Course Name:	Operating Systems and Compilers	Semester:	VI
Date of Performance:	04 / 04 / 2025	Batch No:	B1
Faculty Name:	Prof. Nilesh Lakade	Roll No:	16014022050
Faculty Sign & Date:		Grade/Marks:	___ / 25

Experiment No.: 8

Title: To learn about various trends and issues in Memory management

Aim and Objective of the Experiment:

Analysis of Recent Trends and Issues in Memory Management.

COs to be achieved:

CO4: Understand Storage management with allocation, segmentation & virtual memory concepts.

Books/ Journals/ Websites referred:

1. Silberschatz A., Galvin P., Gagne G. “Operating Systems Principles”, Willey Eight edition.
2. Achyut S. Godbole , Atul Kahate “Operating Systems” McGraw Hill Third Edition.
3. William Stallings, “Operating System Internal & Design Principles”, Pearson.
4. Andrew S. Tanenbaum, “Modern Operating System”, Prentice Hall.

Theory

Report:

Papers referred to:

[1] https://www.ijmse.org/Volume7/Issue3/paper3.pdf
[2] https://www.sciencedirect.com/science/article/abs/pii/S0026271411004859
[3] https://kar.kent.ac.uk/14600/1/Dynamic.pdf

[1]

The research paper explores memory management techniques, detailing their evolution, effectiveness, and the challenges they currently face. It provides an overview of traditional and modern memory management methods, analysing how different strategies impact system performance, efficiency, and reliability. The paper discusses various techniques such as paging, segmentation, garbage collection, and virtual memory, highlighting their advantages and limitations. Additionally, the study examines advancements in dynamic memory allocation, caching strategies, and AI-driven optimizations to improve memory utilization.

Algorithms	Allocation time	De-Allocation Time
TLSF	O(1)	O(n)
Half Fit	O(1)	O(1)
Hoard	O(n)	O(1)
Tertiry Buddy	O(1)	O(n*)
Bitmapped	O(n)	O(1)
Buddy systems	O(log2n)	O(k)
Segregated Fit	O(1)	O(1)
Sequential Fit	O(n)	O(1)

Fig 11 Worst Case Time Complexity

The research paper compares various memory allocation algorithms based on allocation time, deallocation time, fragmentation, and response time.

1. Sequential Fit Algorithm:
   - Uses a doubly linked list to traverse memory blocks.
   - Best Fit Strategy minimizes fragmentation by finding an optimal memory block, splitting it, and freeing excess space.
   - However, it is slow due to traversal time and causes significant fragmentation.
2. Segregated Free List:
   - Works well when combined with other algorithms but, on its own, results in high fragmentation.
   - Used in advanced techniques like Hoard and Two-Level Segregated Fit (TLSF).
3. Indexed Fit Algorithm:
   - Similar performance to Bitmapped and Segregated Fit algorithms.
   - Efficient in some cases but not optimal for all workloads.
4. Two-Level Segregated Fit (TLSF):
   - Performs better than other algorithms due to its lower worst-case time.
   - Minimizes fragmentation and offers fast response times, making it ideal for real-time applications.

The research paper explores various dynamic memory allocation techniques, analyzing their efficiency based on internal fragmentation, response time, allocation and deallocation time, and memory footprint. Many of these techniques, including Sequential Fit, Segregated Fit, and TLSF (Two-Level Segregated Fit), are improvements over earlier approaches. Among them, TLSF is identified as the most suitable for real-time systems due to its low internal fragmentation, fast response time, and constant-time allocation and deallocation, making it significantly more efficient than traditional methods.

On the other hand, conventional memory allocators such as Segregated Fit, Indexed Fit, Bitmapped Fit, and the Buddy System are found to be inefficient for real-time applications due to higher fragmentation, slower response times, and greater memory overhead. Real-time systems require strict timing constraints and bounded rationality, making these traditional methods unsuitable. Instead, Hoard, Tertiary Buddy System, and TLSF are recommended for real-time applications as they provide faster response times and minimal fragmented memory, ensuring optimal performance in time-sensitive environments.

[2]

The research paper "Emerging Memory Technologies: Trends, Challenges, and Modeling Methods" explores the limitations of traditional memory technologies like DRAM and Flash, emphasizing the need for more efficient and durable memory solutions. With the demand for faster, energy-efficient, and non-volatile memory on the rise, this study investigates several emerging technologies that could potentially replace or complement existing memory solutions. It provides a detailed examination of emerging memory technologies such as Resistive RAM (ReRAM), Phase-Change Memory (PCM), Spin-Transfer Torque Magnetic RAM (STT-MRAM), Ferroelectric RAM (FeRAM), and 3D XPoint (Intel Optane), highlighting their advantages and challenges.

ReRAM offers fast switching, low power, and high endurance but faces reliability issues and variability in switching behavior. PCM provides better endurance than Flash and faster write speeds but suffers from high programming currents and scalability limitations. STT-MRAM delivers high-speed operation, low power, and excellent endurance but is hindered by high fabrication costs and energy consumption during switching. FeRAM is known for ultra-low power consumption and fast read/write speeds, though it struggles with scalability and a complex fabrication process. 3D XPoint bridges the performance gap between DRAM and Flash, offering better endurance and higher density than Flash, but it remains costly and has seen slow adoption.

The paper also explores the modeling and simulation methods used to assess these emerging technologies, including physical, compact, and system-level modeling. These approaches help predict the performance and feasibility of integrating new memory technologies into real-world applications.

Challenges for emerging memory technologies include scalability issues, energy efficiency, reliability, and high manufacturing costs. Many of these technologies are not yet ready to replace DRAM in large-scale applications, but advancements in material science, device engineering, and system integration are gradually making them viable.

Conclusion:
The paper concludes that while no single memory technology meets all criteria for a universal memory solution, ongoing research in emerging memory technologies could lead to hybrid architectures that optimize performance, energy efficiency, and cost-effectiveness.

[3]

The research paper "Dynamic Memory Management: Challenges for Today and Tomorrow" by Richard Jones focuses on the complexities of garbage collection (GC), a crucial component in modern memory management. The paper outlines the evolution of GC techniques, covering approaches such

as mark-sweep, reference counting, and generational collection, and their role in modern systems. It highlights the challenges faced by current GC systems, especially in high-performance and multiprocessor environments, where traditional methods struggle with handling large heaps and parallelism.

A key issue addressed is the performance overhead caused by GC, which introduces pauses and additional computational costs. These pauses are often seen as detrimental, especially in real-time systems where strict timing guarantees are essential. While incremental garbage collection methods help, they sometimes fail to meet the stringent demands of hard real-time applications. Another issue discussed is fragmentation, which occurs in non-moving collectors that leave objects in place, leading to degraded performance over time. Compacting the heap incrementally is challenging, particularly in concurrent systems, due to the need for atomic updates and the complexity of maintaining correctness.

The paper also discusses the growing challenges posed by multiprocessor systems, particularly those with high-core-count processors, and the difficulty in managing memory efficiently without impacting parallel execution. For embedded systems, the need for energy-efficient GC techniques is emphasized, with ongoing research exploring solutions like selectively powering down memory banks.

To address these issues, the paper proposes solutions such as improved generational garbage collectors, which minimize overhead by segregating objects into different generations based on their lifespan. Incremental compacting techniques are also suggested to reduce fragmentation and ensure more predictable performance. Future directions for GC include better integration with the operating system and hardware, and the development of real-time garbage collection methods for systems with strict timing requirements.

Jones concludes that while significant progress has been made in the field of garbage collection over the past 40 years, numerous challenges remain, especially with modern hardware and real-time, high-performance systems. He calls for continued research to refine GC techniques and adapt them to new computing environments.

Conclusion:

In conclusion, this experiment highlights the ongoing challenges and advancements in garbage collection (GC) and memory management.

Signature of faculty in-charge with Date: