1. ¿Cómo funciona una pila o stack? ¿Cuál es su punto de acceso, su límite y su base? ¿Es una lista de tipo LIFO o FIFO?

- Una pila es un conjunto ordenado de elementos tal que en cada momento solo se puede acceder al más recientemente añadido. El punto de acceso es la cima de la pila, la base es la primera posición de la pila (evita intentar extraer algo de una pila vacía) y el límite es la dirección de la cima del bloque (evita intentar agregar algo a una pila llena). La pila es una lista tipo LIFO ya que el último en ingresar es el primero en salir .

2. ¿Qué utilidad puede tener la utilización de una estructura de tipo pila o stack? ¿Qué implicaciones en cuanto a la programación tiene este tipo de estructura?

- **Utilidad:** Las pilas son útiles para gestionar la memoria en la llamada de funciones, almacenar el contexto de ejecución, realizar cálculos aritméticos y de expresiones, y manejar retrocesos en algoritmos.
- **Implicaciones:** En programación, el uso de pilas facilita la implementación de recursión, el manejo de variables locales y la realización de operaciones de deshacer/rehacer. También requiere una gestión cuidadosa para evitar desbordamientos de pila (stack overflow).

3. ¿Qué relación hay entre una pila o stack y las operaciones PUSH y POP?

- **PUSH:** Es la operación que añade un elemento al tope de la pila.
- **POP:** Es la operación que elimina el elemento del tope de la pila.

- **Relación:** PUSH y POP son las operaciones básicas que permiten insertar y eliminar elementos en una pila siguiendo el principio LIFO.

4. ¿Cuáles son los objetivos de un SO?

- **Facilidad de uso:** facilitar el uso del computador.
- **Eficiencia:** administrar los recursos para un mejor rendimiento.
- **Capacidad para evolucionar:** se deben poder desarrollar e introducir nuevas funciones sin interferir con su servicio.

5. Una vez que el SO está en funcionamiento, ¿El núcleo o kernel siempre permanece cargado en memoria? O bien el mismo se carga y descarga de la memoria según lo requiera la administración de memoria del SO.

- Sí, el núcleo o kernel siempre permanece cargado en memoria una vez que el sistema operativo está en funcionamiento. Esto es necesario para que el sistema operativo pueda gestionar recursos, manejar interrupciones y ejecutar otras funciones críticas en cualquier momento.

6. ¿Cuáles eran los principales problemas del procesamiento en Serie de los primeros SO's?

- **Baja Utilización del CPU:** El CPU estaba inactivo mientras esperaba operaciones de E/S.
- **Tiempo de Espera Prolongado:** Los trabajos tenían que esperar en cola, incrementando el tiempo de espera.

- **Interacción Limitada:** No había interacción en tiempo real con los usuarios.
- **Ineficiencia General:** El sistema no aprovechaba eficientemente los recursos disponibles.

7. En el procesamiento en Serie, ¿había algún tipo de aislación (isolation) entre el usuario y el hardware utilizado?

- No, en el procesamiento en serie no había una aislación efectiva entre el usuario y el hardware. Los usuarios tenían acceso directo a los recursos del hardware, lo que podía llevar a errores y conflictos.

8. En el procesamiento batch o por lotes, ¿había algún tipo de aislación (isolation) entre el usuario y el hardware utilizado?

- Sí, en el procesamiento por lotes había una mayor aislación entre el usuario y el hardware. Los trabajos se agrupaban y se procesaban sin intervención directa del usuario, permitiendo al sistema operativo gestionar los recursos de manera más eficiente.

9. ¿De qué manera el usuario podía interactuar con el Monitor?

- En los sistemas de procesamiento por lotes, el usuario interactuaba con el Monitor (SO) mediante el envío de trabajos en lotes, usualmente a través de tarjetas perforadas o cintas magnéticas. El Monitor luego ejecutaba los trabajos en secuencia.

10. En un sistema de procesamiento batch o por lotes, ¿Qué sucede cuando un programa del usuario ejecuta una instrucción privilegiada (asumiendo que el hardware tiene la facilidad de contar con este tipo de instrucciones)? ¿Cuál cree Ud. que es la razón de ello?

- Cuando un programa de usuario ejecuta una instrucción privilegiada, el hardware genera una interrupción o trampa (trap) y transfiere el control al sistema operativo. El SO maneja la instrucción de acuerdo con las políticas de seguridad y protección. Esto se hace para evitar que los programas de usuario interfieran con el control del sistema.

11. Las instrucciones privilegiadas incluyen también operaciones de I/O, entonces, ¿Cómo es posible que un programa de usuario pueda realizar operaciones de I/O?

- Los programas de usuario realizan operaciones de I/O a través de llamadas al sistema (system calls). Estas llamadas solicitan al sistema operativo que ejecute las operaciones de I/O en su nombre, asegurando que se mantenga el control y la seguridad.

12. ¿Qué áreas de memoria están disponibles para un programa de usuario cuando éste se ejecuta en “modo usuario” (user mode)?

- En “modo usuario”, un programa tiene acceso a su propio espacio de direcciones, que incluye la memoria asignada para su código, datos y pila. No tiene acceso directo a la memoria del sistema ni a la de otros programas.

13. ¿Qué áreas de memoria están disponibles para el monitor teniendo en cuenta que éste se ejecuta en “modo núcleo” (kernel mode)?

- En “modo núcleo”, el monitor (o kernel) tiene acceso a toda la memoria del sistema, incluyendo la memoria del sistema operativo, la memoria de los programas de usuario, y las áreas reservadas para la gestión de hardware.

14. En un sistema de tiempo compartido o time sharing, ¿el sistema de procesamiento batch o por lotes es reemplazado por la terminal? O bien son sistemas en donde pueden convivir tanto el procesamiento batch como procesos interactivos

- En un sistema de tiempo compartido, los sistemas de procesamiento batch y los procesos interactivos pueden coexistir. La terminal permite la interacción en tiempo real con los usuarios, mientras que los trabajos batch se pueden ejecutar en segundo plano, permitiendo un uso eficiente de los recursos del sistema.