1.1. Enumere y defina brevemente los cuatro elementos principales de un computador.

Al más alto nivel, un computador consta del procesador, la memoria y los componentes de E/S, incluyendo uno o más módulos de cada tipo. Estos componentes se interconectan de manera que se pueda lograr la función principal del computador, que es ejecutar programas. Por tanto, hay cuatro elementos estructurales principales:

* **Procesador.** Controla el funcionamiento del computador y realiza sus funciones de procesamiento de datos. Cuando sólo hay un procesador, se denomina usualmente **unidad central de proceso** (*Central Processing Unit*, CPU).
* **Memoria principal.** Almacena datos y programas. Esta memoria es habitualmente volátil; es decir, cuando se apaga el computador, se pierde su contenido. En contraste, el contenido de la memoria del disco se mantiene incluso cuando se apaga el computador. A la memoria principal se le denomina también *memoria real* o *memoria primaria*.
* **Módulos de E/S.** Transfieren los datos entre el computador y su entorno externo. El entorno externo está formado por diversos dispositivos, incluyendo dispositivos de memoria secundaria (por ejemplo, discos), equipos de comunicaciones y terminales.
* **Bus del sistema.** Proporciona comunicación entre los procesadores, la memoria principal y los módulos de E/S.

1.2. Defina las dos categorías principales de los registros del procesador.

Un procesador incluye un conjunto de registros que proporcionan un tipo de memoria que es más rápida y de menor capacidad que la memoria principal. Los registros del procesador sirven para dos funciones:

* **Registros visibles para el usuario.** Permiten al programador en lenguaje máquina o en ensamblador minimizar las referencias a memoria principal optimizando el uso de registros. Para lenguajes de alto nivel, un compilador que realice optimización intentará tomar decisiones inteligentes sobre qué variables se asignan a registros y cuáles a posiciones de memoria principal. Algunos lenguajes de alto nivel, tales como C, permiten al programador sugerir al compilador qué variables deberían almacenarse en registros.
* **Registros de control y estado.** Usados por el procesador para controlar su operación y por rutinas privilegiadas del sistema operativo para controlar la ejecución de programas.

No hay una clasificación nítida de los registros entre estas dos categorías. Por ejemplo, en algunas máquinas el contador de programa es visible para el usuario, pero en muchas otras no lo es. Sin embargo, para el estudio que se presenta a continuación, es conveniente utilizar estas categorías.

1.3. En términos generales, ¿cuáles son las cuatro acciones distintas que puede especificar una instrucción de máquina?

Un programa que va a ejecutarse en un procesador consta de un conjunto de instrucciones almacenado en memoria. En su forma más simple, el procesamiento de una instrucción consta de dos pasos: el procesador lee (*busca*) instrucciones de la memoria, una cada vez, y ejecuta cada una de ellas. La ejecución del programa consiste en repetir el proceso de búsqueda y ejecución de instrucciones. La ejecución de la instrucción puede involucrar varias operaciones dependiendo de la naturaleza de la misma.

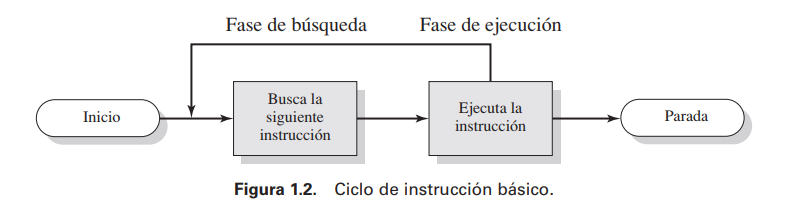
Se denomina *ciclo de instrucción* al procesamiento requerido por una única instrucción.

Al principio de cada ciclo de instrucción, el procesador lee una instrucción de la memoria. En un procesador típico, el contador del programa (PC) almacena la dirección de la siguiente instrucción que se va a leer. A menos que se le indique otra cosa, el procesador siempre incrementa el PC después de cada instrucción ejecutada, de manera que se leerá la siguiente instrucción en orden secuencial (es decir, la instrucción situada en la siguiente dirección de memoria más alta).

La instrucción leída se carga dentro de un registro del procesador conocido como registro de instrucción (IR). La instrucción contiene bits que especifican la acción que debe realizar el procesador. El procesador interpreta la instrucción y lleva a cabo la acción requerida. En general, estas acciones se dividen en cuatro categorías:

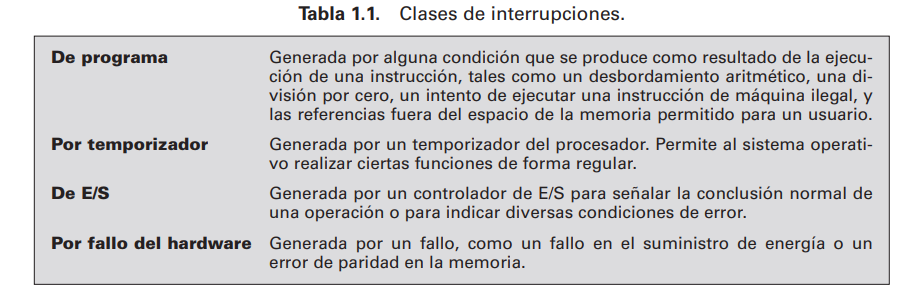
* **Procesador-memoria.** Se pueden transferir datos desde el procesador a la memoria o viceversa.
* **Procesador-E/S.** Se pueden enviar datos a un dispositivo periférico o recibirlos desde el mismo, transfiriéndolos entre el procesador y un módulo de E/S.
* **Procesamiento de datos.** El procesador puede realizar algunas operaciones aritméticas o lógicas sobre los datos.
* **Control.** Una instrucción puede especificar que se va a alterar la secuencia de ejecución. Por ejemplo, el procesador puede leer una instrucción de la posición 149, que especifica que la siguiente instrucción estará en la posición 182. El procesador almacenará en el contador del programa un valor de 182. Como consecuencia, en la siguiente fase de búsqueda, se leerá la instrucción de la posición 182 en vez de la 150.

Una ejecución de una instrucción puede involucrar una combinación de estas acciones.



1.4. ¿Qué es una interrupción?

Una interrupción es un mecanismo esencial en los computadores que permite a otros módulos, como la memoria y los dispositivos de entrada/salida (E/S), interrumpir la secuencia normal de ejecución del procesador. Este sistema se implementa para mejorar la eficiencia del procesador y optimizar su tiempo de uso. Por ejemplo, dado que la velocidad de procesamiento del procesador supera con creces la velocidad de algunos dispositivos de E/S, como las impresoras, el procesador puede quedarse inactivo durante períodos prolongados esperando a que se completen las operaciones de E/S. Las interrupciones permiten que el procesador atienda otras tareas mientras espera la finalización de estas operaciones más lentas, lo que maximiza la utilización de sus recursos y minimiza los tiempos de inactividad.



1.5. ¿Cómo se tratan múltiples interrupciones?

El estudio realizado hasta el momento ha tratado solamente el caso de que se produzca una única interrupción. Supóngase, sin embargo, que se producen múltiples interrupciones. Por ejemplo, un programa puede estar recibiendo datos de una línea de comunicación e imprimiendo resultados al mismo tiempo. La impresora generará una interrupción cada vez que completa una operación de impresión. El controlador de la línea de comunicación generará una interrupción cada vez que llega una unidad de datos. La unidad podría consistir en un único carácter o en un bloque, dependiendo de la naturaleza del protocolo de comunicaciones. En cualquier caso, es posible que se produzca una interrupción de comunicación mientras se está procesando una interrupción de la impresora.

Cuando se enfrenta con múltiples interrupciones, existen dos alternativas principales para su tratamiento:

La primera opción consiste en *inhabilitar las interrupciones* durante el procesamiento de una interrupción. Esto implica que el procesador ignora cualquier nueva señal de petición de interrupción mientras está ocupado manejando una interrupción previa. Sin embargo, esta estrategia no considera la prioridad o urgencia relativa de las interrupciones, lo que podría llevar a la pérdida de datos importantes si las operaciones de E/S no se procesan rápidamente.

La segunda estrategia implica *asignar prioridades* a las interrupciones y permitir que una interrupción de mayor prioridad interrumpa la ejecución de una de menor prioridad. Por ejemplo, en un sistema con múltiples dispositivos de E/S, cada uno tiene una prioridad asignada y se atienden en orden de prioridad. Esta estrategia garantiza que las interrupciones más urgentes se manejen primero, evitando la pérdida de datos y asegurando una respuesta rápida a eventos críticos.

1.6. ¿Qué características distinguen a los diversos elementos de una jerarquía de memoria?

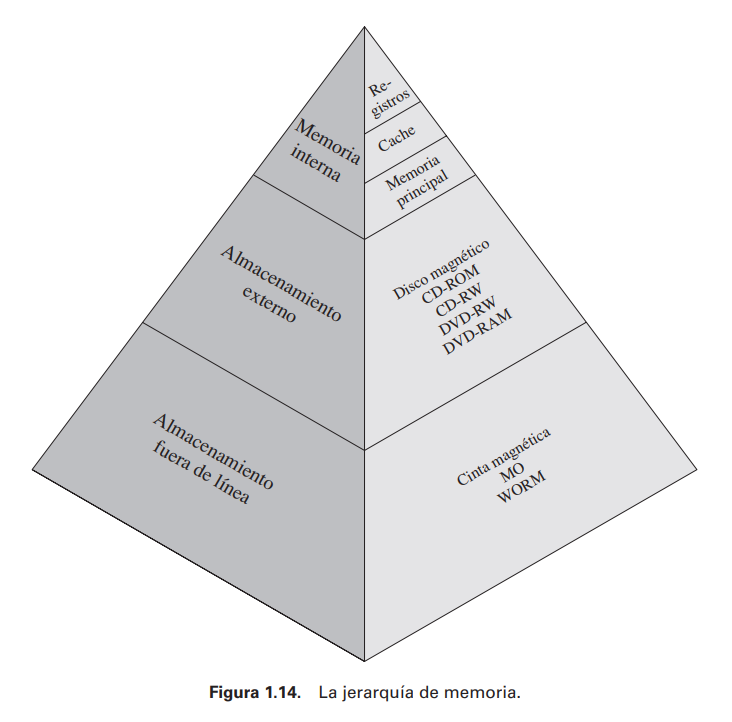
La pregunta sobre cuánta debe ser su capacidad es algo que no tiene límite. Si se dispone de una determinada capacidad, probablemente se desarrollarán aplicaciones que la usarán. La cuestión acerca de la velocidad tiene, hasta cierto tiempo, una respuesta más fácil. Para alcanzar un rendimiento máximo, la memoria debe ser capaz de mantener el ritmo del procesador. Es decir, según el procesador va ejecutando instrucciones, no debería haber pausas esperando que estén disponibles las instrucciones o los operandos. Se debe considerar también la última pregunta. Para un sistema práctico, el coste de la memoria debe ser razonable en relación con los otros componentes.

Como se podría esperar, hay un compromiso entre las tres características fundamentales de la memoria: a saber, coste, capacidad y tiempo de acceso. En cualquier momento dado, se utilizan diversas tecnologías para implementar los sistemas de memoria. En todo este espectro de tecnologías, se cumplen las siguientes relaciones:

* Cuanto menor tiempo de acceso, mayor coste por bit.
* Cuanto mayor capacidad, menor coste por bit.
* Cuanto mayor capacidad, menor velocidad de acceso.

La solución a este dilema consiste en no basarse en un único componente de memoria o en una sola tecnología, sino emplear una **jerarquía de memoria**. En la Figura 1.14 se muestra una jerarquía típica. Según se desciende en la jerarquía, ocurre lo siguiente:

1. Disminución del coste por bit.
2. Aumento de la capacidad.
3. Aumento del tiempo de acceso.
4. Disminución de la frecuencia de acceso a la memoria por parte del procesador.



1.7. ¿Qué es una memoria cache?

Una memoria caché es una memoria de acceso rápido que actúa como un intermediario entre el procesador y la memoria principal de un sistema computacional. Su objetivo es reducir el tiempo de acceso a los datos al proporcionar una ubicación de almacenamiento más cercana al procesador y de mayor velocidad que la memoria principal.

La necesidad de una memoria caché surge debido a la discrepancia en la velocidad entre el procesador y la memoria principal. A medida que la velocidad del procesador ha aumentado constantemente a lo largo de los años, la velocidad de acceso a la memoria principal ha permanecido relativamente más lenta. Para superar esta limitación, se emplea el principio de la proximidad, donde una memoria caché más pequeña y rápida contiene copias de partes críticas de la memoria principal.

Cuando el procesador necesita acceder a datos, primero verifica si están almacenados en la caché. Si se encuentran en la caché (lo que se conoce como un "acceso caché hit"), los datos se entregan al procesador rápidamente. En caso contrario, se realiza una lectura de la memoria principal y se carga un bloque de datos en la caché. Esta estrategia aprovecha la proximidad de referencias, ya que es probable que se acceda a datos cercanos en el tiempo nuevamente en el futuro.

1.8. Enumere y defina brevemente las tres técnicas para las operaciones de E/S.

Las tres técnicas para llevar a cabo operaciones de E/S son:

1. **E/S programada:** En esta técnica, el procesador es responsable de extraer los datos de la memoria principal en una operación de salida y de almacenarlos en ella en una operación de entrada. El software de E/S se escribe de manera que el procesador ejecuta instrucciones que le dan control directo sobre la operación de E/S, incluyendo la verificación del estado del dispositivo, el envío de mandatos de lectura o escritura, y la transferencia de los datos.

2. **E/S dirigida por interrupciones:** Esta técnica aborda el problema de la E/S programada, donde el procesador debe esperar inactivamente hasta que el módulo de E/S esté listo para recibir o enviar más datos. En la E/S dirigida por interrupciones, el procesador genera un mandato de E/S y continúa realizando otras tareas. El módulo de E/S interrumpe más tarde al procesador cuando está listo para intercambiar datos. Esta técnica es más eficiente que la E/S programada, ya que elimina la espera innecesaria del procesador.

3. **Acceso directo a memoria (DMA):** En el acceso directo a memoria, un módulo de DMA se encarga de la transferencia de datos entre la memoria y un dispositivo de E/S sin la intervención activa del procesador. El procesador delega la operación de E/S al módulo de DMA, que se encarga de la transferencia del bloque completo de datos, palabra por palabra, hacia o desde la memoria sin pasar por el procesador. Aunque el procesador está involucrado al principio y al final de la transferencia, durante la transferencia en sí, el procesador puede continuar con otras tareas. Aunque puede haber competencia por el uso del bus entre el módulo de DMA y el procesador, el DMA es mucho más eficiente para transferencias de E/S de múltiples palabras en comparación con las técnicas dirigidas por interrupciones o programadas.

1.9. ¿Cuál es la diferencia entre la proximidad espacial y la temporal?

La **proximidad espacial** se refiere a la tendencia de una ejecución a involucrar posiciones de memoria que están agrupadas. Esto refleja la tendencia de un procesador a acceder secuencialmente a las instrucciones. La proximidad espacial también refleja la tendencia de un programa a acceder de forma secuencial a las posiciones de datos, como cuando se procesa una tabla de datos.

La **proximidad temporal** hace referencia a la tendencia de un procesador a acceder a posiciones de memoria que se han utilizado recientemente. Por ejemplo, cuando se ejecuta un bucle, el procesador ejecuta el mismo juego de instrucciones repetidamente.

Tradicionalmente, la proximidad temporal se explota manteniendo en la memoria cache los valores de las instrucciones y los datos usados recientemente aprovechando una jerarquía de cache. La proximidad espacial se explota generalmente utilizando bloques de cache más grandes e incorporando mecanismos de lectura anticipada (se buscan elementos cuyo uso se prevé) en la lógica de control de la cache. **🡪 1.10**

**1.10.** En general, ¿cuáles son las estrategias para aprovechar la proximidad espacial y la temporal?

Para aprovechar la proximidad espacial y temporal en el diseño de sistemas de memoria y caché, se emplean diversas estrategias:

1. **Proximidad Espacial:**

- **Bloques de caché más grandes:** Al aumentar el tamaño de los bloques de caché, se incrementa la probabilidad de que se almacenen datos adyacentes en la memoria principal en la caché. Esto mejora la eficacia de la caché al explotar la proximidad espacial, ya que es más probable que se acceda a datos cercanos en la memoria principal simultáneamente.

**- Lectura anticipada (prefetching):** Se anticipa la necesidad de datos futuros y se los carga en la caché antes de que se soliciten explícitamente por el procesador. Esto se basa en patrones de acceso históricos o algoritmos predictivos para identificar datos que probablemente se necesitarán en el futuro cercano.

2. **Proximidad Temporal:**

- **Mantenimiento de valores recientes en caché**: Los datos e instrucciones utilizados recientemente se mantienen en la caché para aprovechar la proximidad temporal. Esto permite un acceso más rápido a estos datos si son requeridos nuevamente en un futuro cercano.

- **Políticas de reemplazo inteligente:** Se utilizan algoritmos de reemplazo en la caché que tienen en cuenta la frecuencia y recencia de acceso a los datos. Por ejemplo, el algoritmo LRU (Least Recently Used) reemplaza los datos menos recientes utilizados en la caché, lo que ayuda a mantener los datos más recientes en la caché para aprovechar la proximidad temporal.

En resumen, tanto la proximidad espacial como la temporal se aprovechan mediante el diseño de cachés con políticas de almacenamiento y reemplazo que optimizan el acceso a los datos. El objetivo es maximizar el rendimiento del sistema al minimizar los tiempos de acceso a la memoria y aumentar la eficiencia de la caché.