TRABAJO PRACTICO N° 4

Cuestionario Integrador – Unidad I

Materia: Arquitectura y Sistemas Operativos

Profesor: BENÍTEZ, Teresa

Grupo:

- ALIENDE, Héctor
- AGÜERO, Rosa
- BERTOZZI, Fernando
- SIERRA, Enzo
- STEVENS, Danilo

Comisión: 1.2

AÑO 2024

GP 4: Cuestionario Integrador Unidad I

El siguiente cuestionario sirve como guía de la Unidad I, con el objetivo de ayudar a repasar y asegurarse de que se han comprendido los conceptos clave relacionados con la arquitectura y sistemas operativos de computadoras. Es importante que cada estudiante desarrolle sus respuestas de manera detallada, utilizando ejemplos cuando sea posible para demostrar su comprensión de los conceptos.

Organización y Arquitectura de Computadoras

1. Defina el concepto de computadora elemental.
2. Explique qué es la arquitectura de Von Neumann y describa sus principales componentes.
3. Arquitectura Harvard, diferencias con VonNeumann
4. Arquitecturas RISC y CISC. Conceptos y diferencias.

CPU

5. Defina qué es una CPU y explique su importancia en una computadora.
6. Describa qué son la Unidad de Control (UC) y la Unidad Aritmético Lógica (UAL) y la función de cada una dentro de la CPU.
7. Explique el concepto de programas e instrucciones desde la perspectiva del funcionamiento de la CPU.
8. ¿Cómo procesa la CPU los datos para realizar operaciones?

Memoria

9. Diferencie entre memoria principal y memorias secundarias, proporcionando ejemplos de cada tipo.
10. Explique qué es la RAM y su rol dentro de la computadora.
11. Describa el propósito y funcionamiento de los dispositivos de almacenamiento fuera de línea y los dispositivos extraíbles.

Entrada y Salida de Datos

12. Defina el concepto de periféricos y clasifíquelos en categorías según su función.
13. Explique los mecanismos básicos de procesamiento de entrada/salida (E/S).
14. ¿Cómo gestiona un sistema operativo los dispositivos de E/S y qué problemas puede solucionar al hacerlo?

Software de Base

15. Introduzca el concepto de sistemas operativos y explique su importancia en la gestión de recursos informáticos. Ejemplos
16. Defina qué es un kernel y describa los tipos de kernels que existen.
17. Explique cómo el diseño y la implementación de un kernel afectan al rendimiento y seguridad de un sistema operativo.

DESARROLLO

1. Una **computadora elemental** es un dispositivo diseñado para realizar cálculos y procesar datos mediante un conjunto básico de instrucciones. Está formada por componentes esenciales como una unidad de procesamiento (CPU), memoria y dispositivos de entrada/salida. Su funcionamiento se basa en la ejecución de instrucciones que transforman datos de entrada en resultados de salida.
2. La **arquitectura de Von Neumann** es un modelo de diseño de computadoras propuesto por John von Neumann en 1945. Es la base de la mayoría de las computadoras modernas y se caracteriza porque el programa y los datos se almacenan en la misma memoria.

   Principales componentes:

   1. **Unidad Central de Procesamiento (CPU):**
      - **Unidad de Control (CU):** Coordina y controla la ejecución de instrucciones.
      - **Unidad Aritmético-Lógica (ALU):** Realiza cálculos matemáticos y operaciones lógicas.
   2. **Memoria Principal:** Almacena datos e instrucciones que la CPU necesita para operar.
   3. **Dispositivos de Entrada/Salida (I/O):** Permiten la interacción entre la computadora y el usuario o el mundo exterior.
   4. **Bus de Comunicación:** Sistema de transmisión que conecta todos los componentes de la computadora.

3. La **arquitectura Harvard** es un modelo de diseño en el que la memoria de datos y la memoria de instrucciones están separadas. Esto permite que los datos y las instrucciones se procesen simultáneamente, aumentando la eficiencia en ciertos contextos.

   Diferencias principales:

Característica	Arquitectura Von Neumann	Arquitectura Harvard
Memoria	Una única memoria para datos e instrucciones.	Memorias separadas para datos e instrucciones.
Velocidad	Procesa instrucciones y datos de forma secuencial.	Procesa instrucciones y datos en paralelo.
Complejidad de diseño	Más sencilla.	Más compleja y costosa de implementar.
Uso común	Computadoras personales, servidores.	Sistemas embebidos y microcontroladores.

4.

   1. **RISC (Reduced Instruction Set Computer):**
      - Diseñada con un conjunto reducido y optimizado de instrucciones.
      - Cada instrucción tiene un tamaño fijo y se ejecuta en un solo ciclo de reloj.
      - Ejemplo: ARM, MIPS.
   2. **CISC (Complex Instruction Set Computer):**
      - Utiliza un conjunto amplio y complejo de instrucciones.

- Cada instrucción puede realizar múltiples operaciones, pero toma varios ciclos de reloj.
- Ejemplo: x86 (Intel, AMD).

Diferencias principales:

Característica	RISC	CISC
Conjunto de instrucciones	Reducido y simple.	Amplio y complejo.
Ejecución	Instrucciones rápidas y sencillas.	Instrucciones más lentas y complejas.
Consumo de energía	Bajo (ideal para dispositivos móviles).	Mayor (más adecuado para PCs).
Tamaño del código	Mayor, requiere más instrucciones para tareas complejas.	Menor, al usar instrucciones complejas.

5. La **CPU (Unidad Central de Procesamiento)** es el componente principal de una computadora encargado de interpretar y ejecutar las instrucciones de los programas. Es conocida como el "cerebro" de la computadora, ya que realiza cálculos, toma decisiones y gestiona el flujo de datos entre los distintos componentes del sistema. Su importancia radica en que determina la velocidad y capacidad de procesamiento del dispositivo.

6.

- **Unidad de Control (UC):** Coordina las operaciones de la computadora. Interpreta las instrucciones de los programas y dirige el flujo de datos entre la CPU, la memoria y los dispositivos periféricos.
- **Unidad Aritmético-Lógica (UAL):** Realiza operaciones matemáticas (suma, resta, etc.) y lógicas (AND, OR, NOT). Es esencial para realizar cálculos y evaluaciones de condiciones.

Ambas trabajan juntas para garantizar que las instrucciones se ejecuten correctamente y de manera eficiente.

7. Los **programas** son conjuntos de instrucciones diseñadas para realizar tareas específicas. Desde la perspectiva de la CPU, una **instrucción** es un comando que indica una operación específica, como un cálculo, el movimiento de datos o una decisión lógica. La CPU ejecuta estas instrucciones de manera secuencial, descomponiéndolas en operaciones más simples según su arquitectura
.
8. La CPU procesa los datos siguiendo estas etapas principales:

1. **Fetch (Obtención):** Recupera las instrucciones desde la memoria.
2. **Decode (Decodificación):** Interpreta la instrucción para determinar qué operación realizar.
3. **Execute (Ejecución):** Realiza la operación utilizando la UAL si es necesario.
4. **Write-back (Escritura):** Guarda los resultados en la memoria o registros para su uso posterior.

Este ciclo, conocido como **ciclo de instrucción**, se repite constantemente mientras la CPU esté en funcionamiento.

9.

- **Memoria principal:**
  Es la memoria que la CPU utiliza directamente para almacenar datos e instrucciones temporalmente mientras los procesa. Es rápida y volátil, lo que significa que pierde su contenido al apagarse el dispositivo.
  Ejemplos: RAM, ROM, caché.
- **Memorias secundarias:**
  Almacenan datos de manera permanente o a largo plazo. Son más lentas que la memoria principal pero tienen mayor capacidad.
  Ejemplos: Discos duros (HDD), unidades de estado sólido (SSD), y unidades ópticas (CD/DVD).

10. La **RAM (Memoria de Acceso Aleatorio)** es un tipo de memoria principal volátil utilizada para almacenar temporalmente los datos e instrucciones de los programas que la CPU necesita durante su operación. Su rol es crucial porque permite acceder rápidamente a la información, lo que acelera el procesamiento de tareas en comparación con el acceso a la memoria secundaria.

11.

- **Dispositivos de almacenamiento fuera de línea:**
  Permiten almacenar datos de forma independiente del sistema principal y suelen usarse para respaldo o transporte de información.
  Ejemplos: Discos duros externos, cintas magnéticas.
- **Dispositivos extraíbles:**
  Son medios de almacenamiento portátiles diseñados para conectarse y desconectarse fácilmente de la computadora. Se utilizan para transferir datos entre dispositivos o como almacenamiento temporal.
  Ejemplos: Memorias USB, tarjetas SD.

Ambos son importantes para el manejo de grandes volúmenes de datos y garantizar la portabilidad y seguridad de la información.

12. Los **periféricos** son dispositivos externos que se conectan a una computadora para permitir la interacción con el usuario o con otros sistemas. Según su función, se clasifican en:

- **Periféricos de entrada:** Permiten introducir datos en la computadora.
  Ejemplos: Teclado, mouse, escáner.
- **Periféricos de salida:** Muestran o transmiten la información procesada por la computadora.
  Ejemplos: Monitor, impresora, altavoces.
- **Periféricos de entrada/salida (E/S):** Realizan ambas funciones, tanto de entrada como de salida.
  Ejemplos: Pantallas táctiles, discos externos, memorias USB.

13. Los mecanismos básicos de procesamiento de entrada/salida (E/S) son:

    1. **E/S Programada:** La CPU controla directamente los dispositivos de E/S, esperando que cada operación termine antes de continuar. Es simple pero ineficiente.
    2. **E/S por Interrupciones:** El dispositivo notifica a la CPU mediante interrupciones cuando necesita atención, liberando a la CPU para realizar otras tareas mientras tanto.
    3. **Acceso Directo a Memoria (DMA):** Un controlador especializado transfiere datos entre la memoria y los dispositivos de E/S sin intervención de la CPU, aumentando la eficiencia.

14. El sistema operativo gestiona los dispositivos de E/S mediante:

    - **Controladores (drivers):** Traducen las instrucciones del sistema operativo a comandos específicos del hardware.
    - **Planificación de E/S:** Coordina el acceso a los dispositivos para evitar conflictos entre múltiples procesos.
    - **Buffering y Caching:** Almacena temporalmente datos en memoria para optimizar el rendimiento y reducir los tiempos de espera.

    Problemas que puede solucionar:

    1. **Conflictos de recursos:** Asegura que múltiples procesos puedan compartir dispositivos de forma ordenada.
    2. **Dispositivos lentos:** Utiliza técnicas como el almacenamiento en búfer para minimizar el impacto de dispositivos más lentos.
    3. **Errores en dispositivos:** Detecta y gestiona fallos en el hardware para evitar que afecten el sistema completo.

15. Los **sistemas operativos (SO)** son software de base que actúan como intermediarios entre el hardware de una computadora y los programas que el usuario utiliza. Son esenciales para gestionar recursos como la CPU, memoria, almacenamiento y dispositivos de entrada/salida, asegurando un funcionamiento eficiente y coordinado.

    Importancia:

    - Permiten la ejecución de múltiples aplicaciones simultáneamente.
    - Proveen una interfaz de usuario (gráfica o de comandos).
    - Gestionan la seguridad y estabilidad del sistema.

    Ejemplos:
    Windows, macOS, Linux, Android, iOS.

16. El **kernel** es el núcleo del sistema operativo, encargado de gestionar los recursos del sistema y de mediar entre el hardware y el software. Controla procesos, memoria, dispositivos de E/S y la seguridad del sistema.

    Tipos de kernels:

1. **Monolíticos:** Todo el código del kernel se ejecuta en un único espacio de memoria, lo que los hace rápidos pero difíciles de depurar.
   Ejemplo: Linux.
2. **Microkernels:** Dividen las funciones del kernel en pequeños módulos, minimizando el código que se ejecuta en el núcleo para mejorar la seguridad y estabilidad.
   Ejemplo: Minix.
3. **Híbridos:** Combinan características de monolíticos y microkernels para un equilibrio entre rendimiento y modularidad.
   Ejemplo: Windows NT.
4. **Exokernels:** Permiten que las aplicaciones gestionen directamente los recursos, brindando gran flexibilidad.
   Ejemplo: Aegis.

17. El diseño y la implementación del kernel afectan el rendimiento y la seguridad de un sistema operativo de las siguientes maneras:

    - **Rendimiento:**
      - Kernels monolíticos tienden a ser más rápidos porque evitan la sobrecarga de comunicación entre módulos.
      - Microkernels pueden ser más lentos debido a la mayor cantidad de interacciones entre componentes.
    - **Seguridad:**
      - Microkernels ofrecen mayor seguridad porque separan funcionalidades, limitando el impacto de fallos o vulnerabilidades.
      - Los kernels monolíticos son más vulnerables a fallos porque cualquier error en un módulo puede afectar todo el sistema.

    Un diseño balanceado es clave para un sistema operativo que sea eficiente y seguro, adaptándose a las necesidades específicas del usuario o entorno.