweight" processes that enable multiple execution streams within a single process. Each thread has its own stack and program counter, but they share the same address space and resources as other threads within the process.

One important factor affecting thread performance is the number of available resources, such as memory and processing power. The more resources available, the more threads can be created and executed in parallel. However, too many threads can lead to contention for resources and decreased performance (Silberschatz et al., 2018).

There are several techniques for achieving thread in OS design, including user-level threading, kernel-level threading, and hybrid threading. User-level threading involves implementing threads entirely in user space, while kernel-level threading relies on the operating system kernel to manage threads. Hybrid threading combines elements of both approaches, utilizing both user-level and kernel-level threading for improved performance and flexibility (Silberschatz et al., 2018).

## Synchronization

Synchronization is critical in multiprocessor and multicore systems as it helps manage access to shared resources and ensures correct operation of concurrent processes. According to Silberschatz et al. (2018), synchronization involves coordinating the execution of processes and threads to prevent race conditions, deadlocks, and other problems that can arise when multiple processes access shared resources simultaneously.

There are various types of synchronization mechanisms used in OS design, including locks, semaphores, and monitors. Locks provide mutual exclusion, allowing only one process to access a shared resource at a time. Semaphores can be used to control access to a shared resource by multiple processes. Monitors are a higher-level synchronization construct that allows processes to wait for a condition to be true before proceeding (Silberschatz et al., 2018).

Achieving synchronization in OS design presents several challenges, such as balancing performance and fairness, avoiding deadlocks and livelocks, and minimizing overhead (Silberschatz et al., 2018).

## Scheduling

Scheduling is a crucial component of operating systems in multiprocessor and multicore systems, as it determines how system resources are allocated to processes and threads. According to Abraham Silberschatz et al. (2018), scheduling involves selecting the next process/thread to execute from the ready queue.

There are several types of scheduling algorithms, including First-Come, First-Served (FCFS), Round Robin, Priority, and Shortest Job First (SJF). FCFS and Round Robin are examples of non-preemptive scheduling, where the CPU is not taken away from the currently executing process until it voluntarily relinquishes it or completes its execution. Priority and SJF are examples of preemptive scheduling, where the CPU can be taken away from the currently executing process if a higher-priority process becomes ready to run.

Achieving scheduling in OS design can be accomplished using a variety of techniques, such as implementing priority levels, using time-sharing algorithms, and employing real-time scheduling methods (Silberschatz et al., 2018).

## Memory management

Memory management in a multiprocessor system faces similar challenges to a uniprocessor system, but the operating system must also utilize hardware parallelism to optimize performance, coordination between processors is necessary to ensure consistency when sharing pages and to determine page replacement, the primary concern is ensuring that a physical page's old contents are no longer accessible before being reused (Stallings, W., 2017).

Memory management in multicore systems is similar to that in multiprocessor systems. Each core in a multicore system has its own cache and local memory, but they share the same main memory. The operating system must manage the sharing and allocation of main memory among the cores efficiently to avoid performance degradation due to cache coherence and contention issues.

Figure 1 (multicore and multiprocessor memory management)

Source: (ScalerTopics, n.d.)

A diagram of a computer

Description automatically generated with low confidence

## Reliability and fault tolerance

Reliability and fault tolerance are crucial aspects of operating systems in multiprocessor and multicore systems. According to Tanenbaum and Bos (2015), fault tolerance refers to the ability of a system to continue functioning even in the presence of hardware or software faults, while reliability refers to the probability of a system to operate without failure over a specified time period. In multiprocessor and multicore systems, where multiple components are interconnected and interdependent, ensuring reliability and fault tolerance is essential to prevent system downtime and data loss.

To achieve efficient reliability and fault tolerance in OS design, various techniques can be employed. These include redundancy, checkpointing, and error detection and correction mechanisms (Tanenbaum & Bos, 2015). Duplicating essential components promotes redundancy by offering a fallback in the event of a failure.while fault tolerance systems can find and fix mistakes before they cause system failure, checkpointing includes periodically recording system state to aid recovery in the case of a failure.

## Other OS design Considerations

There are more significant factors that OS designers must take into account while building OS in addition to the fundamental design concerns for OS in multiprocessor and multicore systems. These considerations include power management, security and reliability, and performance monitoring and optimization.

### Power Management

Power management is an important design consideration for operating systems, especially in mobile devices and laptops. According to Walls (2014), effective power management can significantly extend the battery life of a device, reduce energy consumption, and contribute to sustainability. Operating systems achieve power management by implementing various techniques such as CPU frequency scaling, idle state management, and display brightness control. For instance, in the Linux operating system, the CPU frequency scaling technique reduces the frequency of the processor when the system is idle or underutilized, thereby reducing power consumption (Walls, 2014).

### Security and Reliability

Any OS design should take security and stability into account, but in multiprocessor and multicore systems, these factors take on much greater significance. These systems are more vulnerable to attacks, and problems with a single processor or core can have a negative impact on the entire system. To prevent unauthorised access and data breaches, OS designers must integrate security mechanisms like access limits and encryption. Additionally, they must put reliability measures in place, such as error detection and correction, to guarantee that the system keeps working even if some hardware fails.

### Performance Monitoring and Optimization

Performance monitoring and optimization are crucial in ensuring that the system runs efficiently and meets the desired performance goals. OS designers must provide tools for monitoring performance metrics such as CPU utilization, memory usage, and disk I/O, as well as for diagnosing performance issues. They must also optimize the system's performance by minimizing overhead, reducing latency, and improving throughput.

# Challenges and Future Treads

Operating systems for multiprocessor and multicore systems are constantly facing new challenges and opportunities due to the rapid development of hardware technology. As stated by Martin (n.d.), one of the challenges is the efficient distribution of tasks across multiple processors and cores while minimizing communication overhead. This is particularly challenging when dealing with irregular workloads that are difficult to partition. Additionally, power management becomes even more important in multiprocessor and multicore systems, as the increased number of processors and cores can lead to higher power consumption and heat dissipation, which can affect system stability and reliability (Martin, n.d.).

On the other hand, there are also opportunities for operating systems in multiprocessor and multicore systems. One of them is the potential for higher performance and throughput, as multiple processors and cores can work together to perform tasks in parallel. Another opportunity is the ability to implement more complex and advanced features, such as distributed computing and virtualization, which can improve resource utilization and scalability (Martin, n.d.).

In terms of future trends, operating systems for multiprocessor and multicore systems are likely to continue evolving to meet the demands of new hardware architectures and emerging applications. One trend is the use of heterogeneous computing, where different types of cores are combined to achieve better performance and energy efficiency. Another trend is the use of virtualization to enable efficient sharing of resources between different applications and users (Martin, n.d.).

# Conclusion

Modern computer systems often have multiple processors, each containing multiple cores, which allows for efficient execution of both single programs and multiple applications simultaneously. In summary, operating system design is extremely important for maximising the performance of contemporary computing systems. For the system to perform at its peak, design factors including hardware specifications, concurrent user count, and application kinds should be taken into account. For improving system performance and efficiency, multiprocessors and multicore processors work well. Operating systems must be designed with threads, synchronisation mechanisms, scheduling algorithms, and memory management strategies in order to perform at their best on multiprocessor and multicore systems. In order to accomplish synchronisation in OS design, performance and fairness must be balanced, deadlocks and livelocks must be avoided, and overhead must be kept to a minimum. Additionally, the memory management system should be designed to decide when to replace pages and guarantee consistency when sharing pages.

# Reference Lists

Javatpoint. (n.d.). Multiprocessor and multicore system in operating system [online] available at <<https://www.javatpoint.com/multiprocessor-and-multicore-system-in-operating-system> >[Accessed May 7, 2023]

Martin, C., (n.d.). Multicore Processors: Challenges, Opportunities, Emerging Trends [PDF]. embedded world Conference 2014. Available at < https://www.hs-augsburg.de/Binaries/Binary20964/Multicore-Embeddedfinal-revised.pdf >

ScalerTopics, n.d., *Difference Between Multicore and Multiprocessor System* [Diagram]. Accessed From

<https://scaler.com/topics/images/multicore_and_multiprocessor.webp>

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). Wiley.

Stallings, W., (2017). Operating Systems: Internals and Design Principles. Global Edition. Pearson.

Walls, C., 2014 Managing device power with an operating system [online] available at < https://www.embedded.com/managing-device-power-with-an-operating-system/> [Accessed May 8, 2023]