Trabajo Práctico 1

Arquitectura de Computadoras

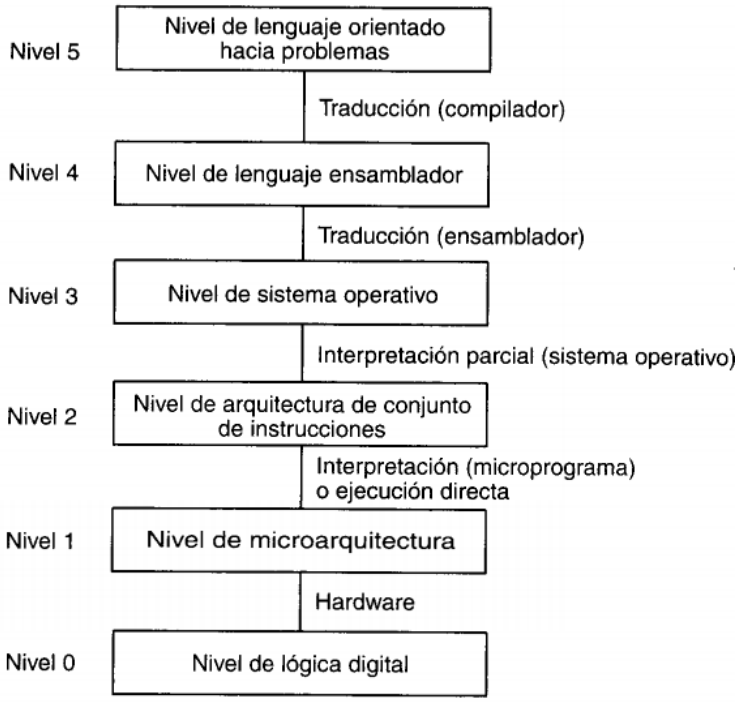
1. **¿Por qué las computadoras se estructuran en capas?**

Las computadoras se organizan en **capas** o **niveles** para facilitar el trabajo a los programadores de aplicaciones. Esta estructuración en capas implica diseñar un nuevo conjunto de instrucciones (**lenguaje**) que sea más fácil de utilizar para las personas que el **lenguaje de máquina** (es decir, el conjunto de instrucciones primitivas que los circuitos electrónicos de una computadora pueden reconocer y ejecutar directamente).

El número de capas de una computadora puede aumentar indefinidamente hasta que se llega a la comodidad deseada. Cada uno de los niveles superiores se basa en su nivel inferior. El nivel más alto es el más cómodo y simple para la persona, mientras que el nivel más bajo es el más simple y rápido de ejecutar para la computadora.

Cabe destacar que, a la hora de ejecutar un programa desarrollado en un lenguaje de alto nivel, estas instrucciones de alto nivel deberán ser “transformadas” en instrucciones que la electrónica de la computadora pueda comprender y ejecutar, es decir instrucciones de bajo nivel. Para ello se utilizan principalmente dos técnicas diferentes llamadas **interpretación** y **traducción**.

Para que estos procesos o técnicas puedan ser llevados a cabo, los lenguajes entre una capa y la siguiente no deben ser muy diferentes, es por eso que las computadoras generalmente se estructuran de la forma que se muestra en la siguiente figura.



Estructura de niveles de una computadora moderna

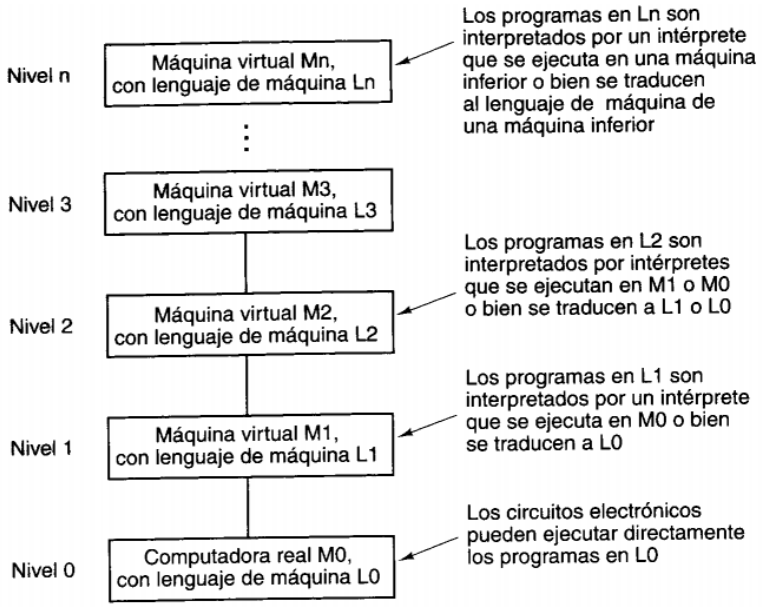
2. **Describa la relación entre capas, máquinas virtuales y lenguajes**

Una **máquina virtual** (M) se puede imaginar como una computadora hipotética que puede ejecutar directamente instrucciones en un determinado lenguaje (L). Así por ejemplo,

podemos pensar en una máquina virtual M1 que pueda ejecutar directamente instrucciones en un lenguaje L1, superior al lenguaje de máquina L0 que puede ser ejecutado por la computadora real M0.

Así, cada máquina virtual posee un cierto lenguaje, que consiste en el conjunto de instrucciones que dicha máquina imaginaria es capaz de ejecutar directamente y, a su vez, cada máquina virtual representa una capa de la computadora.

Por lo tanto, una computadora de n capas puede verse como n máquinas virtuales distintas, cada una con un lenguaje diferente. Los programas escritos en el nivel n deberán ser traducidos o interpretados en un nivel más bajo, llegando eventualmente hasta el nivel 0, para que realmente se ejecuten en la computadora real.



Esquema de una computadora multinivel

3. Describa el concepto de trayectoria de datos

El término **trayectoria de datos** hace referencia al camino por donde fluyen los datos en el nivel de microarquitectura. Básicamente, la trayectoria de datos consiste en seleccionar uno o dos registros, hacer que la ALU (**Unidad Aritmética Lógica**) opere sobre ellos, por ejemplo sumándolos, y luego almacenar el resultado en un registro.

En algunas computadoras existe un software, llamado **microprograma**, encargado de controlar la operación de la trayectoria de datos. En otras máquinas, esto es controlado directamente por el hardware de la computadora.

4. ¿Qué es el microprograma y cuál es su función?

Como mencionamos anteriormente, el **microprograma** es un software encargado de controlar la trayectoria que siguen los datos. Básicamente, es un intérprete de las instrucciones del nivel 2 (nivel ISA), lo que hace es obtener, interpretar y ejecutar las instrucciones una por una.

Por ejemplo, para una operación ADD, el microprograma obtendría la instrucción y la interpretaría, luego localizaría los operandos y los colocaría en unos registros determinados, la ALU calcularía la suma y luego el resultado se enviaría al lugar correcto.

En una máquina donde la trayectoria de datos es controlada por el hardware, este proceso sería idéntico, pero sin un programa almacenado que controle la interpretación de las instrucciones del nivel 2.

5. **Realice una tabla con las capas de una arquitectura típica, y comente brevemente la función de cada una**

Nivel	Función y Características
Nivel 0 o Nivel de Lógica Digital	Este nivel representa el hardware de la computadora, el cual es capaz de ejecutar hasta instrucciones del nivel 1, escritas en lenguaje de máquina. Los elementos que integran este nivel se llaman **compuertas**. Cada compuerta tiene una o varias entradas digitales y, en base a ellas, calcula alguna función sencilla (como AND u OR). Las compuertas pueden combinarse para formar **registros** y otros circuitos lógicos.
Nivel 1 o Nivel de Microarquitectura	Este nivel está compuesto por la ALU y una colección de 8 a 32 registros que forman una memoria local. Los registros se encuentran conectados a la ALU y se forma una trayectoria de datos, por donde fluyen los datos a la hora de ejecutar una instrucción. Para interpretar las instrucciones provenientes del nivel 2, se utiliza el propio hardware o un programa intérprete llamado microprograma. Así, la función principal de este nivel es obtener, interpretar y ejecutar instrucciones provenientes del nivel 2.
Nivel 2 o Nivel de Arquitectura de Conjunto de Instrucciones (ISA)	En este nivel se describe el conjunto de instrucciones básicas que la máquina es capaz de ejecutar. En otras palabras, se describen las instrucciones que el microprograma o el hardware son capaces de ejecutar.
Nivel 3 o Nivel de Sistema Operativo	Es un nivel híbrido, ya que contiene muchas instrucciones del nivel ISA, como así también nuevas instrucciones y otras características adicionales. Todas las nuevas funcionalidades de este nivel, son interpretadas por un programa intérprete en el nivel 2 llamado **sistema operativo**. Por otra parte, el microprograma o el hardware ejecuta directamente las instrucciones del nivel 3 que son idénticas a

	las del nivel 2.
Nivel 4 o Nivel de Lenguaje Ensamblador	A diferencia de los niveles anteriores, está diseñado para que sea utilizado por un programador de aplicaciones, ya que el lenguaje empleado deja de ser numérico y pasa a incorporar algunas palabras y abreviaturas entendibles para las personas. Además, las instrucciones del nivel 4 y superiores, generalmente se traducen, mientras que las de los niveles inferiores se interpretaban. El programa que realiza la traducción de las instrucciones del nivel 4 se llama **ensamblador**
Nivel 5 o Nivel de Lenguaje Orientado hacia problemas	Consta de lenguajes diseñados para ser utilizados por programadores de aplicaciones que quieran resolver problemas del usuario. Estos lenguajes se denominan **lenguajes de alto nivel** (C, C++ Java, etc.). Los programas escritos en estos lenguajes se traducen a lenguajes del nivel 3 o 4 con traductores llamados **compiladores**, aunque a veces también se interpretan. Así, este nivel proporciona herramientas a los programadores de aplicaciones para resolver problemas de una manera cómoda, sin necesidad de entender los complejos lenguajes que se utilizan en los niveles inferiores.

6. **¿Qué es la arquitectura de una computadora, y cuáles son los aspectos de los que se ocupa?**

La **arquitectura de una computadora** consiste en el conjunto de tipos de datos, operaciones y características de cada nivel. La arquitectura se ocupa de los aspectos que el usuario de ese nivel puede ver, por ejemplo, las características que ve el programador o la cantidad de memoria disponible.

7. **¿Qué contiene la CPU?**

La **CPU** (Unidad Central de Procesamiento) tiene la función de ejecutar programas almacenados en la memoria principal, buscándolas y examinándolas para luego ejecutarlas una tras otra.

La CPU está compuesta de las siguientes partes:

- *Unidad de control*: Se encarga de buscar instrucciones de la memoria principal y determinar su tipo.
- *Unidad aritmética lógica (ALU)*: Realiza las operaciones necesarias para ejecutar la instrucción. Algunas operaciones son: sumas, AND, OR, etc.
- *Registros internos*: Se trata de una memoria muy pequeña (<1KB) y de alta velocidad que sirve para almacenar resultados temporales y parámetros de

control. La memoria está compuesta por varios registros, cada uno de los cuales tiene cierta función (todos los registros internos son del mismo tamaño).

8. **Mencione los 3 grandes pasos que realiza la CPU para ejecutar una instrucción, y describa brevemente cada uno**

   Los tres grandes pasos que debe realizar la CPU para ejecutar una instrucción son:

   1. *Búsqueda*: Consiste en buscar la siguiente instrucción de la memoria principal y colocarla en el registro de instrucciones. Esta tarea es realizada por la unidad de control.
   2. *Decodificación*: Consiste en determinar el tipo de instrucción que se trajo y si la instrucción requiere alguna palabra de memoria, determinar dónde está, ir a buscarla y colocarla en un registro. Esto también lo realiza la unidad de control de la CPU.
   3. *Ejecución*: Finalmente, la ALU ejecuta la instrucción, y el resultado de la misma se coloca en el registro de salida. El resultado del registro de salida luego se guarda en otro registro, el cual posteriormente se almacena en la memoria principal si se desea.

9. **¿Cuál es la ventaja de las CPU RISC sobre CISC?**

Una **CPU RISC** emplea un conjunto de instrucciones reducido, compuesto por pocas instrucciones sencillas. Por otra parte, una **CPU CISC** emplea un conjunto de instrucciones complejo, compuesto por muchas instrucciones mucho más avanzadas que las de la CPU RISC.

Una de las principales ventajas de las CPU RISC sobre las CISC es la velocidad de ejecución de los programas, ya que si bien una computadora RISC necesitaba 4 o 5 instrucciones para hacer lo que hacía una CISC, las instrucciones de la RISC se ejecutan 10 veces más rápido ya que no se interpretan.

10. **Verdadero o Falso: Todas las instrucciones se ejecutan en 1 ciclo de CPU.**

Falso. Una instrucción, dependiendo de su complejidad, puede requerir más de un ciclo de CPU. Por otra parte, cuando la CPU realiza una búsqueda de palabras en la memoria principal, esto suele demorar varios ciclos de CPU.

11. **¿Cuál es la diferencia entre el uso de filas de procesamiento y las arquitecturas superescalares?**

El uso de **filas de procesamiento** consiste en dividir la ejecución de una tarea en varias etapas, cada una de las cuales se maneja con un componente de hardware dedicado, y estos componentes pueden operar en paralelo. A diferencia de la ejecución de instrucciones en una arquitectura superescalar, aquí la ejecución de instrucciones es **secuencial**.

En cambio, en una **arquitectura superescalar**, se utiliza una única fila de procesamiento pero muchas unidades de ejecución, por ejemplo, una para la aritmética de enteros, una para la aritmética de punto flotante, otra para las operaciones booleanas, etc. Así, tan pronto como una unidad de ejecución se encuentre libre, busca en el búfer de contención para ver si hay una instrucción que pueda manejar; de ser así, saca la instrucción del búfer y la ejecuta. Una consecuencia de este diseño es que con frecuencia las instrucciones del programa se ejecutan en forma desordenada. En gran parte, es

responsabilidad del hardware asegurarse de que el resultado producido sea el mismo que hubiera producido una implementación secuencial.

12. ¿Cuál es la diferencia entre los multiprocesadores y las multicomputadoras?

Un **multiprocesador** es un sistema con varias CPU, las cuales comparten una memoria común. Esto tiene la ventaja de que la programación de una sola memoria compartida es relativamente sencilla. Sin embargo, a medida que aumenta el número de procesadores, los multiprocesadores se vuelven extremadamente difíciles de construir, ya que se vuelve complicado conectar físicamente todos los procesadores a la memoria común.

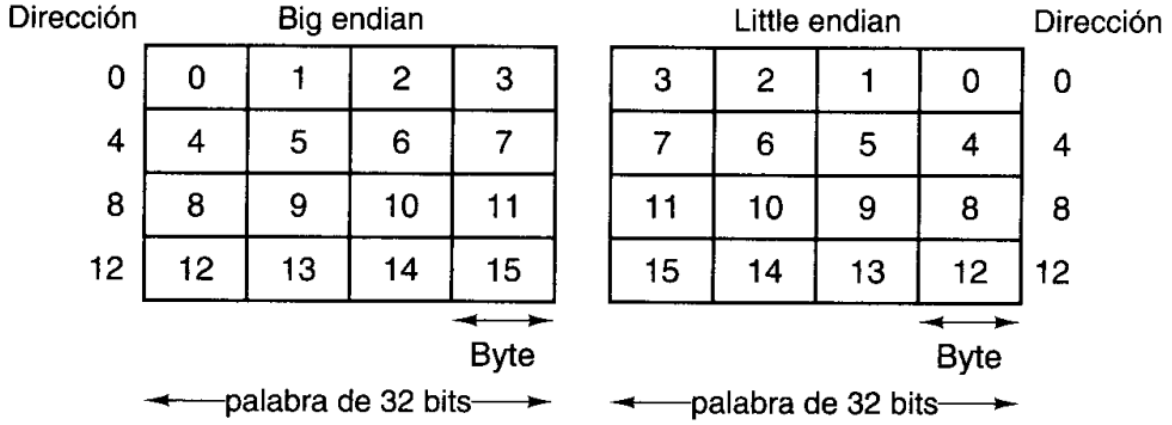
Por otra parte, una **multicomputadora** consiste en muchas CPU interconectadas, cada una de las cuales posee su propia memoria, sin que haya una memoria compartida. En este tipo de sistemas, las CPUs se comunican entre sí a través de mensajes. Las multicomputadoras son mucho más fáciles de construir que los multiprocesadores, pero su programación es más compleja.

13. Verdadero o Falso: el tamaño de la palabra de memoria es igual para todas las memorias

Falso. La mayor parte de las memorias de las computadoras utilizan palabras cuyo número de bits es un múltiplo de 8, por lo tanto, una palabra de 16 bits contiene dos bytes, y una palabra de 32 bits está formada de cuatro bytes. Sin embargo, el tamaño de las palabras de memoria no necesariamente debe ser el mismo para todas las memorias en el mercado.

14. ¿Qué diferencia hay entre el esquema little endian y el esquema big endian?

Los esquemas **little endian** y **big endian** son dos maneras diferentes de ordenar los bytes que forman una palabra, ya sea de izquierda a derecha o de derecha a izquierda. El esquema big endian comienza su numeración por el extremo de orden alto, en contraste con el esquema little endian. En la siguiente figura se muestran las diferencias en la numeración de ambos esquemas para una memoria de 32 bits.



15. ¿Cuál es la función de la memoria caché?

La **memoria caché** es una memoria mucho más pequeña y rápida que la memoria principal que sirve principalmente para mejorar la velocidad a la que se ejecutan los programas en una computadora, ya que el acceso a la memoria principal es muy lento y suele crear cuellos de botella.

Básicamente, en la memoria caché se guardan las palabras de memoria de mayor uso. Así, cuando la CPU necesita una palabra, primero la busca en la memoria caché. Sólo si la palabra no se encuentra allí, recurre a buscarla en la memoria principal. En los casos donde la palabra de memoria se encuentra en la caché, se reduce significativamente los tiempos de ejecución del programa.

16. **¿Cómo se conecta la CPU con la memoria principal y dispositivos de E/S?**

La CPU se comunica con la memoria principal y dispositivos de E/S a través de **buses**. Actualmente casi todos los dispositivos de E/S utilizan el bus PCI (Interconexión de Componentes Periféricos) aunque algunos dispositivos viejos usan el bus ISA (Arquitectura Estándar de la Industria). Por otra parte, la CPU se comunica con la memoria principal a través de un bus dedicado de alta velocidad, denominado bus de memoria.

17. **¿Para qué sirve el registro denominado "Program Counter" (Contador de Programa)?**

El registro **PC** o **Program Counter** es un registro especial el cual contiene la dirección de memoria de la siguiente instrucción a ejecutar. Una vez que dicha instrucción se obtiene de la memoria, el contador de programa se actualiza para apuntar a la siguiente.

18. **Defina y enuncie las diferencias entre compilación, ensamblado e interpretación**

La **interpretación** es una técnica que consiste en que un programa denominado **intérprete** tome como datos de entrada todas las instrucciones de un lenguaje de alto nivel (también denominado **lenguaje fuente**), las examine una por una y ejecute una serie de instrucciones en un lenguaje de menor nivel (**lenguaje objetivo**) que equivalgan a cada instrucción individual de alto nivel.

Por otra parte, hablamos de **compilación** cuando un programa llamado **compilador** traduce cada instrucción escrita en el lenguaje fuente (de alto nivel) por una sucesión de instrucciones en el lenguaje objetivo (de bajo nivel). Así, el compilador produce un programa equivalente escrito exclusivamente en el lenguaje objetivo. Luego, la computadora puede ejecutar este nuevo programa en lugar del programa original escrito en un lenguaje de alto nivel.

Por otra parte, el **ensamblado** es similar a la compilación, con la particularidad de que el lenguaje fuente es una representación simbólica de un lenguaje de máquina numérico. En este caso, el programa traductor se llama **ensamblador** y el lenguaje fuente se llama **lenguaje ensamblador**.

Las técnicas de ensamblado y compilación forman una técnica más general denominada **traducción.**

Así, en estas definiciones podemos ver que la diferencia entre interpretación y traducción es que en la traducción no se ejecuta directamente el programa original en el lenguaje fuente, sino que se genera un programa equivalente en el lenguaje objetivo. Así, en la traducción hay 2 pasos, no simultáneos, bien definidos:

1. Generación de un programa equivalente en el lenguaje objetivo.
2. Ejecución del programa generado.

Por otra parte, en la interpretación, directamente se ejecuta el programa fuente original y no es necesario generar primero un programa equivalente.

Finalmente, vemos que la diferencia principal entre el ensamblado y la compilación es la relación del lenguaje fuente/objetivo. En compilación, el lenguaje fuente es de alto nivel como Java o C y el lenguaje objetivo es un lenguaje máquina numérico o una representación simbólica del mismo, mientras que en el ensamblado, el lenguaje fuente es la representación simbólica del lenguaje máquina y el lenguaje objetivo es exclusivamente el lenguaje máquina numérico.

19. Indique el valor de los siguientes números en sistema decimal, hexadecimal y octal:

a. 01000101 00100101 11001001

Binario	Decimal	Octal	Hexadecimal
01000101	69	105	45
00100101	37	45	25
11001001	201	311	C9

b. 11010011 11000100 10001010

Binario	Decimal	Octal	Hexadecimal
11010011	211	323	D3
11000100	196	304	C4
10001010	138	212	8A

20. Indique el valor de los siguientes números en sistema binario, hexadecimal y octal

a. 7225

b. 6234

Decimal	Binario	Octal	Hexadecimal
7225	1110000111001	16071	1C39
6234	1100001011010	14132	185A

21. ¿Cómo representan las computadoras los números con punto flotante?

Las computadoras representan los números con punto flotante a través de un sistema basado en la siguiente notación científica:

$$n = f \cdot b^e$$

donde f es la **fracción** o **mantisa,** b es la **base** (que suele ser 2, 8 o 16) ye es un entero positivo o negativo, denominado **exponente**. Así, el intervalo de números que la computadora puede representar está determinado por el número de dígitos del exponente, y la precisión está determinada por el número de dígitos de la mantisa.

22. ¿Qué es el desbordamiento y el subdesbordamiento en números de punto flotante?

Los **errores de desbordamiento** y de **subdesbordamiento** se deben a la naturaleza finita en la representación de los números en las computadoras. Básicamente ocurre un error de desbordamiento cuando la magnitud del resultado de una operación es mayor que el número más grande del conjunto representable. Por otra parte, un error de subdesbordamiento se produce cuando la magnitud del resultado es menor que el número más pequeño del conjunto representable.

23. Verdadero o Falso: el error de redondeo absoluto en punto flotante es igual para números pequeños y para números grandes.

Falso. El error de redondeo absoluto en punto flotante es mucho mayor para los números grandes que para números pequeños. Esto se debe a que la distancia entre dos números representables es mucho mayor para números grandes, por lo tanto, al redondear el resultado de una operación al número representable más próximo, el error absoluto es mucho mayor para números grandes.

24. Verdadero o Falso: el error de redondeo relativo en punto flotante es menor para números pequeños y para números grandes.

Falso. El error relativo de redondeo es aproximadamente igual tanto para números pequeños como para números grandes.

Sistemas Operativos

25. ¿Cuáles son las 2 principales funciones del Sistema Operativo (SO)? Explíquelas brevemente y ejemplifique.

El **sistema operativo** es un software cuyas funciones principales son:

- Proporcionar a los programadores de aplicaciones una máquina virtual mucho más fácil de programar que la máquina real, brindando además capacidades adicionales (por ejemplo, memoria virtual, etc.).
- Gestionar todos los recursos de hardware de la computadora, realizando tareas como por ejemplo:
    - Asignación de recursos de cómputo (CPU, impresoras, discos, etc).
    - Evitar colisiones, deadlocks, etc. a causa de la concurrencia

26. ¿Qué es una *llamada al sistema*? ¿Para qué sirve? Ejemplifique.

Las **llamadas al sistema** son una serie de instrucciones “ampliadas” que proporciona el sistema operativo para que los programas de usuario las utilicen. De esta forma, sirven como un mecanismo mediante el cual las diferentes aplicaciones o programas de usuario pueden solicitar un servicio al sistema operativo.

Algunos ejemplos de llamadas al sistema son todas aquellas instrucciones relacionadas con la manipulación de archivos, que incluyen la creación, eliminación, apertura, cierre, escritura y lectura. De esta forma, si un programa necesita leer un archivo, invoca al sistema operativo para que ejecute dicha operación (en **modo kernel**). Cuando se ha completado el trabajo, el control se devuelve al programa de usuario en la instrucción que va después de la llamada al sistema.

27. ¿Qué es un proceso?

Se denomina **proceso** a un programa en ejecución. Cada proceso tiene asociado un **espacio de direcciones** donde el proceso puede leer y escribir. A su vez, el espacio de direcciones contiene el programa ejecutable, los datos del programa y su pila. También hay asociado a cada proceso un conjunto de recursos, que comúnmente incluye registros y toda la demás información necesaria para ejecutar el programa.

28. Verdadero o Falso:

a. Un proceso tiene asociado un único programa

Verdadero.

b. Un programa puede tener asociado un único proceso

Verdadero.

29. Defina

a. Directorio

Un **directorio** es un lugar en donde se pueden almacenar y agrupar los archivos. Técnicamente, el directorio almacena información acerca de los archivos que contiene: como los atributos de los archivos o dónde se encuentran físicamente en el dispositivo de almacenamiento.

b. Ruta de acceso (path)

El **path** de un archivo consiste en la lista de directorios que deben recorrerse desde el **directorio raíz** (es decir, desde el primer nivel de la jerarquía de directorios) para llegar al archivo, y se utilizan barras diagonales para separar los componentes. Es decir, el path sirve para ubicar e identificar a un archivo dentro de una jerarquía de directorios determinada.

c. Directorio de trabajo

Un **directorio de trabajo** de un proceso representa el directorio en el cual dicho proceso se encuentra trabajando en ese instante. Así, el proceso lo utiliza como referencia para las rutas relativas, en lugar del path que es una ruta absoluta.

Los procesos pueden modificar su directorio de trabajo mediante una llamada al sistema que especifique el nuevo directorio de trabajo.

30. ¿Qué son los bits rwx? ¿Para qué sirven?

Los **bits rwx** son un código de protección binario (9 bits) de archivos y directorios. Básicamente, el código de protección consta de campos de 3 bits, uno para el propietario, otro para el grupo del propietario y otro para los demás usuarios. Cada campo tiene un bit para el acceso a la lectura del archivo, otro para su escritura y otro para su ejecución. Así, estos 3 bits se conocen como bits rwx (*Read, Write, eXecute*)

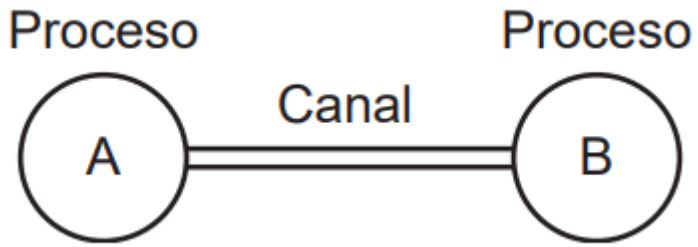
31. ¿Cuál es la diferencia entre

a. un archivo "regular"
b. un archivo "especial"
c. una tubería (pipe)

Los **archivos regulares** son los que contienen información del usuario. es simplemente es una secuencia de bytes que son almacenados en un dispositivo. Están identificados por un nombre, su tipo y el directorio que los contiene. Dependiendo del software que se ejecuta en la computadora, se puede interpretar a diferentes archivos como por ejemplo un programa, un texto o una imagen, basándose en su nombre y contenido.

Por otra parte, un **archivo especial** es una abstracción que permite que los dispositivos de E/S se vean como archivos. De esta forma se puede leer y escribir en ellos utilizando las mismas llamadas al sistema que se utilizan para leer y escribir en archivos. Por convención, los archivos especiales se mantienen en el directorio /dev.

Una **tubería** o **pipe** es un tipo de pseudoarchivo que puede utilizarse para conectar dos procesos. Si un proceso A desea enviar datos a un proceso B, escribe en el pipe como si fuera un archivo de salida. Luego, el proceso B puede leer los datos a través del canal, como si fuera un archivo de entrada.



Comunicación entre dos procesos mediante un canal o pipe

32. Describa cómo se implementa la *multiprogramación* (multiprocessing)

La **multiprogramación** consiste en la conmutación rápida de un proceso a otro. Así, en cualquier sistema de multiprogramación, la CPU conmuta de un proceso a otro con rapidez, ejecutando cada uno durante décimas o centésimas de milisegundos. Hablando en sentido estricto, en cualquier instante la CPU está ejecutando sólo un proceso, sin embargo al conmutar de un proceso a otro tan rápidamente, se logra un efecto de paralelismo (**pseudoparalelismo**).

33. Indique el estado en que se encuentra un proceso en cada caso:

a. El proceso tiene todo lo que necesita para correr, pero no es su turno de utilizar la CPU
El proceso está en estado "Listo".

b. El proceso está esperando datos por la red y no puede continuar
El proceso está en estado "Bloqueado".

c. El proceso recibió los datos de red que estaba esperando.
Si la CPU determina que es el momento de ejecutar este proceso, entonces el mismo se encuentra en estado de "Ejecución". En caso contrario, se encontrará en estado "Listo".

34. ¿Cuáles son las secciones críticas de un proceso?

Las **secciones críticas** de un proceso son aquellas partes de un programa en la que se accede a la memoria compartida. Nunca dos procesos o threads deben estar en sus respectivas secciones críticas al mismo tiempo, puesto que los resultados son impredecibles.

35. ¿En qué consiste la técnica de gestión de la memoria denominada *intercambio* (swapping)?

La técnica de gestión de memoria conocida como **intercambio** consiste en llevar cada proceso completo a memoria, ejecutarlo durante cierto tiempo y después regresarlo al disco. De esta manera, los procesos inactivos mayormente son almacenados en disco, de forma que no ocupan memoria cuando no se están ejecutando (aunque algunos de ellos se despiertan periódicamente para realizar su trabajo y después vuelven a quedar inactivos).

36. **Mencione y describa brevemente 2 técnicas de administración de la memoria.**

En términos generales, hay dos formas de llevar el registro del uso de la memoria: mapas de bits y listas enlazadas:

- *Mapas de bits*: Con un **mapa de bits**, la memoria se divide en unidades de asignación. Para cada unidad de asignación hay un bit correspondiente en el mapa de bits, que es 0 si la unidad está libre y 1 si está ocupada (o viceversa). El problema principal es que, cuando se ha decidido llevar un proceso de k unidades a la memoria, el administrador de memoria debe buscar en el mapa para encontrar una serie de k bits consecutivos con el valor 0 en el mapa de bits. El proceso de buscar en un mapa de bits una serie de cierta longitud es una operación lenta.
- *Listas enlazadas*: Consiste en mantener una lista ligada de segmentos de memoria asignados y libres. Cada entrada en la lista especifica un hueco (H) o un proceso (P), la dirección en la que inicia, la longitud y un apuntador a la siguiente entrada. Por lo general, esta lista suele estar ordenada por dirección de memoria.

37. ¿Para qué sirve la paginación?

La **paginación** es una técnica que utilizan los sistemas de memoria virtual. Básicamente sirve para que la computadora pueda ejecutar programas que sean demasiado grandes como para caber por completo en la memoria principal. La técnica consiste en que cada programa tenga su propio **espacio de direcciones virtuales**, el cual se divide en **páginas**. Estas páginas se asocian a la diversas partes de la memoria física denominadas **marcos de página**, pero no todas tienen que estar en la memoria física para poder ejecutar el programa.

Cuando el programa hace referencia a una parte de su espacio de direcciones que está en la memoria física, el hardware realiza la asociación necesaria al instante. Cuando el programa hace referencia a una parte de su espacio de direcciones que no está en la memoria física, el sistema operativo recibe una alerta para buscar la parte faltante y volver a ejecutar la instrucción que falló.

Así, la paginación permite direccionar un espacio de memoria mayor que la memoria física disponible.

38. ¿Para qué sirve la segmentación?

La **segmentación** consiste en proporcionar a la máquina muchos espacios de direcciones por completo independientes, llamados **segmentos**, con el objetivo de que porciones o tablas de memoria independientes que aumenten o reduzcan su tamaño ocasionalmente no se traslapen en las mismas direcciones de memoria.

Cada segmento consiste en una secuencia lineal de direcciones, desde 0 hasta cierto valor máximo. Los distintos segmentos pueden tener distintas longitudes, y además, las longitudes de los segmentos pueden cambiar durante la ejecución.

Debido a que cada segmento constituye un espacio de direcciones separado, los distintos segmentos pueden crecer o reducirse de manera independiente, sin afectar unos a otros. Así, una tabla de datos particular que necesite más espacio de direcciones para crecer, puedo hacerlo tranquilamente sin afectar a las demás tablas, ya que no hay nada más en su espacio de direcciones. Para especificar una dirección en esta memoria segmentada o bidimensional, el programa debe suministrar una dirección en dos partes, un número de segmento y una dirección dentro del segmento.

Por otra parte, la segmentación también permite aplicar protección mediante permisos, para que un proceso no pueda leer/escribir segmentos que no son suyos.

39. Mencione las operaciones básicas que pueden realizarse sobre los archivos

Las operaciones básicas que pueden realizarse sobre los archivos son:

1. Create
2. Delete
3. Open
4. Close
5. Read
6. Write
7. Append
8. Seek
9. Get attributes
10. Set attributes
11. Rename

40. ¿Para qué sirve la estructura de directorios?

La **estructura de directorios** sirve para organizar y agrupar los distintos archivos que se encuentren almacenados en una computadora. En la mayoría de los casos, los directorios se organizan en un **árbol de directorios**.

41. ¿Cuáles son los objetivos del software de E/S a nivel de Sistema Operativo?

Con respecto a los dispositivos de E/S, una de las funciones principales del sistema operativo es el control de todos los dispositivos de E/S que se encuentran conectados a la computadora. Adicionalmente debe proporcionar una interfaz simple y fácil de usar entre los dispositivos y el resto del sistema.

Por otra parte, el software a nivel de sistema operativo debe cumplir los siguientes objetivos:

1. **Independencia de dispositivos**: Debe ser posible escribir programas que puedan acceder a cualquier dispositivo de E/S sin tener que especificar el dispositivo por adelantado.
2. **Denominación uniforme**: El nombre de un archivo o dispositivo simplemente debe ser una cadena o un entero sin depender del dispositivo de ninguna forma. De esta forma, todos los archivos y dispositivos se direccionan de la misma forma: mediante el nombre de una ruta.
3. **Manejo de errores**: En general, los errores se deben manejar lo más cerca del hardware que sea posible. Si el controlador descubre un error de lectura, debe tratar de corregir el error por sí mismo. Si no puede, entonces el software controlador del dispositivo debe manejarlo.
4. **Simplificación del modelo de transferencia**: La mayoría de las operaciones de E/S son asíncronas: la CPU inicia la transferencia y se va a hacer algo más hasta que llega la interrupción. Sin embargo, los programas de usuario son mucho más fáciles de escribir si las operaciones de E/S son de bloqueo: después de una llamada al sistema read, el programa se suspende de manera automática hasta que haya datos disponibles en el búfer. Depende del sistema operativo hacer que las operaciones que en realidad son controladas por interrupciones parezcan de bloqueo para los programas de usuario.

42. Indique las 4 capas en las que se estructura el software de E/S en un Sistema Operativo, y mencione brevemente la función de cada una.

Los objetivos mencionados anteriormente se logran de una forma eficiente al estructurar el software de E/S en 4 capas. Cada capa tiene una función bien definida que realizar, y una interfaz bien definida para los niveles adyacentes. Las 4 capas son:

1. **Manejadores de interrupciones**: Básicamente consiste en hacer que el controlador que inicia una operación de E/S se bloquee a sí mismo hasta que se haya completado la E/S y ocurra la interrupción. Luego, cuando ocurre la interrupción, el procedimiento de interrupciones hace todo lo necesario para eliminar el bloqueo del proceso que lo inició. Así, la función principal de esta capa es desbloquear al controlador cuando se completa la E/S.
2. **Drivers de dispositivos**: Cada dispositivo de E/S conectado a una computadora necesita cierto código específico para controlarlo. Este código, conocido como

driver, es escrito por el fabricante del dispositivo y se incluye junto con el mismo. Cada driver maneja un tipo de dispositivo o, a lo más, una clase de dispositivos estrechamente relacionados. Un driver tiene varias funciones. La más obvia es aceptar peticiones abstractas de lectura y escritura del software independiente del dispositivo que está por encima de él, y ver que se lleven a cabo.

3. **Software de E/S independiente del dispositivo**: Aunque parte del software de E/S es específico para cada dispositivo, otras partes de éste son independientes de los dispositivos. La función básica del software independiente del dispositivo es realizar las funciones de E/S que son comunes para todos los dispositivos y proveer una interfaz uniforme para el software a nivel de usuario.
4. ***Software de E/S en espacio de usuario***: Aunque la mayor parte del software de E/S está dentro del sistema operativo, una pequeña porción de éste consiste en bibliotecas vinculadas entre sí con programas de usuario, e incluso programas enteros que se ejecutan desde el exterior del kernel. La biblioteca de E/S estándar contiene varios procedimientos que involucran operaciones de E/S y todos se ejecutan como parte de los programas de usuario. Las funciones principales de esta capa a nivel de usuario son hacer la llamada de E/S a través del software de usuario; aplicar formato a la E/S, etc.