Assignment 1

#1

| | Memory |
|---|---|
| 300 | 3005 |
| 301 | 5940 |
| 302 | 7006 |
| . | |
| . | |
| 940 | 0002 |
| 941 | |

Step 1: 3005 —> IR

Step 2: 3 —> AC

Step 3: 5940 —> IR

Step 4: 3 + 2 = 5 —> AC

Step 5: 7006 —> IR

Step 6: AC —> Device 6

#2

An I/O-bound program is a program that spends more time waiting for input/output operations to complete, like reading from a file or writing to a database.

On the other hand, a processor-bound program is one that spends more time using the processor, like performing complex calculations.

The short-term scheduling algorithm is an algorithm that favors programs which have used little processor time in the recent past.

This algorithm is designed to give priority to programs that have been waiting to use the processor. In other words, it's trying to be fair to all programs and make sure they get a turn.

Since I/O-bound programs spend more time waiting for I/O operations, they would have used less processor time in the recent past. This means that the short-term scheduling algorithm would give them higher priority and allocate more processor time to them.

But this does not mean that processor-bound programs are permanently denied processor time. They would still get a turn, just maybe not as often as the I/O-bound programs.

In conclusion, short-term scheduling algorithm is trying to balance the needs of both types of programs, giving I/O-bound programs more processor time since they spend more time waiting, while not completely ignoring processor-bound programs. It's a way of making sure all programs get a fair shot at using the processor.

#3

Time-sharing and multiprogrammed batch systems are two different types of operating systems. And the scheduling policies used for optimizing them are also different.

For a time-sharing system, the goal is to make sure that each user gets a fair amount of processing time and that the system is responsive to their needs.

The scheduling policy in this case would be a priority-based or a round-robin scheduling policy.

The priority-based policy would prioritize tasks based on the priority assigned to each user, while the round-robin policy would allocate processing time to each user in a cyclic manner, ensuring that no user is ignored.

On the other hand, the goal of a multiprogrammed batch system is to maximize the use of the processor and reduce wait times for the user.

In this case, the scheduling policy would be a first-come-first-serve or a shortest-job-first policy.

The first-come-first-serve policy would allocate processing time to tasks in the order that they were received, while the shortest-job-first policy would prioritize tasks based on their expected completion time, making sure that tasks that are expected to complete faster are processed first.

The scheduling policies used in a time-sharing system prioritize responsiveness and fairness, while the policies used in a multiprogrammed batch system prioritize efficiency and reducing wait times. These policies are tailored to the specific goals of each operating system, ensuring optimal performance.

#4

Time = 22

| Process ID | Status | Event on which it is blocked |
|---|---|---|
| P1 | Blocked | time 5: P1 executes a command to read from disk unit 3. |
| P3 | Blocked | time 20: P3 executes a command to read from disk unit 2. |
| P5 | Ready | time 15: P5's time slice expires. |
| P7 | Blocked | time 18: P7 executes a command to write to disk unit 3. |
| P8 | Running | time 38: P8 terminates. |

Time = 37

| Process ID | Status | Event on which it is blocked |
|---|---|---|
| P1 | Ready | time 36: An interrupt occurs from disk unit 3: P1's read is complete. |
| P3 | Ready | time 33: An interrupt occurs from disk unit 2: P3's read is complete. |
| P5 | Suspended | time 28: P5 is swapped out. |
| P7 | Blocked | time 18: P7 executes a command to write to disk unit 3. |
| P8 | Running | time 38: P8 terminates. |

Time = 47

| Process ID | Status | Event on which it is blocked |
|---|---|---|
| P1 | Ready | time 36: An interrupt occurs from disk unit 3: P1's read is complete. |
| P3 | Ready | time 33: An interrupt occurs from disk unit 2: P3's read is complete. |
| P5 | Ready | time 44: P5 is swapped back in. |

| Process ID | Status | Event on which it is blocked |
|---|---|---|
| P7 | Blocked | time 18: P7 executes a command to write to disk unit 3. |
| P8 | Terminated | time 38: P8 terminates. |

#5

The possible transitions between the seven states are:

1. New to Ready/Suspend: The process is created but has not yet been scheduled for execution. It could be moved to the Ready/Suspend state if it has been placed in the queue but is not immediately eligible to run. (Example: Operating system scheduling policies.)

2. New to Ready: The process is created and placed in the ready queue to wait for execution. (Example: Operating system scheduling policies.)

3. Ready/Suspend to Ready: The process is resumed and placed back in the ready queue. (Example: Operating system scheduler resuming the process due to a change in system resources.)

4. Ready to Running: The process is selected by the operating system scheduler to run. (Example: Operating system scheduler selecting the process with the highest priority from the ready queue.)

5. Running to Ready: The process is preempted by a higher-priority process. (Example: Operating system scheduling policies, an interrupt caused by a higher-priority process.)

6. Running to Blocked/Suspend: The process requests an I/O operation that cannot be immediately satisfied. (Example: Disk I/O, network I/O, waiting for user input.)

7. Running to Exit: The process completes its execution. (Example: The process reaches the end of its code, an error occurs, a system call to exit is made.)

8. Blocked/Suspend to Blocked: The process remains blocked because the event it was waiting for has not occurred. (Example: A disk I/O operation is taking longer than expected.)

9. Blocked to Ready: The process's event has occurred and the process is now able to run. (Example: A disk I/O operation has completed, network I/O has completed, user input has been received.)

The impossible transitions between the seven states are:

1. Exit to Ready: Once a process has completed execution, it cannot return to the ready state.

2. Exit to Blocked/Suspend: Once a process has completed execution, it cannot be blocked or suspended.

3. Blocked to Running: A process cannot move directly from blocked to running. It must first be placed in the ready state and then selected to run by the operating system scheduler.

4. Blocked/Suspend to Running: A process cannot move directly from blocked/suspend to running. It must first be resumed and placed in the ready state and then selected to run by the operating system scheduler.

5. New to Running: A process cannot move directly from new to running. It must first be placed in the ready queue and then selected to run by the operating system scheduler.

6. New to Blocked/Suspend: A process cannot be blocked or suspended before it is scheduled to run.

7. Ready to Blocked/Suspend: A process cannot be blocked or suspended while it is waiting in the ready queue.

8. Ready/Suspend to Exit: A process that is suspended cannot complete execution. It must first be resumed and then run to completion.