Operating Systems Lab Assignments Summary

Assignment 1: Introduction to OS and Process Basics

Part A: Introduction to OS (Shell Commands)

- **Objective:** Refresh knowledge of basic shell commands and scripting.
- **Tasks:**
  - Execute and report results/observations for commands: `date`, `cal`, `echo`, `man`, `ls`, `pwd`, `mkdir`, `cd`, `rmdir`, `cat`, `sort`, `cp`, `mv`, `rm`, `wc`, `head`, `tail`, `more`, `pipe`, `tr`, `chmod`, `chown`, `chgrp`.
  - Write a shell script to generate the Fibonacci series.

Part B: Process Basics

- **Objective:** Understand process creation, monitoring, and resource usage.
- **Tasks:**
  - Use commands (`ps`, `pstree`, `top`, `pgrep`, `renice`, `kill`, `xkill`) to monitor processes. Note memory consumption and states.
  - Write a C program using `fork()` to create a process (print PID/PPID).
  - Write a C program using `exec()` and analyze differences from `fork()`.
  - Write a program to store process state information (CPU time, memory, etc.) in a table structure.
  - Analyze resource assignment to ensure no single resource is assigned to multiple processes and check for repeated requests.

Assignment 2: CPU Scheduling

Setup: Create k processes with random arrival times (0-10) and CPU burst times (0-100).

Algorithms: Implement functions for:

- **FCFS** (First-Come, First-Served)
- **SJF** (Shortest Job First)
- **RR** (Round Robin) with time quanta: 5, 10, 15, 20, 25.

Analysis: Calculate and compare Turnaround Time (TAT), Average TAT, Response Time, and Waiting Time.

Assignment 3: Scheduling and Memory Management

Q1: CPU Scheduling

- Implement **Highest Response Ratio Next (HRRN)**.
- Implement **Multilevel Queue** scheduling.

Q2: Memory Allocation

- Implement **Best fit**, **Worst fit**, **First fit**, and **Next Fit** algorithms.
- Calculate internal and external fragmentation.

Q3: TLB Simulation

- Implement Translation Lookaside Buffer (TLB) based memory access.
- Compare experimental vs. theoretical time using hot ratio (h), physical memory time (T), and TLB access time (t).

Assignment 4: Paging, Shared Libraries, and Protection

Part 1: Simulating Shared Libraries

- **Objective:** Simulate an MMU where processes share a code segment.
- **Tasks:**
  - Construct Page Tables and a global Frame Table.
  - Map shared library pages to the same physical frames for all processes.
  - Map private data to unique frames.
  - Implement address translation and print tables.

Part 2: Memory Protection

- **Objective:** Enforce access rights (Read, Write, Execute).
- **Tasks:**
  - Modify Page Table Entry (PTE) to include permission flags.
  - Process memory requests: Check permissions $\rightarrow$ Translate address or report "PROTECTION FAULT".

Assignment 5: Address Translation Problems

Q1: Logical to Physical (Paging)

- **Given:** 128 pages, 1KB page size, 64 frames.
- **Task:** Translate logical addresses to physical addresses or identify Page Faults.

Q2: Two-Level Paging

- **Given:** 32-bit logical address (10-bit Directory, 10-bit Table, 12-bit Offset).
- **Task:** Translate addresses using provided Page Directory and Page Tables.

Q6: Segmentation

- **Given:** Segment Table with Base and Limit.
- **Task:** Calculate physical addresses or identify Segmentation Faults.

Assignment 6: Page Replacement Policies

Framework: Simulate demand paging with m frames and a reference string.

Algorithms: Implement and compare:

- **FIFO** (First-In, First-Out)
- **LRU** (Least Recently Used)
- **OPT** (Optimal)

Output: Trace frame contents step-by-step, count page faults, and check for Belady's anomaly.

Assignment 7: Working Set and Deadlock Detection

Q1: Working Set Model

- Calculate Working Set at specific times ($t=4, 6, 11, 18$) given a window size Δ.
- Analyze thrashing if working set size > available frames.

Q2: Deadlock Detection (Graph)

- Build a Resource Allocation Graph (RAG) using an adjacency matrix.
- Implement graph traversal (DFS) to detect cycles.

- Identify deadlocked processes if a cycle exists.

Assignment 8: Banker's Algorithm and Deadlock Simulation

Assignment 1: Banker's Algorithm

- Implement Safety Algorithm and Resource Request Algorithm.

Assignment 2: Detection Algorithm

- Implement logic to detect if existing processes are deadlocked.

Assignment 3: System Simulator

- **Mode A:** Deadlock Avoidance (Banker's).
- **Mode B:** Deadlock Detection (Grant all $\rightarrow$ Detect $\rightarrow$ Preempt).
- Compare performance (time taken, preemptions).

Assignment 9: Synchronization using Semaphores

Q1: Shared Counter

- Synchronize two threads incrementing a shared counter to prevent race conditions.

Q2: Producer-Consumer

- Simulate bounded buffer problem using semaphores.

Q3: Dining Philosophers

- Simulate 5 philosophers sharing forks, ensuring no two adjacent philosophers eat simultaneously.