SPOS Viva Question–Answer Notes

1 & 2. Two Pass Assembler

Q1. What is an assembler?

An assembler converts assembly language programs into machine code. It translates mnemonics into opcodes and resolves symbols and literals.

Q2. What is a two-pass assembler?

A two-pass assembler scans the source program twice — Pass-I builds tables and intermediate code, Pass-II generates final machine code.

Q3. What are the functions of Pass-I?

Pass-I assigns addresses, builds the Symbol and Literal Tables, handles assembler directives, and produces intermediate code.

Q4. What are the functions of Pass-II?

Pass-II uses intermediate code, symbol, and literal tables to generate actual machine code instructions.

Q5. What are the key data structures used?

Symbol Table (SYMTAB), Literal Table (LITTAB), Machine Op Table (MOT), Pseudo Op Table (POT), Intermediate Code (IC).

Q6. What are assembler directives?

Instructions to the assembler such as START, END, DS, DC, ORIGIN, EQU. They do not produce machine code.

Q7. What is the role of the Location Counter (LC)?

LC keeps track of the current memory address for each instruction or variable.

Q8. What is intermediate code?

Intermediate code represents symbolic instructions with placeholders like (IS,04)(S,1) for easy translation in Pass-II.

3 & 4. Two Pass Macro Processor

Q1. What is a macro?

A macro is a group of assembly statements that can be invoked using a single name.

Q2. What is the function of a macro processor?

It expands macro definitions before the assembly process.

Q3. What are MNT, MDT, and ALA?

MNT (Macro Name Table) stores macro names and MDT indices. MDT (Macro Definition Table) stores macro statements. ALA (Argument List Array) stores arguments used in macros.

Q4. What happens in Pass-I?

Pass-I identifies macro definitions, builds MNT, MDT, and ALA, and removes macro definitions from the code.

Q5. What happens in Pass-II?

Pass-II expands macro calls using MNT, MDT, and ALA, replacing formal parameters with actual arguments.

Q6. Why are macros useful?

They provide code reusability, reduce repetition, and simplify programming.

5. Dynamic Link Library (DLL) and JNI

Q1. What is a DLL?

A Dynamic Link Library is a collection of reusable functions stored separately and linked at runtime.

Q2. What is JNI?

Java Native Interface allows Java programs to call native code written in C/C++.

Q3. Why are DLLs useful?

They allow modularity, code sharing between applications, and reduced memory usage.

6, 7 & 8. CPU Scheduling Algorithms

Q1. What is CPU Scheduling?

It determines the order in which processes are executed by the CPU.

Q2. What is SJF Scheduling?

Shortest Job First chooses the process with the smallest burst time first. It minimizes waiting time.

Q3. What is Priority Scheduling?

Each process is assigned a priority; higher-priority processes execute before lower-priority ones.

Q4. What is Round Robin Scheduling?

Processes are given equal time quantum in cyclic order. It is preemptive and fair.

Q5. Define waiting time and turnaround time.

Waiting time (WT) is total waiting in the ready queue. Turnaround time (TAT) = Completion - Arrival.

Q6. Define context switching.

Switching the CPU from one process to another, saving and restoring their states.

Q7. Define starvation and aging.

Starvation occurs when low-priority processes never execute. Aging gradually increases their priority to prevent this.

9. Synchronization Problems

Q1. What is process synchronization?

It ensures orderly execution of concurrent processes that share resources.

Q2. What are semaphores?

Synchronization tools using integer variables for signaling; they use wait() and signal() operations.

Q3. What is a mutex?

A mutual exclusion lock that allows only one thread to access a resource at a time.

Q4. What are classical problems?

Producer–Consumer, Reader–Writer, Dining Philosophers demonstrate synchronization and deadlock handling.

Q5. What is deadlock?

A situation where processes wait indefinitely for each other’s resources.

Q6. What is starvation?

A process is perpetually denied access to resources. Usually avoided by fair scheduling.

10–13. Memory Allocation Strategies

Q1. What is memory allocation?

It is the process of assigning memory blocks to processes in the main memory.

Q2. What are the types of allocation strategies?

First Fit, Best Fit, Next Fit, and Worst Fit.

Q3. Explain First Fit.

Allocates the first available block large enough for the process.

Q4. Explain Best Fit.

Allocates the smallest block that fits the process requirement.

Q5. Explain Worst Fit.

Allocates the largest available block to minimize leftover space.

Q6. What is fragmentation?

Wastage of memory due to small unusable gaps. Internal occurs inside allocated blocks; external occurs between free blocks.

14. Page Replacement Algorithms

Q1. What is paging?

A memory management technique that divides processes and memory into fixed-size pages and frames.

Q2. What is a page fault?

Occurs when a page referenced by a process is not present in physical memory.

Q3. Explain FIFO Page Replacement.

The oldest page in memory is replaced first.

Q4. Explain LRU Page Replacement.

Replaces the page that has not been used for the longest time.

Q5. Explain Optimal Page Replacement.

Replaces the page that will not be used for the longest time in the future. It gives minimum possible page faults.

Q6. Compare FIFO, LRU, and Optimal.

Optimal gives least faults but is theoretical; LRU is practical; FIFO is simplest but inefficient.

General Viva Questions

Q1. Difference between Assembler, Compiler, and Interpreter?

Assembler translates assembly to machine code; Compiler translates entire source to object code; Interpreter executes line by line.

Q2. What is internal vs external fragmentation?

Internal is unused space within allocated blocks; external is scattered free space between blocks.

Q3. What is context switching overhead?

The CPU time spent saving and restoring process states instead of executing instructions.

Q4. What is thrashing?

When excessive paging reduces CPU performance due to continuous page faults.

Q5. What is the role of operating system in memory management?

It keeps track of used and free memory, allocates and deallocates memory to processes efficiently.

Tip: Keep answers short and emphasize data structures, tables, and algorithm purpose. Revise examples and formulas for WT and TAT.