OPERATING SYSTEM Paper 3

SECTION A

1. • Explain the architecture of a CPU based on the Von Neumann model and its relevance today.
2. • Discuss the importance of system buses in ensuring efficient data transfer in a CPU.
3. • Compare how the ALU and CU work together to execute instructions.
4. • Analyze the impact of bus width and clock speed on the overall performance of modern CPUs.
5. • How does the use of multiple cores affect the overall efficiency of a CPU?
6. • Evaluate how cache memory optimizes CPU operations and prevents bottlenecks.
7. • Discuss the trade-offs involved in overclocking a CPU.
8. • Explain how logical and arithmetic shifts can be used to optimize bitwise operations in computer programming.
9. • Analyze how interrupts are handled in multi-core systems.
10. • Discuss the advantages and disadvantages of HDMI compared to VGA.
11. • How does the fetch-execute cycle work in a pipelined processor architecture?
12. • Analyze how the use of interrupts allows for multitasking in modern operating systems.
13. • Evaluate the effectiveness of bit manipulation in controlling and monitoring devices.
14. • How does the clock speed affect a CPU's heat generation and power consumption?
15. • Discuss the limitations of USB technology in modern high-speed data transfer.
16. • How does bit manipulation with masking help manage multiple devices in a control system?
17. • Compare the efficiency of dual-core and quad-core processors in handling complex tasks.
18. • How does the CPU's status register manage multiple flags during complex operations?
19. • Explain the process of handling multiple interrupts in a real-time operating system.
20. • Discuss how binary shifts can be used to perform efficient division and multiplication in assembly language.
21. • Evaluate how HDMI's HDCP technology prevents illegal copying of high-definition content.
22. • Discuss the challenges of using older VGA technology in modern systems.
23. • How does the CPU handle conflicts between different interrupts with varying priorities?
24. • Analyze the role of the program counter (PC) and current instruction register (CIR) in optimizing CPU performance.
25. • How does the concept of bus width and word length affect memory access speed?

26. • Discuss how overclocking can lead to system instability and how to mitigate these effects.
27. • Evaluate the importance of using logical shifts for data manipulation in low-level programming.
28. • How does the CPU use masking to test, set, and clear bits in bit manipulation?
29. • Discuss the relationship between clock speed and the fetch-execute cycle.
30. • Analyze how multi-core CPUs improve performance in parallel processing tasks.
31. • How does an interrupt handler ensure system stability when multiple interrupts occur simultaneously?
32. • Discuss the role of binary shifts in efficient memory management in low-level programming.
33. • How does cache memory influence the speed of the fetch-execute cycle?
34. • Analyze the importance of status flags in detecting and handling errors during CPU operations.
35. • Discuss the benefits and limitations of using cyclic shifts in bitwise operations.
36. • Evaluate the impact of bus architecture on modern CPU designs.
37. • How do binary shifts enable faster computations in arithmetic operations?
38. • Analyze how different addressing modes (direct, indirect, indexed) affect CPU performance.
39. • Discuss how interrupt priority affects multitasking in an operating system.
40. • How do status flags help the CPU manage complex arithmetic operations?
41. • Evaluate the benefits and challenges of using quad-core processors for real-time applications.
42. • Discuss how interrupt service routines (ISR) are used to maintain system responsiveness.
43. • How does the CPU's use of bit manipulation improve device control and monitoring?
44. • Analyze the role of the control bus in managing signals across different CPU components.
45. • How does the width of the address bus limit the total amount of addressable memory in a system?
46. • Discuss how binary shifts can be used to optimize data compression algorithms.
47. • How does the CPU handle multiple flags in the status register during complex calculations?
48. • Evaluate the importance of interrupts in ensuring real-time performance in embedded systems.
49. • Discuss the challenges involved in increasing clock speed without overheating the CPU.
50. • How do modern CPUs handle conflicts between multiple devices requesting data via the system bus?

SECTION B

1. • Explain how virtual memory allows a system to execute programs larger than physical memory.
2. • Compare and contrast the four main scheduling algorithms: FCFS, SJF, SRTF, and Round Robin.
3. • How does the kernel optimize resource allocation during multitasking?
4. • Describe the process by which an OS handles a page fault.
5. • Analyze how disk thrashing can affect the overall performance of a system using virtual memory.
6. • How does the OS prevent data corruption when the system runs out of RAM?
7. • Evaluate the trade-offs between preemptive and non-preemptive multitasking.
8. • How do page tables manage logical-to-physical memory address translations?
9. • Analyze how different scheduling algorithms impact the efficiency of a system with limited resources.
10. • How does context switching affect the performance of an OS?
11. • Explain the advantages and disadvantages of different page replacement algorithms.
12. • Compare the efficiency of paging vs. segmentation in memory management.
13. • Discuss the limitations of optimal page replacement in practical use.
14. • How does the OS handle interrupts from multiple devices at once?
15. • What factors influence the effectiveness of page replacement algorithms in reducing page faults?
16. • How does the SRTF scheduling algorithm optimize turnaround time for processes?
17. • Discuss the concept of virtual memory and how it can be both beneficial and detrimental.
18. • How does the kernel manage communication between hardware, software, and memory during multitasking?
19. • Evaluate the effectiveness of the round-robin scheduling algorithm in a real-time system.
20. • Explain how memory optimization is achieved through paging and segmentation.
21. • How does the OS balance the use of RAM and virtual memory during process execution?
22. • What are the long-term effects of excessive disk thrashing on system hardware?
23. • Discuss the challenges involved in implementing dynamic memory allocation.
24. • How does the DMA controller improve the performance of input/output operations?
25. • What are the limitations of using a fixed time quantum in round-robin scheduling?
26. • Analyze how memory fragmentation affects system performance.
27. • How does the system detect and resolve page faults in real-time applications?

28. • Explain the role of the kernel in managing low-level scheduling conflicts.
29. • What are the challenges of managing memory in a system with multiple processes?
30. • How does an OS kernel optimize resource allocation for CPU-bound and I/O-bound processes?
31. • Evaluate the impact of process scheduling algorithms on system throughput.
32. • How does the OS prevent deadlock during process scheduling?
33. • How does a page fault interrupt differ from other types of interrupts?
34. • Discuss the role of swap space in improving the performance of virtual memory.
35. • Analyze how a system determines which page to replace during a page fault.
36. • What are the challenges of implementing a segmented memory management system?
37. • How do different scheduling algorithms handle I/O-bound processes?
38. • Explain how interrupt priority levels (IPLs) are used to manage multiple interrupts.
39. • Discuss the trade-offs between static and dynamic page replacement strategies.
40. • How does an OS manage high levels of concurrency in a multitasking environment?
41. • Analyze the impact of memory paging on system performance under heavy load.
42. • How does process state management ensure smooth task transitions?
43. • Explain the role of the kernel in handling hardware interrupts.
44. • Discuss the challenges of optimizing memory usage in a fragmented system.
45. • How do modern operating systems manage real-time process scheduling?
46. • Analyze the relationship between scheduling algorithms and process waiting time.
47. • How does a system maintain stability during high memory usage scenarios?
48. • What are the key differences between hardware and software interrupts?
49. • How does the OS kernel handle multiple interrupts from various devices?
50. • Discuss the challenges involved in scaling virtual memory systems in multi-core environments.

SECTION C

1. A 32-bit wide I/O device is connected to the main memory of a computer. The CPU can execute 3 × 1093 \times 10^93 × 109 instructions per second. Each instruction takes 6 processor cycles, with 4 cycles being used by the data bus. Each memory read/write operation requires two processor cycles. Assume the CPU is 75% utilized for tasks that do not involve an I/O operation.
   Estimate the data transfer rate using DMA for this system.
2. A 64-bit data bus connects an I/O device to the main memory of a system. The CPU is capable of executing 5 × 1095 \times 10^95 × 109 instructions per second, and each instruction uses 8 processor cycles, of which 5 are dedicated to the data bus. A memory read/write operation requires 2 processor cycles. The CPU spends 70% of its time on tasks unrelated to I/O operations.
   Determine the data transfer rate using DMA for this configuration.
3. An I/O device is connected to a system that uses a 64-bit data bus and runs at a clock frequency of 5 GHz. Each instruction takes 12 cycles to complete, with 8 of those cycles being used by the data bus for I/O operations. Each memory read/write operation takes 2 clock cycles.
   Assume that 25% of the total cycles are available for DMA data transfers. How much data (in gigabytes) can the system transfer in 1 second?
4. A system has a 128-bit wide data bus and the CPU can execute 3 × 1093 \times 10^93 × 109 instructions per second. The system uses DMA to transfer data, with each instruction taking 4 cycles, and 3 of those cycles are used for data transfer. A memory read/write operation requires one additional cycle.
   If DMA has control over the bus for 35% of the total time, calculate the maximum amount of data (in MB) that can be transferred in 10 milliseconds