D Y PATIL UNIVERSITY
PUNE I AMBI
॥ ज्ञानस्तीर्थं जगत् सर्वम् ॥
School of Engineering & Technology

## Department of CS

# SPOS Question Bank

---

1. **Explain the concept of System Software. What are its main components?**
2. **Describe the working and features of a Text Editor in system software.**
3. **What is a Loader? Explain its types and functions in detail.**
4. **Define an Assembler. How is it different from a compiler?**
5. **Explain the role of a Macro Processor in system programming with an example.**
6. **What is a Compiler? Describe its phases briefly.**
7. **Define a Debugger. What are its important functions in program execution?**
8. **Explain the concept of Machine Structure. How does it affect system software design?**
9. **Differentiate between Machine Language and Assembly Language with suitable examples.**
10. **Explain the general design procedure of an Assembler. Also describe the working of a two-pass assembler in detail.**
11. **Define Macro Instructions. Explain the basic concepts and advantages of using macros in system programming.**
12. **Describe the features of a macro facility. How does it improve program efficiency and modularity?**
13. **Explain the design and working of a two-pass macro processor with a suitable example.**
14. **Differentiate between single-pass and two-pass macro processors. Also explain the concept of a nested macro processor.**
15. **What is a Loader? Explain different loader schemes such as Compile-and-Go, General Loader Scheme, and Absolute Loader.**
16. **Describe subroutine linkage. How is it used during program loading and execution?**
17. **Explain the concepts of Relocating Loader and Direct Linking Loader. Highlight their advantages.**
18. **Discuss the design of an Absolute Loader. How does it function step-by-step?**
19. **Explain the design of a Direct Linking Loader. How does it handle external references?**

20. **What is a Linker? Explain relocation and linking concepts, and discuss static vs dynamic linking libraries along with the role of callback functions. Also briefly explain loading phases using Java as a case study.**
21. **Explain the role of lexical analysis in a compiler. Why is it considered the first phase of compilation?**
22. **Define Token, Pattern, and Lexeme with suitable examples.**
23. **What are lexical errors? Explain different types of lexical errors with examples.**
24. **Explain regular definitions used for language constructs such as identifiers, numbers, and keywords.**
25. **Describe how strings, sequences, and comments are handled during lexical analysis.**
26. **What is a transition diagram? Explain how it is used for recognition of tokens like identifiers and reserved words.**
27. **Explain the difference between reserved words and identifiers with examples.**
28. **Describe the general model of a compiler with a neat diagram.**
29. **Compare a Compiler and an Interpreter. Also explain the components of an interpreter and its uses.**
30. **Give an overview of LEX and YACC. Explain their specifications, features, and how they are used in compiler design.**
31. **Explain the concept of an Operating System. Discuss different types of operating systems with a focus on Real-Time Operating Systems (RTOS).**
32. **Describe the system components of an operating system and explain the various OS services provided to users and programs.**
33. **Explain the layered approach in operating system structure. What are its advantages and disadvantages?**
34. **Define a process. Explain the process states with a suitable diagram.**
35. **What is a Process Control Block (PCB)? Explain its structure and importance.**
36. **Define threads. Differentiate between processes and threads.**
37. **Explain different types of process schedulers (long-term, short-term, and medium-term schedulers).**
38. **Differentiate between preemptive and non-preemptive scheduling with examples.**
39. **Explain the following CPU scheduling algorithms with examples:**

    **FCFS (First Come First Serve)**

    **SJF (Shortest Job First)**

    **Round Robin**

    **Priority Scheduling**

40. **What is a deadlock? Explain methods of handling deadlocks including prevention, avoidance, detection, and recovery. Also briefly explain process management in multi-core operating systems.**
41. **Explain the programming model of the Intel 80386. Discuss its key features related to memory management.**
42. **Differentiate between contiguous and non-contiguous memory allocation with suitable examples.**
43. **What is swapping? Explain its working and advantages in memory management.**
44. **Explain the concept of paging. How does it eliminate external fragmentation?**
45. **Describe segmentation. How is it different from paging?**
46. **Explain segmentation with paging. Why is it used in modern systems?**
47. **What is virtual memory? Explain its need and advantages.**
48. **Describe demand paging. How does it improve system performance?**
49. **Explain the following page replacement algorithms with examples:**
    **FIFO (First In First Out)**
    **LRU (Least Recently Used)**
    **Optimal Page Replacement**
50. **What is thrashing? Explain its causes and how it can be controlled. Also discuss memory management in multi-core operating systems as a case study.**
51. **Explain different types of I/O Devices and their role in an operating system.**
52. **Describe the organization of I/O functions in an operating system. How does the OS manage communication between devices and CPU?**
53. **What is I/O buffering? Explain its types and advantages.**
54. **Explain various disk scheduling algorithms such as FIFO, LIFO, SSTF, SCAN, and C-SCAN with examples.**
55. **Define a file in an operating system. Explain different file access methods.**
56. **Describe different types of directory structures used in file systems.**
57. **What is file protection? Explain various protection mechanisms used in file systems.**
58. **Explain file system implementation and directory implementation in detail.**
59. **Describe different file allocation methods and free space management techniques.**
60. **Explain I/O and File Management in multi-core operating systems. Also compare lightweight and heavyweight operating systems with examples like Linux and Tizen.**