9

Memorias

El presente escrito es una versión peliminar del Capítulo 9 de Arquitectura de Computadoras de Patricia Quiroga.

Contenido

Objetivos

- Aprender las características de la tecnología de memorias de semiconductores.
- Comprender las ventajas de una organización de memoria jerarquizada.
- Describir e interpretar a nivel bloque las relaciones entre distintos chips de un módulo de memoria.
- Obtener una visión generalizada basada en los distintos tipos de clasificación de las memorias.
- Estudiar en detalle las técnicas de memoria virtual y los mecanismos de traducción de direcciones.

9.1 Introducción

Las memorias de las computadoras se utilizan para retener los bits que constituyen el alfabeto digital para representar programas y datos, o cualquier otra entidad operable para una computadora. Uno de los parámetros más importantes para medir el rendimiento es la capacidad del procesador para gestionar los accesos a sus memorias.

Imagine que usted lee este capítulo y devuelve el libro a la biblioteca. Es difícil que luego de un tiempo pueda recordar su contenido. Sin embargo, nunca olvidará cómo respirar; parece que esta actividad permanece inalterable, como en una memoria fija durante toda su vida.

Podemos indicar que los seres humanos tenemos básicamente dos tipos de memoria: Una transitoria, cuyo contenido puede variar, y otra fija, que se mantiene inalterable atendiendo actividades imprescindibles y no modificables. Con estos ejemplos sencillos se intenta una semejanza con los dos tipos básicos de almacenamiento que se encuentran en una computadora, y que constituyen lo que se denomina memoria principal o interna. Una inalterable y otra modificable. En las memorias de semiconductores el primer caso se corresponde con las tecnologías ROM y el segundo, con las tecnologías RAM.

La memoria de semiconductor almacena bits en celdas binarias constituidas por organizaciones de componentes electrónicos, como los transistores y los condensadores.

Las memorias pueden ser volátiles o perennes. Cuando se quieren procesar programas o datos intercambiables se utilizan memorias volátiles, que se pueden leer y escribir, son de tecnología de semiconductores y constituyen el área de trabajo de una computadora. Su nombre en el mercado se ha generalizado como memoria RAM. El término volátil referido a una memoria indica que la información representada en los bits almacenados se pierde cuando la computadora se apaga (o sea que las celdas binarias que almacenan bits dependen del suministro de corriente).

Los programas y los datos fijos, asociados en forma directa a la CPU, se almacenan en memorias no volátiles (o perennes) de semiconductores, que sólo se pueden leer, por lo que su contenido no se altera. Estas memorias no necesitan estar conectadas en forma permanente a la fuente de suministro para retener sus bits y su nombre en el mercado es ROM.

La RAM y la ROM son memorias a las que por lo general se accede al azar (o son de acceso aleatorio o de acceso *random*); o sea que se puede ubicar una unidad de información sin necesidad de acceder previamente a otras. A los efectos de dar un ejemplo más claro, debemos imaginar los bits almacenados en un bloque de disco (aunque estos no son memorias de semiconductores sino que representan un bit con un campo magnético); en el disco cualquier byte se deberá localizar en una zona magnetizada hasta encontrarlo (por ejemplo un sector o cluster), esto significa que no se accede al azar dentro del bloque sino en forma secuencial. Por lo tanto, el tiempo para localizar ese byte depende de su posición respecto del momento en el que éste comenzó a leer el sector.

En las memorias de acceso aleatorio la información se ubica con una dirección decodificada por un decodificador asociado a la memoria, que permite la selección de sólo una unidad de información por vez. En este caso no se requiere el acceso previo a otras unidades de información.

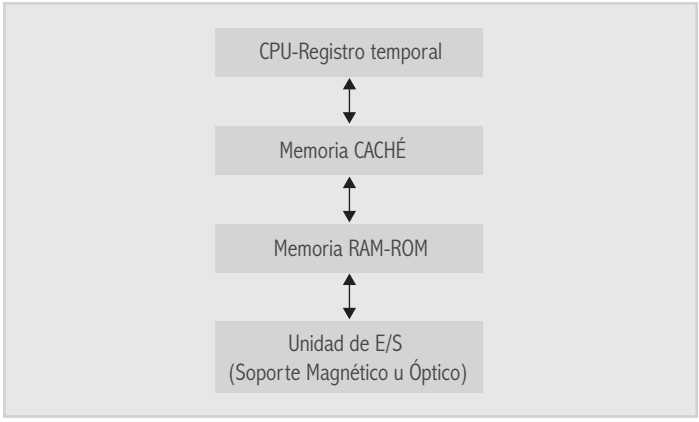
Otra manera de memorizar datos es hacerlo en forma externa, por ejemplo, en un disco. Se denominan dispositivos de almacenamiento masivo las unidades de disco –CD, DVD y otros medios– que almacenan los bits utilizando tecnología magnética u óptica, y que permiten el almacenamiento "masivo" de bits por su bajo costo. El soporte de los bits de este tipo constituye memorias perennes o no volátiles, que en su mayoría se pueden leer y escribir, y reciben el nombre de memorias externas, masivas o secundarias. Si bien es necesario que estas memorias estén asociadas a la CPU para acceder a ellas, el soporte de bits, por ejemplo, un CD, puede guardarse en un cajón sin que se pierda la información grabada en él. Podemos imaginarlas como la copia en papel de este capítulo (memoria secundaria, de masa o externa), que mientras la leemos se almacena de manera temporaria en nuestra memoria humana (memoria principal o interna).

El término **unidad de información** se utiliza para identificar un grupo de bytes a los que se accede en forma conjunta según el tipo de soporte. Para las memorias de semiconductores (RAM o ROM) también se utiliza el término **palabra** o **posición de memoria**, y para las memorias sobre medio magnético, por lo general, **bloque**.

Considerando el tiempo que involucra la actividad de "leer memoria", se denomina **tiempo de acceso** al lapso transcurrido entre la orden de lectura y la disponibilidad efectiva de la unidad de información buscada.

El **modo de acceso** es la técnica que permite la búsqueda de la unidad de información en una memoria.

El siguiente diagrama de flujo muestra la relación más común que se establece entre distintos soportes de memoria en una computadora. El ejemplo pretende establecer una jerarquía entre ellos (desde la más rápida hacia la más lenta, o desde la de menor capacidad hasta la de mayor capacidad).



El término **unidad de información** se utiliza para identificar un grupo de bytes a los que se accede en forma conjunta según el tipo de soporte.

9.2 Clasificación de memorias

Para abarcar su estudio es necesario clasificarlas según ciertos parámetros que las diferencian:

1. El modo de acceso a la unidad de información.
2. Las operaciones que aceptan por cada acceso.
3. La duración de la información en el soporte.

9.2.1 Clasificación según el modo de acceso a la unidad de información

Una memoria es de **acceso aleatorio** cuando un **componente de selección** habilita una palabra (o posición) e inhabilita a las demás. En cada acceso el componente de selección de palabra (o decodificador de direcciones) recibe "el identificador" (que es único) de la unidad de información implicada. El tiempo de acceso es independiente del lugar físico que ocupa la unidad de información en el soporte.

Una memoria es de **acceso secuencial** cuando para acceder a una unidad de información se establece una posición de referencia a partir de la cual comienza un rastreo de la unidad de información que consiste en la lectura de todas las unidades que la precedan, hasta lograr la

Una memoria es de **acceso aleatorio** cuando un **componente de selección** habilita una palabra (o posición) e inhabilita a las demás.
Una memoria es de **acceso secuencial** cuando para acceder a una unidad de información se establece una posición de referencia a partir de la cual comienza un rastreo de la unidad de información que consiste en la lectura de todas las unidades que la precedan.

buscada. En este caso el tiempo de acceso depende de la distancia entre la posición inicial y la unidad de información.

Una memoria es de **acceso asociativo** cuando la búsqueda de la unidad de información implica la comparación de un grupo de bits de la unidad de información con el contenido de una posición de memoria. Las palabras no se identifican por su dirección, y el objetivo de su lectura es verificar si este grupo de bits coincide con el contenido de alguna de ellas. Para lograr esto se envía a la memoria el grupo de bits que actúa de rótulo (en inglés, *label*), que se compara con todas las unidades de información con el fin de hallar una coincidencia. Ésta se logra si se verifica la igualdad entre los bits del rótulo y los comparados.

Memoria de **acceso asociativo**: Es cuando la búsqueda de la unidad de información implica la comparación de un grupo de bits de la unidad de información con el contenido de una posición de memoria.

9.2.2 Clasificación según las operaciones que aceptan por cada acceso

Una memoria es de **lectura/escritura** cuando admite ambas operaciones (algunos autores las denominan "**vivas**"), y es **sólo de lectura** cuando permite únicamente esta operación (algunos autores las denominan "**muertas**").

9.2.3 Clasificación según la duración de la información

Las memorias son **volátiles** cuando pierden su información con el corte de suministro de corriente, y **perennes**, **permanentes** o **no volátiles**, en caso contrario.

La clasificación también comprende una tecnología de memoria RAM denominada dinámica, que aunque sus celdas binarias degradan la información con el tiempo, el proceso de recuperación es gestionado por hardware, y, por lo tanto, la "pérdida momentánea" de información se hace transparente al entorno.

9.3 Dimensión de la memoria

Se denomina **capacidad** de memoria a la cantidad de información que se puede almacenar en ella. La capacidad se puede expresar en bits, bytes o unidades que los agrupen:

BIT

La capacidad medida en bits se utiliza para los registros, que son los soportes de información de menor capacidad. Por ejemplo, se puede decir "un registro de 16 bits" o un "registro de *64 bits*".

BYTE

De la misma manera, es aplicable a registros. Se puede decir "un registro de dos bytes" o "un registro de *8 bytes*". La capacidad de memorias mayores se expresa en unidades que agrupen bytes.

KILOBYTE (KB o K)

El KB (kilobyte), o simplemente *K*, permite definir una agrupación de *1024 bytes*.

1 KB = 2^{10} bytes = 1024 bytes.

Cuando se dice "*64 K* de memoria", significa que su capacidad de almacenamiento es de *64* por *1024*, o sea, *65536 bytes*.

MEGABYTE (MB o M, o MEGA)

Un *MB* (megabyte), o simplemente *M*, equivale a *1024 K*, esto es, un *K* por *K*, *1024* veces *1024*, o *1.048.576 bytes*. Alrededor de un millón de bytes de almacenamiento (2^{20}).

GIGABYTE (GB o GIGA)

Se refiere a *1024* mega, o sea, un *K x K x K* (2^{30}).

TERABYTE (TB o TERA)

Se refiere a *1024 giga*, o sea, un *K x K x K x K* (2^{40}).

Si bien las unidades descriptas en general se asocian con bytes, se pueden utilizar para grupos de bits o de palabra. Por ejemplo, *1 Mb/seg* significa "un megabit por segundo".

Otras unidades que se utilizan con menos frecuencia son:

Peta (P) = 2^{50}; *Exa (E)* = 2^{60}; *Zetta (Z)* = 2^{70} y *Yotta (Y)* = 2^{80}

9.4 Memorias RAM estáticas y dinámicas

Una celda binaria es un elemento que se puede mantener estable en uno de dos estados, 0 o 1.

En una memoria de semiconductores se dice que cada bit se aloja en una **celda binaria**.

9.4.1 Memorias SRAM *(static random access memory)*

Las SRAM de semiconductores son memorias vivas, volátiles y estáticas. Cada celda es un elemento biestable diseñado con compuertas, que puede modificarse sólo con un proceso de escritura, si no, permanece inalterable siempre que esté conectada a una fuente de suministro.

9.4.2 Memorias DRAM *(dynamic random access memory)*

Las DRAM son memorias vivas, volátiles y dinámicas. El término dinámica hace referencia a la característica ya descripta en cuanto a que estas memorias degradan su información con el transcurso del tiempo, aun cuando están conectadas a la fuente de suministro.

Cada celda almacena un "*1*" que se representa con la carga de un condensador. Éste conserva su carga con cierto nivel de deterioro durante un período preestablecido; por lo tanto, antes de que la información se pierda hay que restablecer la carga. Este proceso de regeneración se denomina **ciclo de refresco** (*refresh cycle*) y es un procedimiento físico que se produce a intervalos regulares. El control del acceso y el ciclo de refresco deben estar a cargo del **controlador de memoria**, cuya función es simplificar su complejidad para los dispositivos que la accedan.

Son memorias más lentas que las SRAM, pero tienen mayor capacidad. Considere que una celda binaria de tipo estático está constituida por seis transistores, luego el costo por bit de almacenamiento es mucho menor en las DRAM que en las SRAM, debido a que se gana mayor densidad de almacenamiento de bits en el mismo espacio.

9.4.3 RAM con acceso directo

Para permitir el acceso a la información en forma *random* o al azar una memoria se organiza de manera matricial en filas y columnas ($p \cdot q$). Cada unidad de información o palabra de memoria es una fila y está asociada a un componente de selección; por ejemplo, un decodificador n a 2^n, para una memoria de $2^n = p$ palabras. Cada fila o palabra de memoria o posición tiene igual cantidad de bits (q) para toda la matriz, esto es, q columnas. Las columnas están formadas por los bits de igual peso de todas las filas.

Para acceder a cada palabra de memoria, cada una de ellas se identifica con su número de fila dentro de la estructura matricial. El número que identifica la palabra en un acceso *random* o al azar se denomina **dirección física** y representa en realidad al número ordinal que le corresponde dentro de la matriz, comenzando de 0 hasta *p-1*.

Se denomina **dirección física** al número ordinal que le corresponde dentro de la matriz.

Caso ejemplo de RAM estática

Para dar una idea aproximada de su relación con el entorno se presenta un dibujo esquemático que muestra con la figura del rectángulo central un supuesto chip de *1024* palabras de *8 bits* cada una (la capacidad de almacenamiento total es de *1 KB*); las siglas asociadas representan la función y la cantidad de las líneas que accederían al chip.

La línea de alimentación VCC suministra el voltaje de régimen para el suministro de corriente y GND, su conexión a tierra.

Las líneas de datos coinciden con la cantidad de bits que se leen o escriben por cada acceso random. El sentido de estas líneas es bidireccional, lo que indica que se puede leer o escribir una palabra de *8 bits* (*D0.....D7*), o sea, "*q*" *bits* por vez.

Las líneas de dirección son diez (*A0....A9*). La combinación de diez bits permite numerar $2^{10} = 1024$ palabras; estos números son las direcciones *0* a *1023* (*0* a *p-1*). El dispositivo que acceda al chip deberá "conocer" la dirección de la palabra que se ha de acceder. Las líneas se conectan a las n entradas del componente de selección, de manera que permitan habilitar una palabra e inhibir el acceso a las demás. La línea WE indica que con un "*1*" en esta línea se da una orden de escritura (WRITE ENABLE), entonces con un "*0*" se da una orden de lectura.

La línea EN (ENABLE) indica con un "*1*" que este chip se habilitó para su acceso.

Para realizar una lectura sobre el chip se deben secuenciar los siguientes pasos:

1. Habilitar el chip y enviar la dirección de la palabra;
2. dar orden de lectura;
3. transferir el contenido de la palabra desde el chip hacia el dispositivo que solicita la información utilizando las líneas de dato.

Para realizar una escritura sobre el chip se deben secuenciar los siguientes pasos:

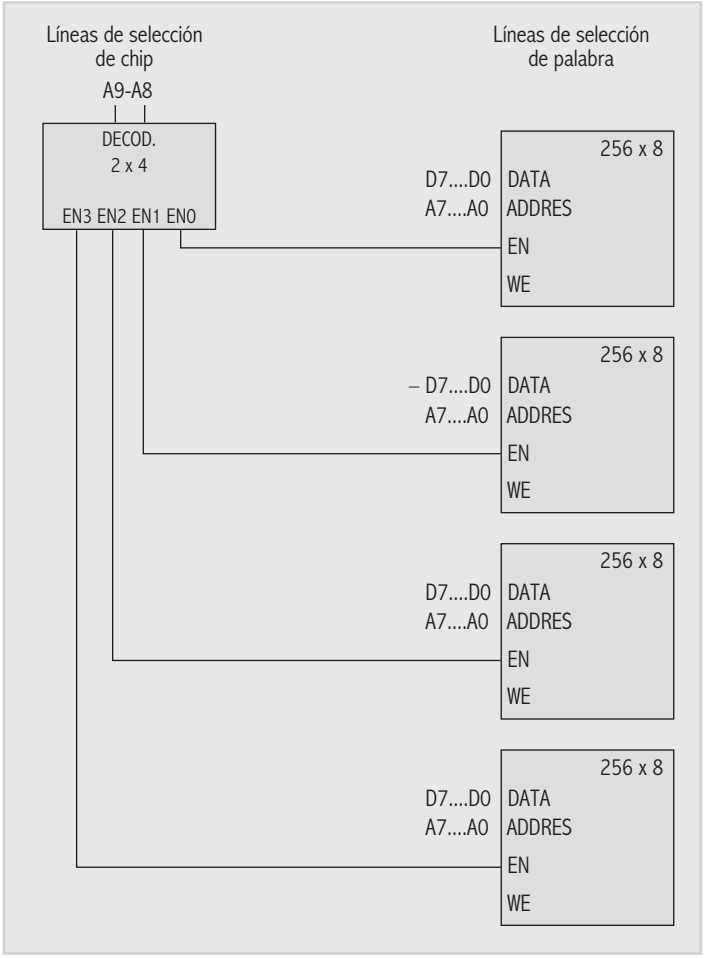
1. Habilitar el chip y enviar la dirección de la palabra;
2. transferir el contenido de la palabra desde el dispositivo que solicita el almacenamiento de la información utilizando las líneas de dato;
3. dar orden de escritura.

El esquema siguiente representa la conexión necesaria para observar la relación entre distintos chips que forman una matriz final de *1 K x 8* (idéntica capacidad de almacenamiento que en la figura anterior, pero organizado con chips de menor capacidad).

Cada palabra se accede con una dirección de *10 bits* que se transfieren a través de *10* líneas (las ocho de orden inferior son las líneas de selección de palabra y las dos de orden superior corresponden a las líneas de selección de chip). Un decodificador *2 x 4* habilita uno de los cuatro chips e inhabilita los restantes (entrada EN –*enable*–).

La señal WE habilita con 1 la escritura y con 0 la lectura. Esta línea llega a todos los chips pero sólo se accederá al activado por EN.

9.4.3.1 Diagrama de interconexión

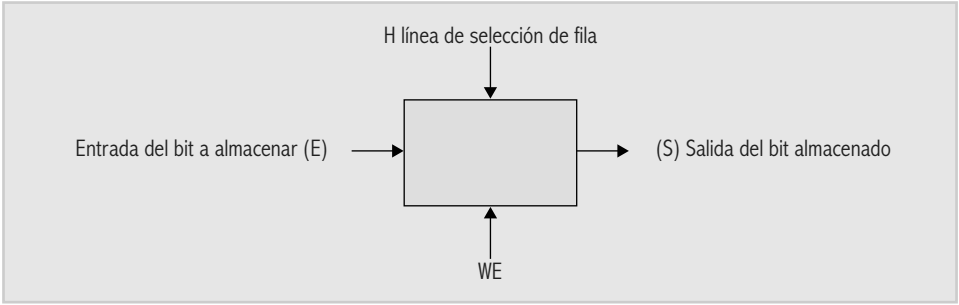


También podríamos pensar en un módulo de memoria RAM de *4096 (2^{12}) · 8*, que podría estar constituido por 4 chips separados de *1024 . 8*. En este caso todos los chips serían seleccionados por las mismas líneas de selección de chip y compartirían las mismas líneas de selección de palabra *A9* a *A0*, y dos para la selección del chip *A10* y *A11*.

Un módulo de memoria es una agrupación de chips distribuidos sobre una placa. Estos chips también suelen llamarse cápsulas o pastillas de memoria.

9.4.3.2 Biestable asociado a una matriz

El tipo de memoria de semiconductores, que utiliza biestables de varios transistores como celdas de bits (Bi,j), se denomina SRAM (*static random access memory* o memoria de acceso aleatorio de tipo estático). Como ya se mencionó, cada biestable tiene dos salidas, una para el valor normal del bit almacenado, que llamaremos *Q* y otra que es su complemento no *Q*. Este último no se representa en la matriz del ejemplo.



Considere que el biestable *Bi,j* es de tipo *R-S*. Recuerde su tabla de funcionamiento

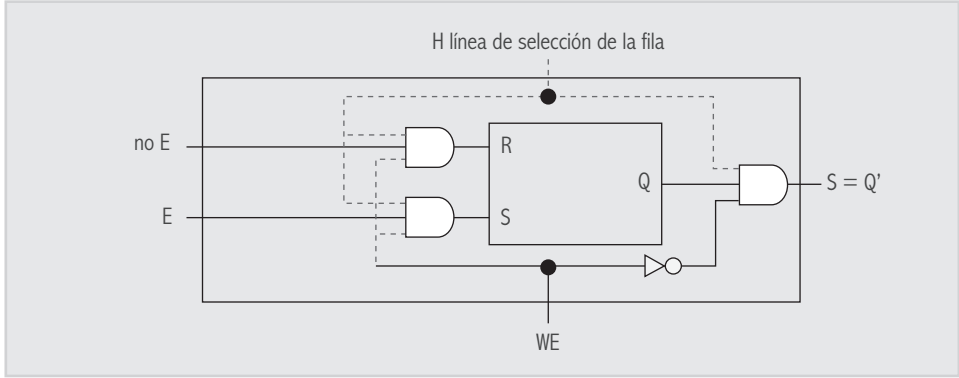
R	*S*	*Q*	*Q´*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0

No confunda *Q'* con no *Q*. *Q'* es el estado posterior de *Q*.

Una celda SRAM puede hallarse en tres estados distintos:

1. Reposo (*stand by*): Cuando no se realizan tareas de acceso al circuito;
2. Lectura (*reading*): Cuando se solicitó la información
3. Escritura (*writing*): Cuando se actualizan los contenidos.

A continuación, se muestra un diagrama más detallado de la celda. Las compuertas adicionales sirven para controlar el acceso a la celda durante las operaciones de lectura o escritura. Estas relaciones son necesarias para incluirlo dentro de la matriz de la memoria:



Una memoria estática está constituida por biestables, en cambio, en las RAM dinámicas los bits se almacenan en condensadores, o sea, un "1" lógico se almacena con el condensador cargado y un "0" lógico, cuando no hay carga. La ventaja principal es que esta celda binaria ocupa menos espacio en un circuito integrado.
La JEDEC (Asociación de Tecnología de Estado Sólido) se encarga de la estandarización de la ingeniería de semiconductores.

La conexión de selección del biestable habilita con *1* la compuerta de salida *(S)* y las compuertas de entrada (que generan *R* y *S*), e inhabilita con *0* las tres compuertas.

La conexión WE habilita con *1* la escritura y con *0* la lectura.

Cuando la operación es lectura y se seleccionó el biestable, la conexión *S* depende del valor de *Q*.

Cuando la operación es escritura y se selecciona el biestable, si la entrada *E* es igual a *1* (*no E* es igual a *0*), *setea* el biestable (*S = 1, R = 0*), y si la conexión *E* es igual a *0* (*no E* es igual a *1*), resetea el biestable (*S = 0, R = 1*).

Cuando no se selecciona el biestable, *R = 0* y *S = 0*, luego, *Q* retiene el bit almacenado en él.

9.4.4 RAM con acceso asociativo

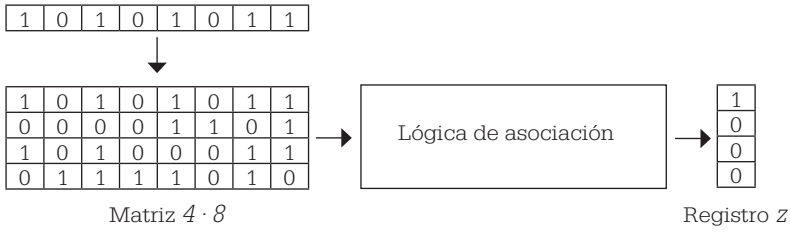
Las memorias asociativas son accesibles por contenido. El contenido buscado se denomina **rótulo**, **descriptor** o **argumento**, y puede tener la longitud en bits de la palabra que se ha de acceder.

La posibilidad de asociación requiere que todas las celdas de almacenamiento se relacionen con circuitos que permitan la comparación, razón por la que se vuelven más caras y su uso se justifica en aplicaciones en las que sea imprescindible la búsqueda por contenido.

Cada bit de la matriz (llamémoslo *bi,j*) se relaciona con un bit del registro de argumento o descriptor (*xj*), por ejemplo, por la siguiente expresión algebraica: ($\overline{xj \oplus bi,j}$ o *xj xnor bi,j*).

Para cada fila de bits de la matriz se logra un nuevo bit único sobre el bit correspondiente del registro de marcas (*zi*), que indica con un *1* la selección de la palabra, si ésta logra el apareamiento buscado.

Supongamos el contenido de una matriz asociativa ideal de *4 x 8*:



Como podemos apreciar, para acceder en modo asociativo a una matriz estática sencilla, se le debe agregar toda la circuitería necesaria que permita la representación física de las expresiones algebraicas; así se crea un sistema de memoria más complejo. La función algebraica que corresponde a *Zo* es:

$$z_0 = (x_7 \text{ xnor } b_{0,7}) \cdot (x_6 \text{ xnor } b_{0,6}) \cdot (x_5 \text{ xnor } b_{0,5}) \cdot (x_4 \text{ xnor } b_{0,4}) \cdot$$
$$(x_3 \text{ xnor } b_{0,3}) \cdot (x_2 \text{ xnor } b_{0,2}) \cdot (x_1 \text{ xnor } b_{0,1}) \cdot (x_0 \text{ xnor } b_{0,0})$$

Como ejercicio diseñe el diagrama de lógica de la función z_0, y comprenderá la complejidad de la lógica de comparación.

9.5 Jerarquía de memorias

La determinación de una jerarquía de memoria está dada básicamente por tres atributos: Velocidad de acceso, costo de la celda (bit) y capacidad de almacenamiento. Un dispositivo de

memoria ideal es aquel que permite gran capacidad de almacenamiento a bajo costo y con el menor tiempo de acceso. Cuando se elige un medio de almacenamiento, se limita al menos una de estas características, las memorias de acceso rápido suelen son de mayor costo y de menor capacidad de almacenamiento. En el caso del diseño de una computadora se toman en cuenta todas estas consideraciones. Si queremos establecer niveles de jerarquía válidos en cuanto a memorias de lectura y escritura (excluimos las memorias ROM y demás dispositivos de lógica programable a nivel físico), en los sistemas actuales podemos indicar los siguientes niveles:

- Registros
- Memoria Caché
- Memoria DRAM
- Memoria secundaria, auxiliar o externa.

El primer nivel corresponde a los registros internos, del procesador, denominados también registros de propósito general, y aquellos a los que se accede por instrucciones del sistema operativo de más alto privilegio, utilizados durante el procesamiento de instrucciones de máquina. Estos registros son de alta velocidad, y los más importantes, clasificados según su función, son: Aquellos que se utilizan como registros de almacenamiento de operandos o registros de cálculo, registros para almacenar información de control y estado del sistema, y registros para almacenar temporalmente identificadores de posiciones en la memoria principal, que en su mayoría actúan como punteros de direccionamiento.

En una memoria de semiconductores se denomina tiempo de acceso al lapso que transcurre desde el momento en que el módulo de memoria recibe una solicitud de datos hasta el instante en que esos datos están disponibles para su transferencia al lugar de destino.

En el segundo y el tercer nivel encontramos soportes para el almacenamiento temporal de instrucciones y datos intercambiables, a los que accede el microprocesador en forma directa. Estos soportes se denominan en forma genérica memorias RAM, los tiempos de respuesta DRAM suelen ser mayores que los tiempos de respuesta del procesador. En función de este parámetro se subclasifican en dos tipos, uno perteneciente al segundo nivel de la jerarquía y el otro al tercero:

- La memoria caché, que es una memoria de semiconductores, es más rápida que la DRAM de mayor complejidad y, por lo tanto, de poca capacidad, cuya velocidad de respuesta se adapta a las exigencias del microprocesador.
- La memoria DRAM, que es una memoria de semiconductores lenta, de menor complejidad y, por lo tanto, utilizada para dotar al sistema de mayor capacidad de memoria.

Entonces, podemos suponer como relación válida entre ambas memorias que, mientras que la caché puede ser mucho más rápida, es probable que sea varias veces más pequeña y más cara que la DRAM.

La memoria caché es de alta velocidad y se diseñó especialmente para proporcionar al procesador las instrucciones y los datos utilizados con mayor frecuencia.

En resumen, la memoria DRAM es de diseño más simple pero más lenta, la memoria caché tiene tecnología SRAM y, por lo tanto, presenta una mayor cantidad de componentes por celda y es mucho más rápida; esto lleva a una solución intermedia que considera las ventajas y las desventajas de cada una: Utilizar en mayor medida DRAM y guardar en una pequeña memoria SRAM copias de ella de "zonas" a las que se acceda con frecuencia.

El siguiente y último nivel corresponde a las memorias auxiliares, cuya capacidad de almacenamiento supera de manera increíble la capacidad de la memoria DRAM, pero no puede tener una relación directa con el procesador si no es a través de ella. La demora en el acceso a la información, producida en su mayor parte a causa de los movimientos mecánicos necesarios para el acceso al soporte, hacen que se desaproveche lastimosamente el tiempo de la CPU. Por

eso se gestiona la transferencia desde la memoria auxiliar hacia la memoria DRAM, para que el programa pueda ejecutarse desde esta última. El soporte de bits en las memorias auxiliares corresponde a la tecnología de medios magnéticos y ópticos.

Las memorias de disco permiten acceso directo a la información, y su capacidad se mide en gigabytes *($2^{10} \cdot 2^{10} \cdot 2^{10}$ bytes)*, terabytes *($2^{10} \cdot 2^{10} \cdot 2^{10} \cdot 2^{10}$ bytes)* y en petabytes *(2^{50} bytes)*.

9.6 Memorias caché

Las memorias caché son de tecnología de semiconductor de tipo estático (SRAM = *static* RAM), cuya velocidad de respuesta se ajusta de manera muy favorable a los tiempos del procesador. El fundamento que sustenta su inclusión en un sistema es que su velocidad es compatible con las necesidades de obtención de la información por parte del procesador. El tiempo de acceso de una lectura en memoria DRAM puede ocupar varios ciclos de reloj; esto incluye el proceso de recuperación de la memoria, el pedido, la comprobación y el tiempo de acceso de los datos, típico de esta tecnología. Si el procesador accediese directamente a DRAM, debería contemplar ciclos de espera hasta obtener el contenido de la posición direccionada.

En la memoria de etiquetas o tags se almacenan las referencias de memoria principal asociadas a cada bloque.

Para balancear costo, volumen de información y tiempo de acceso en la búsqueda de mejorar el rendimiento global, se utilizan ambas. Desde el punto de vista funcional, la caché se utiliza como memoria intermedia entre el procesador y la memoria DRAM, y almacena en forma temporal la información a la que se accede con mayor frecuencia en esta última.

¿Cómo resume usted con tres verbos lo expuesto hasta ahora?

ALMACENAR en mayor medida en la DRAM a bajo costo por bit

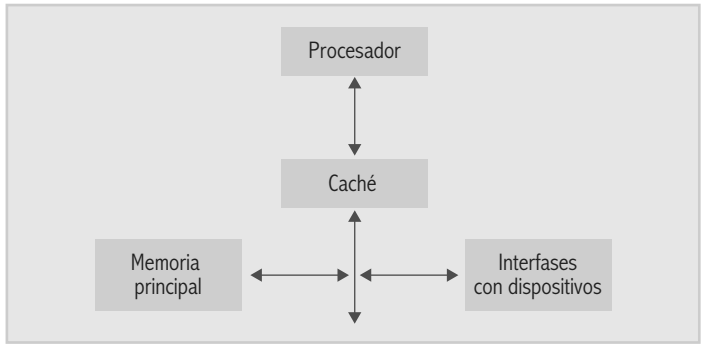
INCORPORAR una pequeña memoria caché de alta velocidad

GUARDAR copias de DRAM en caché.

En adelante, a la memoria caché la llamaremos simplemente caché y a la memoria de lectura/escritura de la memoria principal, simplemente RAM.

El esquema siguiente muestra una posible relación funcional entre ellos:

Típica conexión de caché en serie



Una caché es más que un simple área de almacenamiento, se trata de un subsistema constituido por una memoria de etiquetas, una memoria de datos y un controlador.

El "cerebro" de un sistema de memoria caché se llama controlador de memoria caché. Cuando, por ejemplo, el controlador de memoria caché trae una instrucción de la memoria principal, también trae las próximas instrucciones y las almacena en caché. Esto ocurre porque hay una alta probabilidad de que la próxima instrucción también se necesite. Esto incrementa la posibilidad de que la CPU encuentre la instrucción que precise en la memoria más rápida y, por lo tanto, pueda optimizar el tiempo.

El controlador se utiliza para gestionar su actividad. Una de sus funciones es seleccionar cuáles y cuántos bytes de memoria se copiarán en su matriz de datos. Este procedimiento no es arbitrario sino que sigue algún criterio de actualización de datos o actualización de caché, que describiremos luego. Sin importar cuál fuera en este momento, debemos indicar que el controlador "observa" el espacio de almacenamiento de la memoria RAM como un conjunto de bloques cuyo tamaño es fijo y está determinado por el controlador. Por ejemplo, una RAM de un megabyte de capacidad (*1 M*8*) puede estar dividida desde el punto de vista del controlador en bloques de *32 bytes*, entonces, está dividida en 2^{15} bloques. Para calcular este número se debe dividir el total de la memoria 2^{20} = *1 mega* entre $2^5 = 32$. Entonces, $2^{20}/2^5 = 2^{20-5} = 2^{15} = 32768$ bloques.

De todos estos bloques, sólo algunos se copiarán en la memoria de datos.

Por ejemplo, si la caché asociada a esta memoria tiene una memoria de datos de "cuatro ka" = *4 K* de capacidad, debe considerarse que ésta es una matriz de $p \cdot q$, donde p es la cantidad de líneas de la memoria de datos y q es la cantidad de bits por línea.

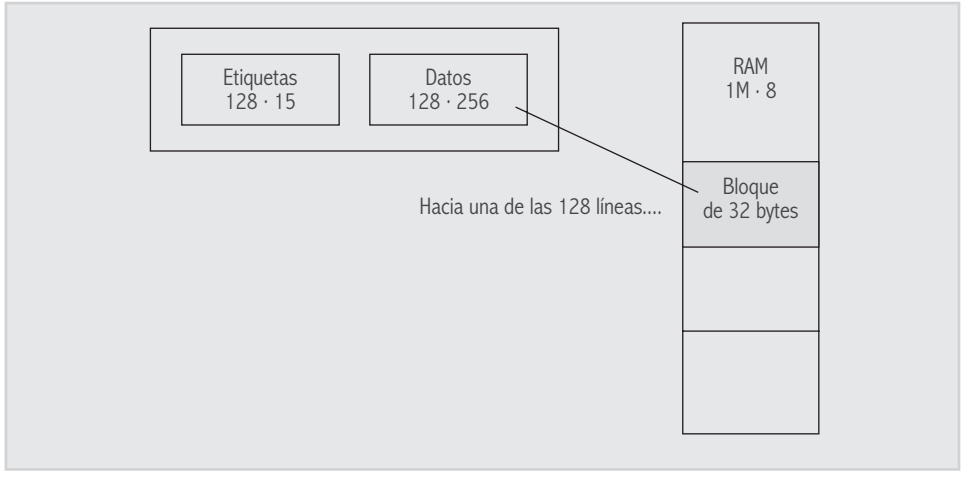
La memoria de datos aloja bloques de memoria principal que pueden contener tanto datos como instrucciones.

Si hacemos coincidir la cantidad de bytes del bloque de memoria principal con el tamaño de la línea, entonces cada línea es de *32 bytes* y, por lo tanto, $q = 256$ *bits*. Si ahora queremos calcular el total de líneas, entonces hay que dividir $2^{12} = 4$ *K* entre $2^5 = 32$. Entonces, $2^{12}/2^5 = 2^{12-5} = 2^7 = 128$ líneas, que es el valor de p. La matriz está dimensionada como de $128 \cdot 256$.

En este ejemplo no se hizo análisis alguno acerca de la relación de tamaño entre una y otra, sólo se quiere mostrar la relación numérica que indica que sólo *128* de los *32768* estarán contenidos en caché en forma simultánea.

La memoria de etiquetas es una matriz asociativa de $m \cdot n$; cada fila está asociada a una línea de la memoria de datos y se denomina etiqueta, por lo tanto, hay tantas etiquetas como líneas, esto es, *128*. El contenido de la etiqueta referencia el número de bloque de memoria principal y que se almacenó en caché; este número es de *15 bits*, porque $2^{15} = 32768$. Entonces la matriz está dimensionada como de $m \cdot m = 128 \cdot 15$.

Las tres memorias vistas por el controlador:



Se deduce que la cantidad de líneas de la caché depende de su capacidad expresada en bytes, y que esta medida es el tamaño de la memoria de datos.

Por ejemplo, si es de *8 K*, entonces $2^3 \cdot 2^{10} = 2^{13}$ entre $2^5 = 2^{13-5} = 2^8 = 256$ líneas de *256 bits* cada una *(256 · 256/8 = 8192 = 8 K)*. Por ende, ahora la memoria de etiquetas es de $256 \cdot 15$.

Cada etiqueta hace referencia a una línea con los mismos *15 bits* que determinan el número de bloque. Duplicar el tamaño de la memoria de datos, aumenta la posibilidad de encontrar la información anticipadamente en ella.

Con respecto a la memoria de etiquetas, como su aumento es de acceso asociativo, incorpora mayor complejidad y por lo tanto mayor costo.

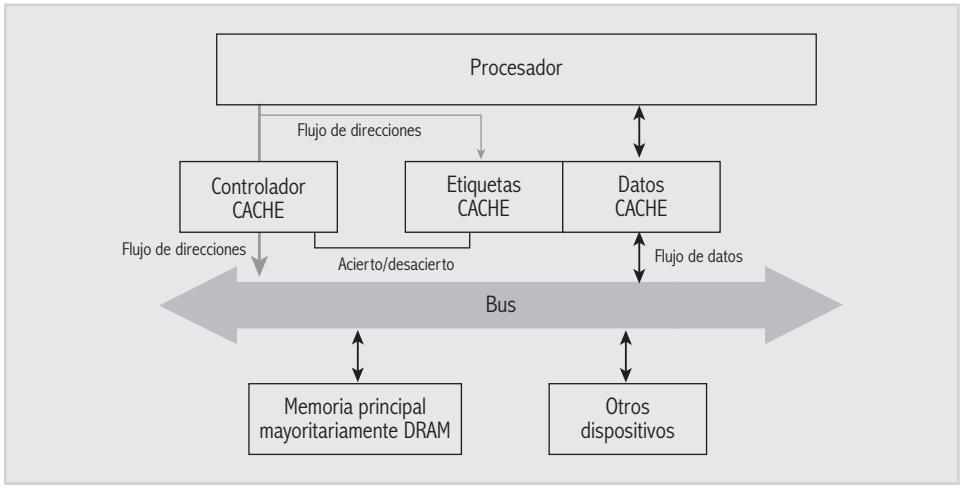
Realice el ejercicio anterior considerando:

Memoria RAM de 1 M

Memoria caché de 16 K y bloques de 8 bytes

Conexión en serie o *look-through*

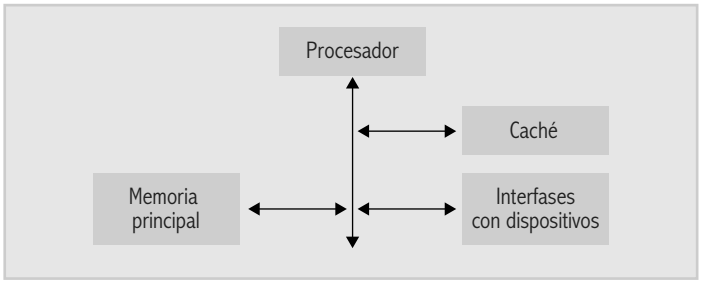
De ambos tipos de conexiones, la conexión en serie es la más utilizada en los procesadores actuales. Ilustrada en la figura que sigue, podemos afirmar que la ventaja principal se observa cuando se produce un acierto, pues el flujo de información se mantiene en la parte punteada del esquema y deja el bus de sistema en relación con su entorno, aislado y libre para su uso.



Debido a que el acceso a caché se realiza por un bus separado –esto es, primero al bus que la conecta con el procesador–, si la referencia no se encuentra, recién entonces se accede al bus de dato de memoria. De este modo, mientras el procesador está accediendo a caché, un maestro de bus asociado a un dispositivo de E/S puede acceder en forma simultánea a la RAM, y de esta manera se optimiza el uso del bus de dato que permite el acceso a memoria principal.

Conexión en paralelo o *look-aside*

Sin embargo, hay sistemas que pueden contemplar un nivel de caché externo. Por esta razón se incluye esta modalidad, al solo efecto de observar las diferencias de una respecto de la otra.



En la conexión en paralelo, todas las solicitudes de las posiciones de memoria que requiere el microprocesador llega en forma simultánea a la memoria principal y a la caché. En caso de "acierto" (la posición buscada se encuentra en caché), el controlador entrega la posición buscada al microprocesador y genera una señal que aborte el ciclo iniciado para la lectura en la memoria principal; en caso de falla, el ciclo no fue abortado y, por lo tanto, la posición se obtiene desde la memoria principal, y se entrega al microprocesador y a caché (para mantenerla actualizada).

Una ventaja de la conexión paralelo es que permite quitar o agregar el subsistema de caché (si no está integrado en otro hardware), sin necesidad de incluir modificaciones al sistema. Además, si la información buscada no se encuentra en caché, ya se está buscando de manera simultánea en la DRAM, por lo tanto, un desacierto (en inglés, *miss*) no altera el tiempo que de todos modos se iba a tardar en encontrar la información. Como desventaja podemos indicar el alto tráfico al que se somete al bus de sistema ya que, a diferencia de la conexión serie, se utiliza el mismo bus para acceder a ambas memorias.

9.6.1 Principios de funcionamiento

El procesador busca información en memoria principal cuando busca una instrucción para ejecutarla o cuando busca un dato que requiera la ejecución de una instrucción, o sea que la comunicación procesador-RAM es continua. Esta comunicación se establece por medio del bus de direcciones (que transfiere la dirección física o absoluta del primer byte al que se accede) y del bus de datos (que transfiere el contenido de uno o más bytes, según la cantidad de líneas utilizadas del bus). En cualquiera de los dos tipos de conexiones enunciadas, el controlador de caché debe "capturar" la dirección para verificar si puede ofrecer al procesador su contenido. La forma en la que se captura la dirección depende del tipo de organización; como indicamos, la memoria de etiquetas hace referencia a un número de bloque de RAM almacenado con anterioridad en una línea de la memoria de datos. Por lo tanto, para el primer ejemplo presentado (un mega de RAM y bloques de *32 bytes*) el controlador considera los *15 bits* de orden superior de la dirección física (argumento de comparación) y verifica su existencia en la memoria de etiquetas; si el argumento coincide con una etiqueta, entonces el bloque reside en caché y el byte en cuestión se identifica con los *5 bits* de dirección restantes. (Recuérdese que la dirección física para el acceso a *1 M* es de *20 bits*, puesto que $2^{20} = 2^{10} \cdot 2^{10} = K \cdot K = 1\ M$).

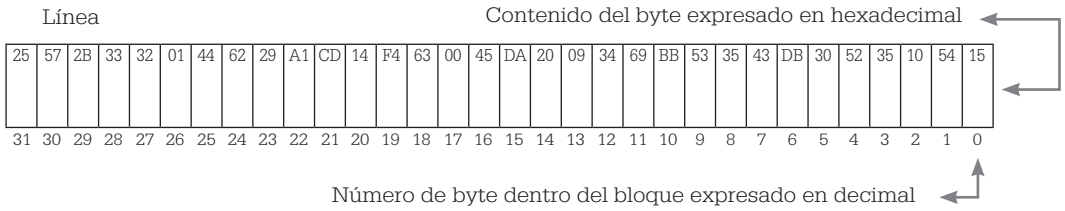
El esquema muestra la etiqueta (expresada en bits) que identifica al bloque *127* y la línea correspondiente (expresada en hexadecimal).

Si el byte en cuestión es el identificado por la dirección hexadecimal *00FC0* (denominada dirección física o absoluta) *00FC0* se pasa a binario y resulta ***0000 0000 1111 11****00 0000,* entonces los *15 bits* de órden superior resaltados en negrita muestran la etiqueta (*127* decimal) y los cinco bits restantes, el número de byte dentro del bloque (0 decimal), en este caso el primero, cuyo contenido es "*15*".

En caché

Etiqueta del bloque 127

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1



Contenido del bloque 127 en la RAM

Nro. de byte en el bloque	Dirección	Contenido
31	00FDF	25
30	00FDE	57
29	00FDD	2B
28	00FDC	33
27	00FDB	32
26	00FDA	01
25	00FD9	44
24	00FD8	62
23	00FD7	29
22	00FD6	A1
21	00FD5	CD
20	00FD4	14
19	00FD3	F4
18	00FD2	63
17	00FD1	00
16	00FD0	45
15	00FCF	DA
14	00FCE	20
13	00FCD	09
12	00FCC	34
11	00FCB	69
10	00FCA	BB
9	00FC9	53
8	00FC8	35
7	00FC7	43
6	00FC6	DB
5	00FC5	30
4	00FC4	52
3	00FC3	35
2	00FC2	10
1	00FC1	54
0	00FC0	15

9.6.2 Caching

El caching es un procedimiento que, gestionado por el controlador, anticipa las necesidades de posiciones de memoria principal de acuerdo con cierto cálculo de probabilidad de uso y utilizando criterios que consideran los principios de vecindad espacial y temporal.

Considere que los programas se ejecutan en pasos secuenciales y que las variables se alojan en zonas adyacentes, o sea que los programas requieren datos e instrucciones que se alojan en posiciones de memoria cercanas a las recientemente accedidas, esto es, que cuanto más vieja sea una referencia a dato, tanto menor será la probabilidad de que se acceda a ella en lo inmediato.

Estas situaciones se demostraron a través de la experiencia, pero para comprenderlas mejor piense en una variable de cálculo incluida en la ejecución de un bucle que se ejecuta n veces. En este caso particular la variable, se referencia varias veces (n) en un tiempo de ejecución breve, o sea, hasta que finaliza el bucle, y por lo tanto cumple con el criterio de vecindad temporal. La capacidad de la caché varía con los avances de la tecnología, se debe tener en cuenta que su rendimiento depende tanto de la efectividad de la gestión de caching como de su tamaño.

9.6.2.1 Traducción de la dirección física (organización interna)

La caché se divide en líneas o bloques de tamaño fijo. Cada bloque se asocia a una etiqueta que opera a modo de referencia utilizando parte de la dirección física de la posición buscada en la RAM; además, para cada línea se puede almacenar información de estado, por ejemplo, "línea actualizada".

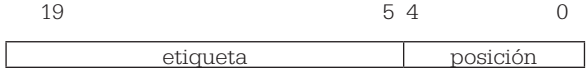
Como el tamaño de la memoria DRAM no coincide con el de la caché, sus respectivos espacios de direccionamiento son distintos, por lo tanto, una dirección física deberá ser traducida o "mapeada" por el controlador, que "adapta" la dirección y comprueba si existe tal referencia en caché. La gestión de traducción puede clasificarse según tres formas de organización del subsistema:

- Totalmente asociativa.
- Asociativa de *1* vía o de correspondencia directa.
- Asociativa de conjunto o de *n* vías.

9.6.2.2 Mapeo totalmente asociativo

La caché utiliza una memoria de tipo asociativo en los casos descriptos, donde a cada línea le corresponde una etiqueta.

A continuación se observa un esquema de la forma en que la dirección física se analiza para el ejemplo anterior.



Se considera la mejor organización, dado que no existe una relación entre el identificador del bloque y su posición dentro de la caché.

Ante un fracaso (no hay equiparamiento, no hay acierto), la palabra se obtiene de la memoria principal siguiendo dos caminos: Hacia el procesador y hacia la caché, para agregarse como "bloque al que se accedió recientemente".

El conjunto de la memoria se puede organizar como un anillo, cada vez que se agrega un bloque, tanto las etiquetas como las líneas que ya estaban se desplazan un lugar, y si la cache está completa la primera de la cola se pierde (que es precisamente la más antigua).

Por citar una desventaja, debido a la gran cantidad de etiquetas la lógica de comparación es compleja y cara.

9.6.2.3 Mapeo asociativo de una vía o de correspondencia directa

Si bien ésta es la organización interna menos utilizada en los subsistemas actuales, en este texto se detalla por su simplicidad para una primera aproximación a la organización más usada, que es la de correspondencia asociativa por conjuntos. La diferencia radica en la forma en que el controlador interpreta la dirección física, y en consecuencia ubica los bloques en caché.

La memoria RAM se divide en grupos del mismo tamaño e igual estructura que la línea de la memoria de datos en caché.

Suponga una memoria principal de *1 megabyte* y una caché de *4 K* .

La caché se puede organizar en 256 líneas o bloques de *16 bytes* cada una.

256 líneas . 16 bytes = 4096 bytes = 4 Kbytes

Los *20 bits* que permiten identificar cualquiera de las un mega posiciones serán representados en hexadecimal por las letras *XXYYZ*, de modo que para esta organización pueden asumir una nueva interpretación:

- El dígito hexadecimal menos significativo indica el número de byte de *0* a $2^4 - 1$, en decimal (*0;15*) que denominaremos *Z*;
- los dos siguientes, el número de *bloque o línea en caché*, que equivale al número sector *de RAM* desde *0* hasta $2^8 - 1$, en decimal (*0;255*), que denominaremos *YY*. Dos grupos pueden tener una identificación de sector idéntica, por ejemplo, *7B***44***8* y *6C***44***7* son distintos grupos de memoria RAM de idéntico sector y no podrían estar en caché en forma simultánea, ya que les correspondería la única línea identificada como **44**,
- y el resto de la dirección, que denominaremos *XX*, se considera etiqueta, como en la organización anterior.

De este modo, hay una etiqueta por número *sector RAM o línea caché*, o sea, 256 etiquetas.

Ya que la futura ubicación del bloque está predeterminada por el valor *YY*, para acceder a la CACHÉ se consideran *YYZ* como su dirección de acceso, en este ejemplo *12 bits* ($2^{12} = 4$ *K*), que oscila entre *000H* y *FFFH*, en decimal (*0;4095*) (acceso random a la memoria de datos). La información buscada se encuentra en caché, si para *YYZ* la etiqueta asociada coincide con *XX*.

En esta organización se puede prescindir de una memoria de carácter asociativo para la memoria de etiquetas, ya que sólo se deben comparar los bits *XX* de la nueva dirección con la única etiqueta asociada a la línea. Esto reduce la complejidad y el costo, además de hacer más rápido el acceso. No requiere algoritmo de sustitución, puesto que se conoce el emplazamiento de cada sector en caché (línea *YY*), es por esto que indicamos que dos bloques o grupos de RAM con idénticos *YY* no pueden estar en caché en forma simultánea.

Inténtelo con una RAM de *1 K* y una caché de 8 líneas o bloques de *8 bytes* cada una, esto es, de *64 bytes* de memoria de datos

1 K = 2^{10} dividido en 8 líneas $(8 = 2^3) = \frac{2^{10}}{2^3} = 2^{10-3} = 2^7 = 128$ grupos.

La etiqueta sólo necesita los *4 bits* de orden superior, pues los *6 bits* restantes se utilizan de la siguiente manera: *3* para identificar el número de la línea y *3* para identificar el byte dentro del bloque.

Ejercicio 1:

Un procesador asociado a una caché de *8* líneas con *8 bytes* cada una y a una RAM de *1 K* realiza dos accesos de lectura sobre las direcciones *1F2* y *3CD* (considere sólo los *10 bits* de orden inferior cuando pase las cadenas hexadecimales a binario puesto que en el pasaje directo se generan *12*). ¿Qué bloque o bloques de RAM resultarán sustituidos? Responda con el número hexadecimal que identifica al bloque o los bloques, o NINGUNO si no existe sustitución.

	Etiquetas	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
0	1001	A4	54	79	32	45	22	F0	56
1	0111	14	52	33	8D	B5	34	45	32
2	1001	00	FF	64	31	11	A6	33	24
3	0011	32	63	CC	C3	FA	1F	33	53
4	1010	76	88	64	46	25	37	F3	FA
5	0100	DC	14	33	96	8A	7B	34	F0
6	**0010**	15	37	A1	85	AA	B6	42	13
7	1001	77	76	34	90	00	15	61	24

Primero se convierte la dirección hexadecimal a binario y se la mapea así:

9 8 7 6	5 4 3	2 1 0
Etiqueta	Línea	Byte

1F2 = 0001 1111 0010 = 01 1111 0010 = 0111 110 010

Por lo tanto, se busca en la línea *110*, o sea, *6*, la etiqueta *0111*. Como no hay coincidencia, se buscará en RAM y se sustituirá el bloque previamente alojado en caché en esa línea, cuya etiqueta es *0010*.

Como $1\ K = 2^{10}$, se divide entre 2^3, entonces $2^{10-3} = 2^7 = 128$ bloques. El identificador del bloque está constituido por los bits de la etiqueta y la línea *0010*, y *110* se constituye en un número de *7 bits = 0010110 = 16* hexadecimal o *22* en decimal.

Pruebe usted con la otra dirección.

Ejercicio 2:

Indique en hexadecimal la dirección que permite acceder con éxito a la línea identificada con el nº "3" y referencie al byte cuyo contenido es "32".

Respuesta: 0011011111 = 0DF

9.6.2.4 Mapeo asociativo de n vías o de n conjuntos

Es similar al mapeo directo, pero cada línea admite *n* etiquetas y n matrices de datos, esto implica una caché n veces más grande y menor posibilidad de fracaso. Se indica que las *n* etiquetas y las n matrices de datos de una misma línea constituyen un conjunto.

Si ejemplificamos con *n = 2*, entonces el apareamiento se produce primero a la línea y luego a la etiqueta.

Vía o conjunto 0

	Etiquetas	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
0	1001	A4	54	79	32	45	22	F0	56
1	0111	14	52	33	8D	B5	34	45	32
2	1001	00	FF	64	31	11	A6	33	24
3	0011	32	63	CC	C3	FA	1F	33	53
4	1010	76	88	64	46	25	37	F3	FA
5	0100	DC	14	33	96	8A	7B	34	F0
6	0010	15	37	A1	85	AA	B6	42	13
7	1001	77	76	34	90	00	15	61	24

Vía o conjunto 1

	Etiquetas	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
0	1011	BB	45	95	69	56	11	81	63
1	0100	35	54	A7	B3	B5	00	31	66
2	0000	00	00	00	00	00	00	00	00
3	0010	24	35	CD	B2	F1	31	23	32
4	0110	36	99	75	64	31	33	DD	00
5	1111	00	00	00	00	00	00	00	00
6	1010	25	36	FF	45	44	32	12	99
7	1101	88	84	01	50	34	64	BB	C1

Para facilitar su comprensión, en la primera vía o vía *0* se utilizó la misma matriz que en el ejemplo anterior. Como se puede observar, la segunda vía o vía *1* tiene idéntica estructura. En esta caché cada línea está asociada a dos etiquetas, por ejemplo, la línea *0* contiene dos bloques de RAM, o sea que se almacena el doble de información, a saber:

0	1001	A4	54	79	32	45	22	F0	56		1011	BB	45	95	69	56	11	81	63

La dirección del primero se armaría así: *1001 000 xxx*; donde el *1001* es la etiqueta, el *000* es el identificador de la línea, y *xxx*, el número de byte o posición en el bloque.

La dirección del segundo se armaría así: *1011 000 xxx*; donde el *1011* es la etiqueta, el *000* es el identificador de la línea, y *xxx*, el número de byte o posición en el bloque.

Entonces, ambos pertenecen al mismo sector RAM *000*. Si queremos calcular cuál es el identificador de grupo o bloque en RAM, debemos pensar que si el KB se dividió en grupos de *8 bytes* y en el ejercicio anterior ya calculamos que entonces hay 128 grupos, su identificador lo constituyen los *7 bits* de orden superior de la dirección física total de *10 bits* (esto es, sin los *xxx*), por lo tanto, el primer bloque es el *1001000* binario o *48* en hexadecimal, y el segundo es el *1011000* en binario o *58* en hexadecimal.

La asignación de una política de reemplazo afecta el rendimiento del subsistema de la misma forma que lo afecta su organización.

9.6.3 Actualización de caché

9.6.3.1 Políticas de sustitución

El momento de actualizar una caché es cuando se detecta una falla o ausencia de la palabra buscada. Esto es, una palabra no presente está en la memoria principal y desde allí el controlador debe "interceptarla" en su camino al microprocesador vía el bus de datos; una vez escrita la memoria de datos y la de etiquetas quedan actualizadas con un acceso reciente, según el criterio ya explicado. Si aun no se completó la caché la escritura se realiza sin ningún tipo de sustitución; no obstante, por lo general ya se encuentran almacenados datos válidos con la consecuente necesidad de que el controlador asuma un criterio coherente para efectivizar el reemplazo.

En el caso de las organizaciones sectorizadas no hay otro criterio que el de reemplazar la única posición posible dentro de la caché (recuerde que por cada sector o línea se establece la relación con una sola etiqueta). Sin embargo, en las cachés con más de un nivel de asociatividad se utiliza una política de reemplazo o política de sustitución.

La asignación de una política de reemplazo afecta el rendimiento del subsistema de la misma forma que lo afecta su organización. El controlador ejecuta un algoritmo fijo y predeterminado, que se puede clasificar en por lo menos tres categorías:

1. *Least Recently Used* (LRU), la de uso menos reciente.
2. *First In Fist Out* (FIFO), la primera en entrar es la primera en salir.
3. *Random* (RND), aleatorio.

LRU

Sustituye la información que hace más tiempo que fue referenciada (*least recently used* significa la de uso menos reciente). Entonces, se debe considerar que parte de la información almacenada asociada a cada línea se utiliza para mantener atributos; uno de esos atributos es un grupo de bits que represente esta característica, y que el controlador utiliza para identificar la que lleve más tiempo sin que se haya accedido a ella, que no es necesariamente la más antigua.

FIFO

En este caso las posiciones pueden desplazarse en el sentido de una cola, de modo que la primera que ingresó es la que primero sale, lo que es fácil de aplicar en una caché de tipo asociativo.

RANDOM

En este caso el reemplazo se realiza al azar sobre cualquier línea. Como ventaja podemos interpretar que la aplicación del algoritmo es sencilla y, por lo tanto, rápida, pero su gestión puede caer en la contradicción de sustituir una, a la que se accedió en forma reciente.

9.6.4 Actualización de la memoria principal

Una modificación en cache implica la actualización de memoria; cuando un sistema cuenta con un subsistema caché, las modificaciones a la información que gestiona el procesador se realizan en primer lugar sobre la caché. Esto es así porque se supone que es muy probable que se vuelva a acceder a aquello a lo que se accedió recientemente, según el criterio de vecindad temporal. Tome como ejemplo ilustrativo el caso de una variable incluida en varios términos de un cálculo complejo o incluida en un bucle.

Obviamente, debemos pensar que dos copias de la misma información en soportes distintos deben tener el mismo contenido. Si utilizamos este criterio, entonces por cada actualización de la caché se actualiza la memoria principal. Sin embargo, en cierto sentido este procedimiento no es muy eficiente, la escritura en caché es rápida pero la que se realiza en la memoria es lenta. Por esta razón, también existen políticas de actualización de la memoria principal que se pueden clasificar en, por lo menos, las categorías siguientes:

- Escritura inmediata, *write through*.
- Escritura obligada, *write back*.

Escritura inmediata

Este método es simple y consiste en actualizar de manera simultánea ambas memorias, de modo tal que no se genere incongruencia entre la información almacenada en ambos niveles. Como ya explicamos, esto reduce el rendimiento debido al tiempo empleado en la escritura sobre la memoria principal, con la consecuencia adicional de mantener el bus de dato ocupado por más tiempo. Sin embargo, son los subsistemas más económicos.

Escritura obligada o *write back*

Este método admite sólo las actualizaciones estrictamente necesarias en memoria principal, o sea que cierta información alojada en caché puede actualizarse varias veces antes de ser efectivamente actualizada en memoria principal. Esto es razonable de acuerdo con el criterio de vecindad temporal que indica que, por ejemplo, una variable de cálculo puede sufrir varias actualizaciones durante el procesamiento sin que se tomen en cuenta sus resultados parciales; ése es el caso de una sumatoria de 1 a n de una misma variable. Sin embargo, el método genera incongruencia entre la información almacenada en ambos soportes. Esto puede ser grave si no se actualiza la memoria principal cuando se permite que cualquier otro dispositivo tenga acceso a ella sin intervención directa por parte del microprocesador, que sólo se encarga de otorgar el permiso, o cuando el sistema es multiprocesador (y además varios procesadores comparten la misma memoria principal) y alguno de ellos quiere acceder a esa zona de la memoria. Para evitar un acceso a información obsoleta, antes de que el microprocesador (actual maestro o

"propietario" del bus) conceda el acceso a otro dispositivo, se habilita al controlador de caché para que copie las líneas marcadas como modificadas en la memoria principal.

Otra situación de "escritura obligada" se observa en el caso de un reemplazo. Para reconocer que se actualizó cierta información, el controlador recurre al atributo que podemos identificar como "modificado/no modificado"; en el primer caso se actualiza la memoria principal y luego se realiza la sustitución, y en el segundo se efectúa el reemplazo directamente.

9.6.5 Niveles de caché

En las computadoras actuales es común la utilización de varios niveles de caché. La denominada caché de nivel 1 es de tamaño reducido y se ubica funcionalmente interceptando los datos que entran en el microprocesador desde la memoria, o los que salen ya procesados hacia la memoria.

La caché de nivel 2 es de tamaño mayor y se ubica (si se incluyó en el sistema) entre la caché de primer nivel y la memoria principal. Su carácter optativo nos obliga a considerar cuáles son los beneficios de su incorporación. En primer lugar, podemos indicar que atiende pedidos menos frecuentes que la primaria, ya que normalmente el éxito en el acceso se logrará accediendo al primer nivel; esto significa que se utiliza como ampliación de la anterior. Dada su mayor capacidad de almacenamiento, no sólo contendrá la misma información almacenada en la primaria sino unos cuantos Kbytes más.

Las ventajas de contar con varios niveles se empiezan a notar cuando las aplicaciones son muy grandes o requieren acceso frecuente a datos en la memoria principal. En la medida en que la eficiencia de un nivel secundario, esto es, en la medida en que se acceda a él con mayor frecuencia por no haber encontrado la información en la primaria, se reducirá el número de ciclos de bus utilizados en el acceso a RAM, que resulta en un mejor rendimiento global del sistema. Por otra parte, si la computadora es un servidor en una red de computadoras, el acceso a su memoria principal es elevado, así como el acceso a sus dispositivos de E/S; esto produce un alto tráfico en el bus, que se beneficia con la implementación de estos niveles de memoria.

En el caso de sistemas multiprocesadores su utilización se justifica en gran medida, ya que en estos sistemas muchos comparten la misma memoria, por lo tanto, el acceso frecuente al bus puede llegar a condiciones de saturación.

9.7 Memoria principal

El esquema de memoria presentado en el capítulo 6, "Diseño de una computadora digital", a la que se denominó *X*, tiene una organización física como un espacio lineal de direcciones. En X la instrucción hace referencia a un dato en la memoria con el contenido del campo DATA, de *12 bits*, que permite "direccionar" posiciones en el rango $(0; 2^{12} - 1) = (0; 4095)$ (congruente con la cantidad de posiciones de memoria, o sea, *4096*). En *X* no se incluyó un subsistema caché, por lo tanto, no se realiza ningún procedimiento de mapeo de la dirección física.

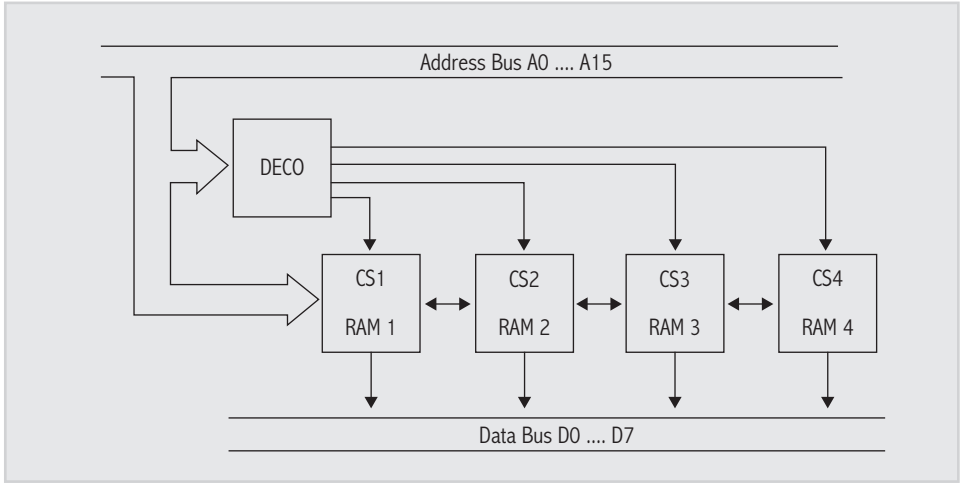
Esta dirección se transfiere a la memoria vía el bus de direcciones (en inglés, *address bus*), también de *12 bits*, y se denomina dirección física, dado que permite el acceso a la posición física sin mapeo previo.

Por ejemplo, un procesador, asociado a un bus de direcciones de *16 bits,* $(0; 2^{16} - 1) = (0; 65535)$, sólo puede disponer de una memoria organizada de manera lineal en *64 K* posiciones (si en cada posición se almacenan 8 bits, entonces se indica que puede direccionar *64 Kbytes*).

9.7.1 Memoria a nivel lógica digital

Aquí vemos un esquema posible para la organización en chips RAM. En él cada una de las líneas del bus de direcciones (en inglés, *address bus*), se identifica con las siglas *Ai*. Como la capacidad de almacenamiento está distribuida en cuatro chips, las dos líneas de orden superior *A15* y *A14* son las que identifican el número de chip: "*RAM 1*" con *00*, "*RAM 2*" con *01*, "*RAM 3*" con *10* y "*RAM 4*" con *11*. Estas líneas ingresan en un decodificador *n a* 2^n que habilita la entrada *EN*, selección de chip (o en inglés, *chip select*) que corresponda. Las líneas *A13* hasta *A0*, *14* líneas en total, seleccionan una posición de 2^{14} posibles, y como 2^{14} es igual a *16384*, ésta es la cantidad de direcciones en cada chip. Por cada posición se almacena un byte, que en el caso de una lectura será transferido por las líneas *D7 … D0*, en total *8* líneas. Si a cada dirección se la asocia a un byte, entonces la capacidad de almacenamiento será de *16384 bytes* o *16 Kbytes*. Como en total hay cuatro chips, entonces la memoria RAM es de *64 Kbytes*. Note que *64 K* es igual a $2^6 \cdot 2^{10}$; como potencias de igual base se suman, se requieren *16* líneas para direccionar toda la memoria, que es precisamente la cantidad de líneas del bus *A15-A0*.

Esquema de la distribución de 4 chips de memoria:



Si un procesador está asociado a un bus de direcciones de *32 bits*, puede acceder potencialmente a *4 giga* posiciones (*0; 2^{32} - 1*). Esto significa que la cantidad de bits de la dirección determina el espacio de direccionamiento máximo al que se puede acceder en forma directa.

Se debe considerar que no siempre es conveniente que en capacidades de almacenamiento muy grandes se utilice el modelo lineal de direcciones presentado para *X*. Ésta será una de las tantas formas en las que se pueda acceder a memoria, pero no la única.

La administración del espacio en memoria es responsabilidad del sistema operativo, por ello, más adelante se explican los modelos lógicos para acceder a ella del modo que sea más eficiente.

Considere la memoria principal o interna como el área de trabajo del procesador, donde residen los elementos de un programa en estado de ejecución (el código del programa, los datos para procesar, los resultados parciales o finales, la pila, su bloque de control, etc.).

En relación con la velocidad de respuesta de las memorias esperadas por el procesador, el número de *GHz* es un parámetro que se ha de considerar en relación con la velocidad del procesador, en la medida en que cuando aumenta también se incrementa la necesidad de memorias más veloces asociadas con él.

La memoria principal está constituida por diversas tecnologías de semiconductores y sus fabricantes utilizan la norma JEDEC que establece un estándar para los fabricantes de semiconductores.

Una de las necesidades más importantes del procesamiento automático de información es aumentar de manera constante la capacidad de memoria. Si bien la tecnología permite cada año incluir en cada chip una mayor cantidad de celdas de bit, los requerimientos de memoria del software (cada vez más sofisticado) también aumentan.

9.7.2 Memorias RAM dinámicas

Las memorias dinámicas (DRAM) son de lectura y escritura y se utilizan para almacenar una mayor cantidad de bytes en la memoria principal de las computadoras. Esto se debe sobre todo a su bajo costo y a su gran capacidad de almacenamiento en relación con las memorias estáticas (cada celda binaria permite el almacenamiento de un bit con una menor cantidad de elementos). Como desventaja podemos señalar que son más lentas que las de tecnología estática.

En un chip de memoria el soporte del bit será un par condensador/transistor, que se replica tantas veces como bits se puedan almacenar en el chip. Como conclusión podemos expresar que la decisión de incluir memorias dinámicas en el sistema se debe a que utilizan menos componentes, lo que posibilita el almacenamiento de mayor cantidad de bits en un mismo espacio. Puesto que cada vez se necesita incorporar una capacidad de almacenamiento superior, las memorias dinámicas permiten aumentar la capacidad sin incrementar su costo global.

La velocidad de respuesta de una memoria depende de la tecnología aplicada en la fabricación del chip y de cómo se resuelva el control del tráfico entre el procesador y la memoria, a cargo del controlador de memoria.

Las memorias dinámicas pueden considerarse menos apropiadas para responder con eficiencia a las exigencias de un procesador, si sólo se considera como parámetro el tiempo de respuesta; por esta razón, la mayoría de los sistemas incluye uno o varios niveles de memoria caché, ubicados entre la CPU y la DRAM.

Para facilitar su comprensión, piense en una celda estática como en un recipiente de vidrio que nunca pierde su contenido de líquido, relaciónelo con una celda dinámica pensando que esta última es un filtro de papel con forma de recipiente. De este modo podemos deducir que para regenerar el contenido de la celda habrá que verter una cuota de "líquido" a intervalos regulares de modo que se mantenga siempre un mismo nivel.

Este procedimiento de "llenado" (por supuesto ahora olvide los términos líquido y recipiente) es físico y se denomina refresco (en inglés, *refresh*).

El refresco de una memoria dinámica consiste en recargar los condensadores que tienen almacenado un 1 para evitar que la información se pierda a causa de las fugas en los condensadores.

9.7.2.1 Controlador de memoria dinámica

En este tipo de memoria se "multiplexan" las direcciones, de manera que para acceder a una celda se considera una parte de la dirección física (la de orden superior) para acceder a la fila y la otra parte para acceder a la columna en la matriz.

Como ambas direcciones se presentan al chip de memoria sobre el mismo bus de direcciones, se deben activar dos señales, una que habilite la selección de la fila y la otra que habilite la selección de la columna; estas dos señales se denominan genéricamente *RAS* y *CAS*.

RAS (*row access strobe*) es la señal que habilita la selección de una fila, y CAS (*column access strobe*) es la que habilita la selección de una columna.

Debido a ciertas condiciones para considerar en el acceso al chip de las memorias dinámicas, se administran por un dispositivo hardware llamado "controlador de memoria dinámica", cuya función es esconder al entorno, por ejemplo, al procesador, tanto los detalles del acceso como los del mantenimiento de los bits en los chips. El controlador de memoria genera todas las señales de control (p. ej., RAS y CAS) y las de tiempo que se necesitan para la actividad en memoria; además, posibilita su inicialización o período de arranque en el que se realizan escrituras sobre todas las celdas, de modo de cargar cada una de ellas a su capacidad máxima.

La función básica del controlador de memoria dinámica es establecer el flujo de información entre la memoria y la CPU.

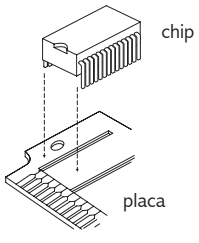
Uno de los parámetros que se ha de evaluar es la latencia CAS (CAS *Latency*), que indica, expresado de un modo simple, el número de ciclos de reloj que transcurren desde que se realiza la demanda de datos hasta que éstos se ponen a disposición del bus de datos.

Para el acceso a la celda DRAM, el controlador debe activar primero la señal RAS. Esta señal habilita la decodificación de la dirección asumida entonces como dirección de fila. Luego activará la señal CAS habilitando la decodificación de los nuevos valores sobre el bus para que se los considere dirección de columna. El tiempo que permanezca activa la señal CAS será el utilizado como parámetro para indicar el tiempo de acceso (CAS *Latency*).

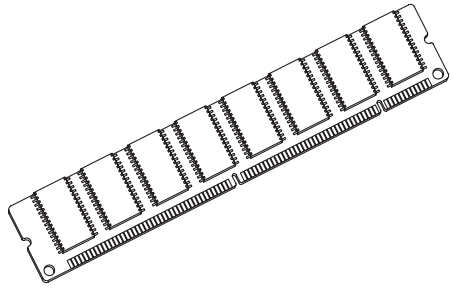
Acorde con las primeras tecnologías en alguna memoria dinámica, cuando se producen dos lecturas consecutivas sobre el mismo chip el tiempo de acceso se penaliza con una cuota de tiempo adicional. Esto se debe a que en cada lectura se descargan ligeramente las celdas de la posición accedida, por lo que es necesario esperar para que se vuelvan a recargar. Este período se denomina "tiempo de precarga".

9.7.2.2 Módulos

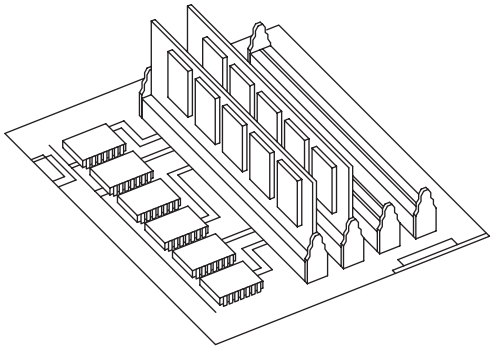
Se trata de la forma en que se juntan los chips de memoria, del tipo que sean, para conectarse a la placa base de una computadora. Son pequeñas placas en las que se sueldan los chips de RAM (del tipo que sean), con patillas o pines en un extremo; al conjunto se lo llama módulo. La diferencia de los DIMM frente a otros módulos más antiguos, como los SIMM, es que son más largos (unos *13 cm* frente a *10,5*) y tienen más contactos eléctricos, además de dos ranuras para facilitar su colocación correcta.



Esquema de un módulo



Dibujo de un chip.



Inserción de los módulos en una placa base

Ejemplo, parte de la descripción de un DRAM DIMM.

Pin	*Identificación*	*Description*	*Tipos de línea*
1	VSS	Ground	Conexión a tierra
2	DQ0	Data 0	Línea de dato
3	DQ1	Data 1	Línea de dato
4	DQ2	Data 2	Línea de dato
5	DQ3	Data 3	Línea de dato
6	VCC	+5 VDC o +3.3 VDC	Alimentación
7	DQ4	Data 4	Línea de dato
8	DQ5	Data 5	Línea de dato
...	...	...	...
...	...	...	...
...	...	...	...
28	/CAS0	Column Address Strobe 0	Línea de control de dirección de columna 0
29	/CAS1	Column Address Strobe 1	Línea de control de dirección de columna 1
30	/RAS0	Row Address Strobe 0	Línea de control de dirección de fila 0
31	/OE0	Output Enable	Habilitación para salida de dato
32	VSS	Ground	
33	A0	Address 0	Línea de dirección
34	A2	Address 2	Línea de dirección
...	...	...	...

9.7.2.3 Velocidad del bus de memoria

Para calcular el ancho de banda de las memorias se realiza el producto entre el ancho de bus expresado en bytes y la frecuencia efectiva de trabajo en MHz, también denominada velocidad física o real.

Por ejemplo, una memoria identificada como *DDR200* se llama también *PC1600*, porque (*64 bits/8) bytes * 200 MHz = 1600 MB/s = 1,6 GB/seg*, que es la velocidad de transferencia o ancho de banda del bus de memoria.

Cuanto más rápido sea un dispositivo, más difícil será de fabricar y, por lo tanto, más caro. Ésta es la razón fundamental por la que la tecnología DDR SDRAM está instalada desde hace bastante tiempo en el mercado de memoria para computadoras.

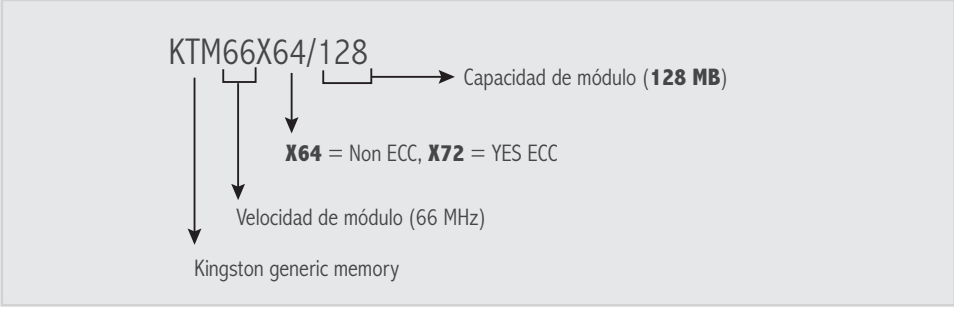
9.7.2.4 Caso ejemplo: DDR SDRAM (double data rate synchronous DRAM).

Éste es un ejemplo de tecnología. Esta memoria envía los datos dos veces (*doble data rate*) por cada ciclo de reloj, como consecuencia opera al doble de velocidad del bus del sistema o bus de memoria, sin necesidad de aumentar la frecuencia de reloj. De esta forma, una memoria con tecnología DDR que funcione con una señal de reloj "real", de, por ejemplo, *100 MHz*, enviará tantos datos como otra sin tecnología DDR, que funcione a *200 MHz*. Por ello, las velocidades de reloj de los DDR se suelen dar en lo que podríamos llamar "MHz efectivos o equivalentes" (en nuestro ejemplo, *200 MHz, "100 MHz x 2"*).

Esto significa que a pesar de utilizar la misma frecuencia que sus antecesoras, la velocidad viene indicada con un valor duplicado: *DDR200* (*100 MHz*). La conocida denominación PCxxx no indica megahertz, sino la tasa de transferencia máxima en MB/s: *PC1600*.

DDR2-800 trabaja a *800 MHz* con un bus de memoria de *400 MHz*, y ofrece un ancho de banda de *6,4 GHz*, que se calculan como *64 bits* del bus dividido *8* para obtener la cantidad de bytes (o sea, *8 bytes*) por *800 MHz (6400 MHz = 6,4 GHz*), de ahí la denominación *PC6400*.

9.7.2.5 Caso ejemplo: identificación de un chip SDRAM



9.7.2.6 Detección y corrección por ECC

El ECC 743 (en inglés, *error checking and correction*) es un código de bits que permite la detección y la corrección de errores, basado en un algoritmo complejo, que se utiliza fundamentalmente en computadoras de alta gama, como los servidores de red. El sistema trabaja en conjunción con el controlador de memoria, y anexa a los bits de datos los denominados ECC, que se almacenan junto con los de datos. Estos bits extras o redundantes, junto con la decodificación correspondiente, sirven para realizar una comprobación en el momento de la lectura.

Su diferencia principal con la paridad es que puede detectar el error de un bit y **corregirlo**, con lo que, en general, el usuario no detecta que el error se produjo. Según el controlador de memoria utilizado, el sistema ECC también puede detectar errores de *2, 3* y *4 bits* (muy raros), aunque en este caso no puede corregirlos; en estas situaciones el propio sistema devuelve un mensaje identificándolo como "un error de paridad".

En ambos casos, paridad o ECC, cuando se detecta un error se produce una interrupción no mascarable, que forma parte de la clasificación de interrupciones externas denominadas NMI, que describiremos en capítulos siguientes.

Ejercicio 4:

Pruebe ahora interpretar la información técnica siguiente:

Capacidad de almacenamiento	*1 GB*
Tipo	DRAM
Tecnología	DDR II SDRAM
Factor de forma	DIMM de 240 pines
Velocidad de memoria	800 MHz (PC2-6400)
Tiempos de latencia	CL5
Comprobación de integridad de datos	ECC
Características de la RAM	8K refresh
Configuración de módulos	128 · 64
Organización de los chips	64 · 8
Voltaje de alimentación	1.8 V
Ranuras compatibles	1 x memoria - DIMM de 240 pines

9. 8 La memoria como en un espacio lógico

Para el procesador la memoria es un hardware (con forma de matriz de m · n, como hemos relatado en capítulos anteriores) que contiene las instrucciones y los datos para procesar, con muchos megabytes o gigabytes de almacenamiento, identificables por una dirección física (en inglés, *physical address*). Ésta se puede considerar el número ordinal del byte dentro del mapa de direcciones a las que se puede acceder.

El procesador no hace diferencia entre las distintas tecnologías que pueden constituir una memoria principal. Simplemente envía la dirección física del byte vía bus de direcciones.

El procesador no hace diferencia entre las distintas tecnologías que pueden constituir una memoria principal. Simplemente envía la dirección físicaa del byte vía bus de direcciones.

La manera en que una memoria se administra y se gestionan sus accesos depende fundamentalmente del sistema operativo. La evolución de los sistemas operativos en cuanto a administración de memoria y las nuevas plataformas de los procesadores, que le permiten administrarla de manera más eficiente, constituyen el núcleo central de este apartado.

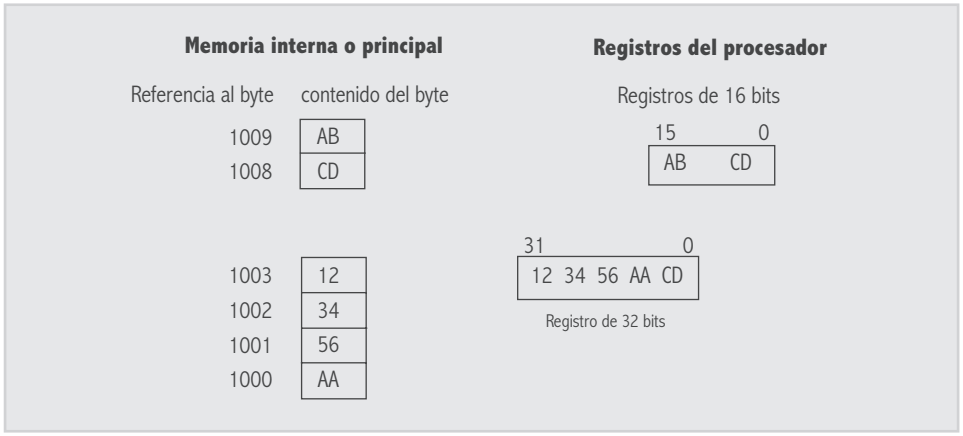
Además, puede suceder que la computadora tenga menos memoria real que su posible capacidad de direccionamiento. Por ejemplo, una computadora puede tener un procesador cuyo posible espacio de direccionamiento es de *4 Gbytes* y sólo tener instalados *2 Gbytes*.

También puede suceder que la cantidad de bits de una referencia a memoria no alcance para acceder a toda la memoria. Por ejemplo, en los microprocesadores *Intel 80286* se utilizaban números de *16 bits* para direccionar a *1 M* direcciones. En ambos casos, la capacidad de direccionamiento lógico no es congruente con la capacidad física real de la memoria. Esto conduce a la aplicación de técnicas de direccionamiento a memoria. En estos casos las direcciones en los programas no son direcciones físicas sino lógicas, y para lograr el acceso deberán traducirse (mapearse). El procedimiento de "mapeo" consiste en aplicar un algoritmo para establecer la correspondencia entre direcciones lógicas y físicas, que a veces incluyen una o más tablas que contienen "una parte" de la dirección.

A continuación se pretende detallar cómo el procesador y los programas que ejecuta "ven" la memoria. Llamaremos modelo a la vista lógica de la memoria que es totalmente independiente de la estructura interna de cada chip y de su organización en la placa.

9.8.1 Almacenamiento de bytes en memoria. Big-Endian y Little-Endian

El almacenamiento en la memoria, ya sea de datos o código de programa, sigue un orden específico. Como se podrá visualizar más adelante, el byte **más significativo** se almacena en la dirección más **baja** y el más significativo en la más alta. Esta forma de almacenamiento se denomina Big-Endian y se aplica tanto al código como a datos de tipo no numérico. Si el dato es numérico y su tamaño es de *2* o *4* u *8 bytes* (palabra, doble palabra, cuádruple palabra), cada octeto se almacena en memoria en forma invertida. Por ejemplo, si la representación hexadecimal de una palabra, es *ABCD*, en memoria se almacena primero *CD* y luego *AB*. O sea que la palabra se lee *CDAB*. En esta convención denominada de almacenamiento inverso o Little-Endian, el byte **menos significativo** se almacena en la dirección más **baja**. Esto ocurre para cualquier entidad numérica, incluso para datos en representación de punto flotante o cuando se almacena una referencia a memoria. Por ejemplo, el desplazamiento hexadecimal *0300* de una instrucción *MOV AH [0300]* se almacena como *8A26**0003***, donde *8A26* es el código de operación almacenado *Big-endian* y la referencia a memoria, *0003* almacenado *Little-Endian*.



Esquemas de almacenamiento en memoria y almacenamiento en registros para estructura de datos tipo palabra y doble palabra

9.8.2 Gestión de memoria y modos de operación de los procesadores

Una de las formas de administrar la memoria es hacerlo sin protección. En los microprocesadores *80x86* al modo de trabajo desprotegido se lo llamó modo real. En los sistemas operativos que admiten multitarea uno de los recursos a compartir es la memoria, por lo tanto, se deberían prevenir conflictos entre las distintas áreas locales de cada una de ellas. Esto es responsabilidad en parte de la administración del sistema operativo, y en parte del soporte hardware que desde el procesador gestiona el mapeo de algunos procesadores. El modo de operación del procesador, denominado modo protegido, implica entre otras cosas protección de memoria. Por supuesto que en ambos modos la memoria no tiene la misma vista lógica, razón por la cual ambos modos de operación no pueden estar activados en forma simultánea; el procesador puede conmutar de un modo al otro. En la actualidad, y por citar un ejemplo, un microprocesador AMD64 puede operar en cuatro modos distintos. O sea que hay una de-

pendencia entre la forma de administrar la memoria y el soporte que brinda el hardware del procesador para hacerlo posible. Éste es el motivo por el que se incluyen algunos conceptos de administración de memoria en relación con el soporte hardware que requiere.

La forma en la que se obtenía una dirección física en modo real se encuentra ampliamente detallada en el anexo A, si bien en el presente ha caído en desuso, algunos procesadores y sistemas operativos permiten conmutar a él. Aquellos lectores que pueden acceder desde sus computadoras personales a este modo pueden realizar los ejercicios allí planteados con el propósito de comprender en la práctica cuestiones más abstractas relacionadas con el estudio de procesadores más evolucionados, pues el enfoque del laboratorio les permite comprender en cada ejercicio los fundamentos teóricos relacionados con el procesamiento de instrucciones, la gestión de memoria y la gestión de archivos.

Por dar un ejemplo, las plataformas de *32 bits* en vigencia admiten varios modelos de gestión de memoria:

- Como un espacio de direcciones físicas, conocido como espacio lineal de direcciones o modelo plano.
- Como un espacio que alberga segmentos puros, que son bloques lógicos de longitud variable o modelo segmentado.
- Como un espacio que alberga páginas, que son bloques lógicos de tamaño fijo o modelo paginado.
- Como un espacio que alberga segmentos-paginados, que es un modelo híbrido.

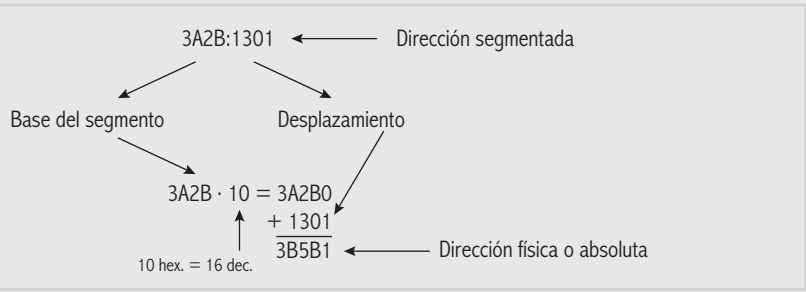
Caso ejemplo: cálculo de direcciones físicas en modo real

Por su simplicidad usaremos como primer ejemplo el modo real de los procesadores Intel x86.

Una dirección de memoria se representa con un número binario que permite identificar cada octeto dentro de ella. Cuando la cantidad de bits de la dirección es *16*, el rango de direccionamiento queda limitado a $(2^{16} - 1) = 65535$, que es la dirección del último octeto direccionable, esto es, que de *0* a *65535* hay *65536* octetos direccionables. Se deduce que la capacidad de memoria total es, entonces, de *64 Kbytes (65536 octetos*). Si se quiere acceder a más memoria, por ejemplo, a *1 mega*, se puede segmentar la memoria en bloques de hasta *64 KB*. Para ello se utilizan dos entidades de *16 bits* y un algoritmo que las relacione. La primera entidad hace referencia a una zona de la memoria dentro del megabyte; la segunda indica el desplazamiento del octeto dentro de esa zona. Como el desplazamiento es de *16 bits*, entonces la capacidad máxima de cada zona de memoria es de *64 KB*. Si el total de la memoria es *1 MB* y cada zona es de *64 KB*, habrá *16 zonas (1024 KB/64 KB)*. Cada una de las zonas es un segmento. La primera entidad indica dónde comienza el segmento y la segunda, la posición del octeto dentro del segmento. Así, un octeto de memoria puede identificarse con una dirección lógica, por ejemplo, *3A2B:1301*.

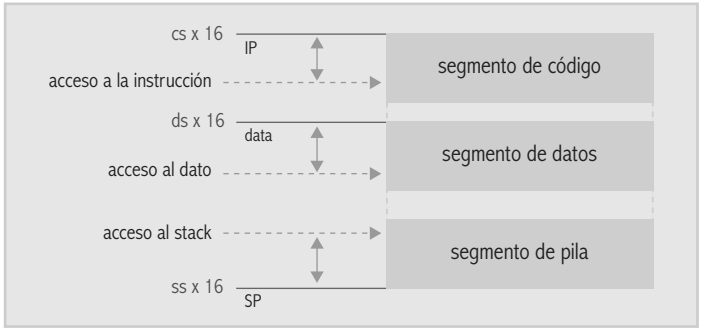
La pregunta ahora es: ¿Cómo se calcula la dirección física? Se sabe que *1 mega es* $1\ K \cdot 1\ K = 2^{10} \cdot 2^{10} = 2^{20}$; esto implica que con *20 bits* se calcula la dirección de la última palabra ($2^{20} - 1$). Se necesita un algoritmo que permita generar con dos referencias de 16 bits una dirección de *20 bits*. Esto se logra así: A $3A2B_{(16)}$ se le agrega un $0_{(16)}$, o, lo que es lo mismo, se lo multiplica por $10_{(16)}$. A este valor se le suma $1301_{(16)}$, que es el desplazamiento dentro del segmento. Entonces, *3B5B1* es la dirección física del octeto en memoria, que también se conoce como dirección absoluta.

Caso ejemplo: algoritmo de cálculo que realiza la MMU (memory manager unit) del procesador en modo real



9.8.3 Modelo de memoria segmentada pura

Para acceder a la memoria física se puede utilizar una técnica denominada segmentación, que conceptualmente consiste en dividirla en territorios pequeños con límites determinados por el software. Un segmento es un bloque lógico en memoria, de tamaño variable. El control del tamaño de los segmentos es responsabilidad del sistema operativo que los administra. Los segmentos contienen un solo tipo de objeto, por ejemplo, en una memoria se puede organizar un segmento para el código ejecutable de un programa, otro para los datos y otro para una pila que él utilice. Segmentos distintos no necesitan estar en zonas contiguas, pero es necesario que un segmento como entidad lógica si lo esté. En el esquema se representan tres segmentos cuyo acceso se referencia según el algoritmo presentado en el caso ejemplo anterior.



Denominaremos dirección lógica a la entidad binaria que identifica una locación de memoria a nivel software, que no se corresponde con la dirección física y, por lo tanto, necesita ser mapeada. Para direccionar una posición de memoria dentro de un segmento, cada dirección lógica se compone de dos entidades: Una para identificar el segmento y otra para identificar un desplazamiento dentro del segmento. El cálculo lo hemos visto en el ejemplo anterior.

9.8.4 Modelo de memoria virtual

Ya en la época de las primeras computadoras comerciales, la memoria principal quedaba chica en relación con los requerimientos del programa, y el concepto de un programa almacenado en su totalidad en memoria principal parecía brillar en la teoría de los libros y hundirse en la práctica. Por aquel entonces, las memorias secundarias de tipo disco ya existían y los

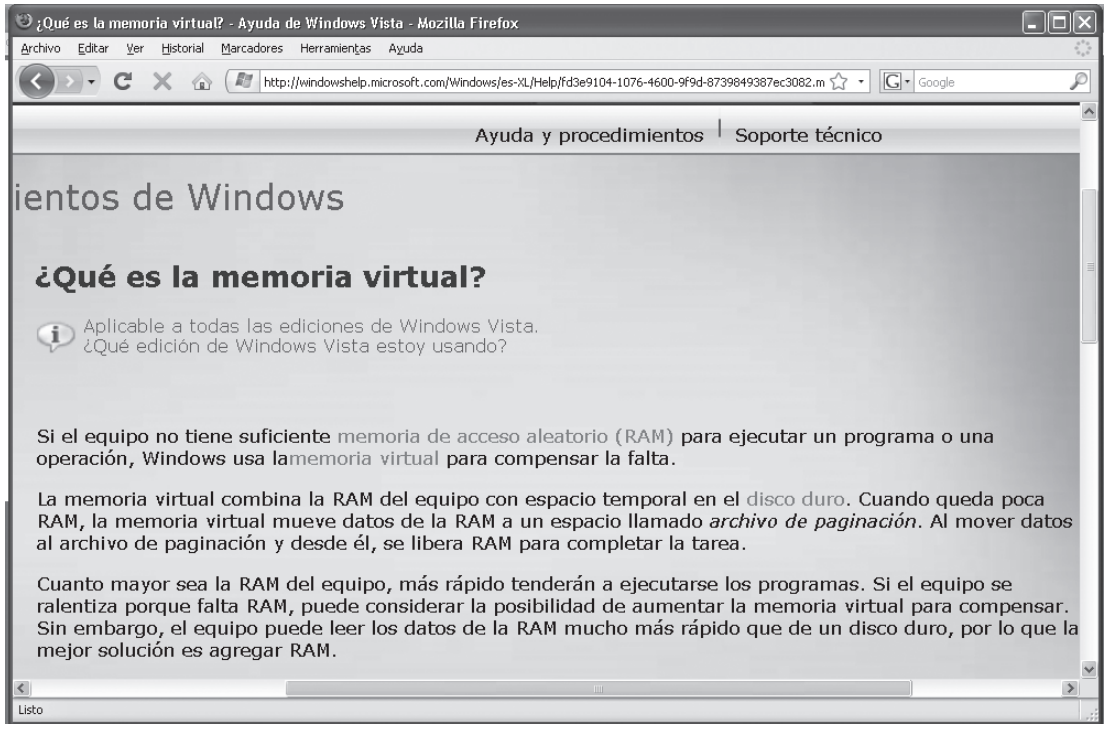
programadores simulaban el concepto original de programa almacenado fragmentando el programa. De esta manera, un fragmento cabía en la memoria física y el programador provocaba la búsqueda del fragmento siguiente cuando el actual ya se había ejecutado. Con el tiempo se fue perfeccionando una técnica para que tanto la fragmentación como el intercambio de fragmentos entre el disco y la memoria se hicieran en forma automática; esto es, en la actualidad la gestión está a cargo del sistema operativo sin intervención del programador. Esta técnica se conoce como memoria virtual y posibilita la ilusión de contar con una memoria principal lo suficientemente amplia como para contener el programa y los datos, aunque esto no sea real.

La técnica de memoria virtual involucra por un lado memoria y discos, y por el otro al procesador y al sistema operativo que la administra. Por supuesto que una dirección lógica perteneciente al espacio de direccionamiento virtual es mayor que una dirección física, y la llamaremos dirección virtual. Cuando se crea una tarea se asigna en disco un archivo temporal, que es copia del archivo ejecutable dividido en secciones, es un área de almacenamiento separada de la que ocupa el archivo ejecutable. Cada sección puede ser un segmento o una página, o un segmento paginado. La idea de "fraccionar" el ejecutable es que se puedan llevar a memoria secciones de código o datos más pequeñas para alojarlas en memoria principal en un menor espacio de almacenamiento. Como no se llevan todas las secciones, cada vez que se agote el uso de cada una se deberá demandar por otras alojadas en disco. De este modo los modelos de memoria que utilizan memoria virtual se conocen como *segmentación por demanda* y *paginación por demanda*, y se describirán a continuación.

Una dirección virtual debe traducirse a una dirección física, o sea que se mapea. En un modelo virtual el mapeo es una actividad que se intercala durante la ejecución del programa, por cada búsqueda de instrucción y por cada búsqueda de dato; asimismo, desde el punto de vista del hardware la gestiona una unidad específica, cuya función es traducir la dirección virtual a dirección física.

La memoria virtual es una forma de administrar la memoria para asignar más de ella a cada tarea, utilizando almacenamiento en disco, que suele ser el dispositivo de acceso directo más rápido; así, se simula que se dispone de una memoria mucho más amplia que la que el sistema tiene en realidad.

Un segmento es un bloque lógico de tamaño variable. Cada segmento contiene información de la misma clase (código, datos, pila).



9.8.5 Modelo de memoria virtual paginada o paginación por demanda

9.8.5.1 Mapeo directo

Para administrar un archivo temporal bajo esta modalidad, se requiere una tabla de mapeo alojada en la memoria principal, con tantas entradas como páginas virtuales tenga el archivo. El método consiste en dividir la memoria principal en bloques de longitud fija, llamados páginas físicas, y el espacio de direccionamiento virtual en bloques de igual longitud, llamados páginas virtuales. Por ejemplo, si se considera como longitud fija de una página *1 Kbyte*, una memoria principal organizada como una matriz de *8192 x 8* quedará dividida en 8 páginas físicas.

Si el espacio de direccionamiento virtual es de *64 Kbytes*, quedará dividido en *64* páginas virtuales. Así, la dirección virtual referencia con *6 bits* el número de página; 2^6 es igual a *64*, y 2^{10} es igual a *1 K*. Los *10 bits* restantes hacen referencia a una de las *1024 (1 K)* posiciones dentro de la página. En cambio, el identificador de página física es de *3 bits* para referenciar a una de las *8* páginas físicas. Uno de los bits de atributo de página, alojado en la tabla de mapeo indica la presencia o no de una página virtual en la memoria principal, en la entrada de la tabla que le corresponda. Por ejemplo, en una página que contiene datos y está presente, los *3 bits* identificadores de la página física concatenados con los bits del campo DATA de la instrucción logran la dirección física que recibe el MAR (dirección de *3 + 10 = 13 bits*, donde $2^{13} = 8192$ posiciones totales de la memoria principal).

Todo programa tiene asignado un espacio de memoria virtual (grande) mapeado en memoria física (más pequeña). Mientras el proceso de mapeo sea exitoso, o sea que una dirección virtual se transforme en una real de una página presente, no se producen errores; en cambio, si el programa intenta acceder a una posición perteneciente a una página no presente en la memoria principal, el procesador genera un error conocido como *page fault* (falta de página), que provoca una interrupción de tipo excepción. El error se advierte al sistema operativo, cuya función será buscar una nueva página en el disco.

Antes de esto libera una página, y si ésta se actualizó, primero la copia en el disco antes de reemplazarla (este procedimiento se denomina en inglés *swap out*). Recién entonces se utiliza esa página física con la página demandada por el programa. Cuando se recupera una página almacenada en disco, el procedimiento se denomina *swap in*. Al producirse continuas liberaciones y recuperaciones de páginas la gestión de memoria virtual puede congestionarse y entrar en un estado conocido como *trashing*, término que denota el carácter de depósito que tiene la memoria de disco durante la gestión de memoria virtual (*trash* = basurero). Un término más adecuado para denotar un intercambio excesivo de páginas es hiperpaginación. La utilización eficiente de la memoria virtual depende básicamente del sistema operativo, esto es, de una técnica de software que determine el número de página virtual que debe retirarse de la memoria principal con un algoritmo de reemplazo eficiente. Éste selecciona la posición que tenga menos probabilidad de ser referenciada a *posteriori*, y así evita page-faults futuros; también realiza la gestión de recuperación de una página desde disco y su alocación en la memoria principal.

De todas estas actividades la gestión del sistema operativo que más tiempo demora es la de una E/S (memoria-disco y disco-memoria), por lo tanto, no es deseable una sobrecarga de *swaps*, incluso puede provocar que el uso de memoria virtual, si no está bien administrada, traiga más problemas que soluciones.

9.8.5.2 Mapeo asociativo

La cantidad de páginas físicas en memoria real será siempre menor que la cantidad de páginas virtuales posibles. Así, para el ejemplo dado, $2^3 = 8$ páginas físicas y $2^7 = 128$ páginas virtuales, resulta una tabla de mapeo de *128* entradas, de las cuales sólo ocho contendrán in-

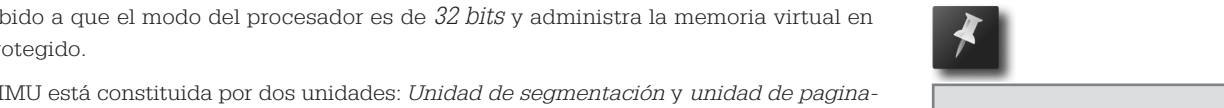
formación de referencia a página física, o sea que *120* están siempre vacías; esto resulta ineficiente respecto del espacio de almacenamiento utilizado en memoria real para alojar la tabla.

Una mejor idea es crear una tabla que tenga igual cantidad de entradas como páginas físicas existan y que pueda implementarse en una pequeña memoria asociativa.

Cada entrada de la tabla contiene un campo que indica un número de página virtual actualmente presente en memoria principal y un campo que indique el número de página física correspondiente. Cuando se produce una consulta a la tabla, se compara el número de página virtual demandado con todos los campos "número de página virtual" de la tabla; si hay coincidencia y, por ejemplo, se está buscando un dato, entonces la dirección se logra concatenando los bits del número de página física con los del campo DATA del código de la instrucción que está en ejecución.

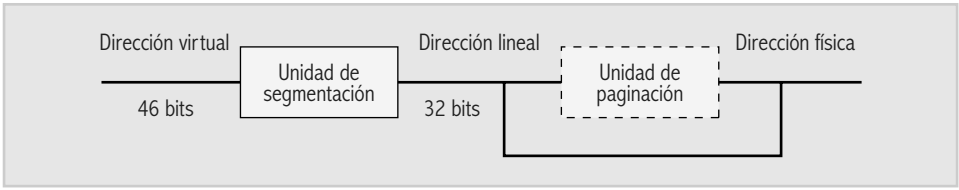
9.8.6 Memoria virtual segmentada o segmentación por demanda

En este modelo se organiza el espacio de direcciones virtuales en bloques de tamaño variable, los segmentos. El concepto de segmentación es el mismo, sólo que ahora los segmentos pueden residir tanto en la memoria principal como en el disco. Tomaremos como caso ejemplo el modelo de un procesador INTEL de *32 bits*. En él la unidad de cálculo de dirección física se denomina unidad de gestión de memoria (MMU por sus siglas en inglés, *memory manager unit*), pero esta vez el cálculo de la dirección física se realiza de otra manera, debido a que el modo del procesador es de *32 bits* y administra la memoria virtual en modo protegido.



La MMU está constituida por dos unidades: **Unidad de segmentación** y **unidad de paginación**.

La MMU está constituida por dos unidades: *Unidad de segmentación* y *unidad de paginación*, como se muestra en el esquema que sigue. Como la memoria principal es más pequeña que la virtual, la unidad de segmentación se encarga de detectar si un segmento se encuentra en el disco y no en la memoria física, y se lo comunica al sistema operativo para que posibilite el traslado. Cuando sólo se activa la unidad de segmentación, la traducción de una dirección virtual de *46 bits* genera una dirección lineal que corresponde directamente a una dirección física de *32 bits*, o sea que la unidad de paginación está inhabilitada.



Esquema de las unidades de la MMU.

La **tabla descriptora de segmentos** se referencia con un registro base especial asociado al microprocesador, denominado **registro de tabla local activa** (Local Descriptor Table Register o LDTR)

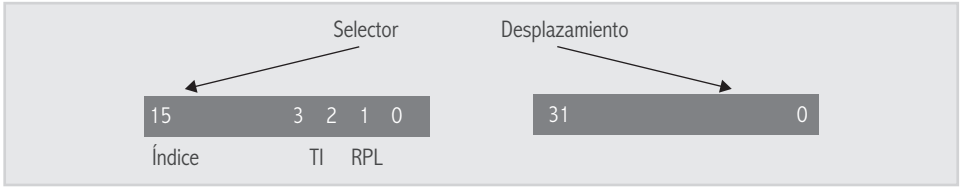
Una tarea se asocia con su *tabla descriptora de segmentos*. Cada entrada de esta tabla contiene: Un campo de *32 bits* identificador de la base del segmento (información que sólo tiene sentido si el segmento reside en memoria principal), un campo de *20 bits* que representa el tamaño máximo que el segmento puede alcanzar, y un campo de *12 bits* que permite determinar los atributos, por ejemplo, un bit de presencia, el bit de segmento actualizado, un bit de tipo de segmento (código, dato creciente, dato decreciente), etc.

Si el sistema operativo admite que se puedan cargar varias tareas distintas en la memoria, cada una estará asociada con su *tabla descriptora de segmentos*. La tabla se referencia con un registro base especial asociado al microprocesador, denominado *registro de tabla local activa*

(LDTR). Este registro almacena la dirección de comienzo de la tabla asociada a la tarea que se ejecuta en la CPU. Para conmutar de una tarea a otra, el sistema operativo cambia el valor del LDTR, y así cambia el puntero a la nueva tabla descriptora. La LDT de una tarea permanece en memoria mientras la tarea exista, como un recurso para administrar la gestión, y es creada, actualizada y eliminada por el sistema operativo. La MMU consulta esta tabla por cada nuevo cálculo de dirección física, ya sea de una instrucción en fase de búsqueda o de un dato en fase de ejecución. Los datos pueden ser del programa, conocidos con el atributo dato creciente, o de la pila, conocidos como "dato decreciente" (habida cuenta del criterio LIFO de esta estructura).

La pila es una estructura de datos con un criterio de acceso LIFO (el último que entró es el primero en salir), que, asociada a un proceso, permite almacenar parámetros y direcciones de retorno.

Una dirección virtual está constituida por dos entidades: un campo selector y un campo desplazamiento. Para acceder a un segmento de código de la tarea en una *x86*, se consideran como selector los *14 bits* de orden superior del registro de segmento denominado CS (*code segment*), y como desplazamiento el valor del EIP (*instruction pointer de 32 bits*). Estas dos entidades constituyen la dirección virtual que ingresa en la MMU, cuando se debe calcular la dirección física de una instrucción. El selector se considera el índice que apunta a una de las entradas de la *tabla de descriptores de segmentos* (o sea que es el desplazamiento respecto del valor del LDTR).



Interpretación de una dirección virtual

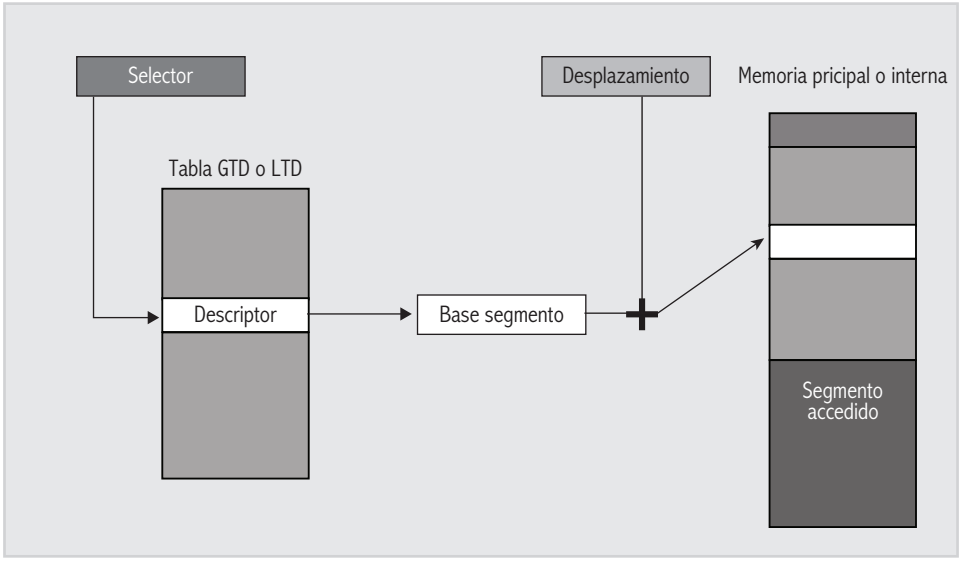
Caso ejemplo: dirección virtual IA-32

El campo selector tomado de un registro de segmento es "interpretado" por la unidad de segmentación, de modo que los *14 bits* de orden superior constituyen el índice en la tabla de descriptores. TI indica si se está accediendo a la tabla de descriptores locales o globales, y RPL marca el nivel de privilegio del segmento. Los *32 bits* restantes se toman del campo EDATA de la instrucción, en el caso de un segmento de dato, o del EIP o del ESP, en el caso de segmentos de código o pila, respectivamente.

En el caso de un acceso a segmento de código, la dirección física se calcula sumando el valor de la base del segmento, ubicada en el descriptor, al valor del registro de desplazamiento EIP, y se conoce como dirección lineal que, en este caso, es la dirección física buscada.

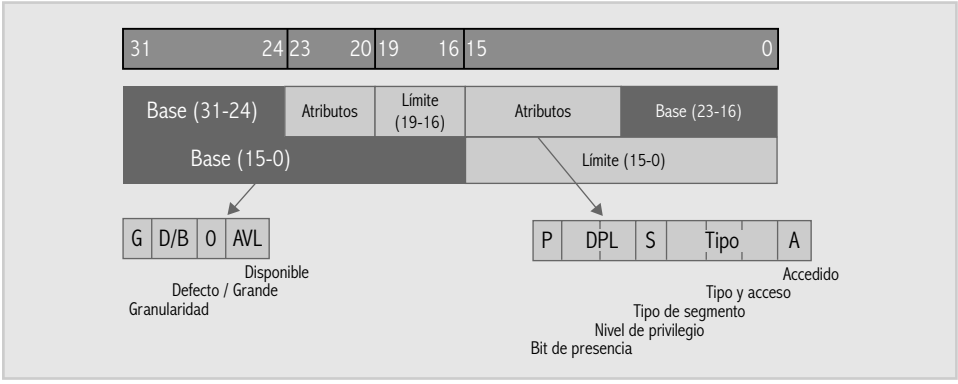
Una ventaja de este modelo es el nivel de privilegio asociado a cada segmento, especificada en los *2 bits* de orden inferior del registro de segmento (en este caso CS), que inhibe, por ejemplo, la operación de escritura si el segmento contiene código ejecutable, que no debe modificarse. La asignación de un segmento a distintas tareas, vía la tabla de segmentos, permite que varios procesos utilicen un segmento almacenado una sola vez en memoria sin necesidad de copias, con el consiguiente ahorro de memoria principal. En este caso el segmento se almacena en un área común determinada por el sistema operativo y conocida como área global. El área global cuenta con su propia *tabla descriptora de*

segmentos globales y con su propio registro base que apunta a la tabla GDTR (*registro de tabla global activa - global descriptor table register*).



Unidad de segmentación.

9.8.8.1 Caso ejemplo de un descriptor de segmento en una IA-32 bits



Como se ve en el esquema, un bit en el campo atributo indica si el segmento está presente en la memoria principal. Si esto es así, entonces los campos BASE se unifican de modo de indicar la base del segmento presente en memoria principal; a este valor se sumará el campo desplazamiento de la dirección virtual. Por supuesto que la información escrita como base sólo tiene sentido para un segmento con bit de presencia seteado.

Descripción de algunos atributos

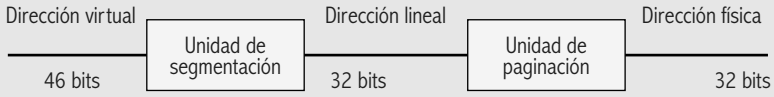
G bit de granularidad indica si el segmento está a su vez dividido en páginas. En este caso se debe considerar el campo límite como cantidad de páginas.

P bit Present indica si el segmento está en memoria principal.

S System indica si el segmento es de sistema.

Modelo de memoria virtual con segmentos-paginados.

MMU con la unidad de paginación habilitada.



En este caso cada segmento es una agrupación de páginas de longitud fija, o sea que la longitud del segmento varía en relación con las páginas asociadas a él. Se debe pensar que un segmento es una unidad de información físicamente contigua; si el tamaño del segmento de una tarea es mayor que los huecos libres que existen en la memoria principal, la tarea deberá esperar a que se produzca un espacio de tamaño suficiente como para contenerlo. La idea de dividir un segmento en páginas de tamaño fijo y reducido permite que el segmento mantenga su propiedad de entidad lógicamente indivisible, cuyo tamaño se adapta al objeto que contiene (código, datos o pila), pero que pueda estar físicamente distribuido en zonas de almacenamiento en memoria principal no contiguas. Además, no es necesario que se aloje en la memoria en su totalidad, vasta con encontrar un espacio físicamente contiguo para una de sus páginas. Esto permite el aprovechamiento de huecos libres de menor tamaño.

En este nuevo modelo la *dirección lineal* ingresa en la unidad hardware denominada *unidad de paginación* que la traduce a una *dirección física* (una posición dentro de una página física). El funcionamiento de la unidad de paginación es optativo. La paginación en este procesador opera sólo en modo protegido. Teniendo en cuenta que con *32 bits* de la dirección lineal se alcanza un espacio de direccionamiento de 4 gigabytes (2^{32}) y que el tamaño de cada página está definido en *4 KB* (2^{12}), la cantidad de páginas virtuales posibles es un millón ($2^{32}/2^{10} = 2^{20}$); en consecuencia, para direccionar una página se necesitan *20 bits*.

Cada vez que la unidad de paginación detecta ausencia de página en memoria genera la excepción *page fault*, que provoca un llamado al sistema operativo para que traiga la página del disco.

En el modelo presentado como mapeo directo, para saber si una página está en memoria principal o no, se debería mantener, para ésta nueva dimensión, una tabla de un millón de entradas que indicase su ausencia o presencia. Para no alojar una tabla tan grande en memoria, la traducción de la dirección se efectúa en dos niveles:

La unidad de paginación crea para cada tarea una tabla de *1 K* entradas de *32 bits* cada una, denominada *directorio de tablas de página*, que reside en memoria principal. El registro base que apunta al comienzo de esta tabla es un registro del microprocesador denominado *CR3*. El valor de los *10 bits* de orden superior de la dirección lineal de *32 bits* constituye el desplazamiento dentro del directorio de tablas, y los *32 bits* del contenido de la entrada accedida en el directorio de páginas constituyen la base de una segunda tabla de páginas. Dentro de esta tabla los *10 bits* centrales de la dirección lineal constituyen el nuevo desplazamiento. Por último, la entrada así accedida de la tabla de páginas de *32 bits* se divide en *20* y *12*. Los *20* de orden superior se concatenan con los *12 bits* de orden inferior de la dirección lineal para formar la dirección física final. Los *12 bits* de orden inferior de la entrada de la tabla de páginas, son los bits del campo atributo de página, de los cuales no todos se utilizan. A continuación, se indican los atributos más importantes asociados con ellos.

D Bit dirty, indica si la página se escribió.

P Bit present, indica si la página está presente.

A Bit Access, indica si se accedió a la página tanto para lectura como para escritura.

TLB, *translation lookaside buffer*

Es una memoria de tipo caché que almacena las últimas direcciones físicas traducidas por la unidad de paginación. Su inclusión en el diseño pretende solucionar el retardo producido por el acceso a las tablas de página almacenadas en la memoria principal, cuyas zonas de almacenamiento para lectura/escritura son de tecnología dinámica. Si la dirección está presente en la caché, se obtiene la dirección física en pocos nanosegundos; si no se encuentra, el tiempo de traducción sólo se penaliza en esos pocos nanosegundos.

9.9 Administración de memorias externas

Toda memoria alojada en una unidad de E/S se denomina memoria externa o secundaria; para algunos autores también se llama memoria de masa o masiva. Su inclusión en este apartado tiene como objetivo que se pueda establecer su relación desde el punto de vista lógico con los dos niveles jerárquicos superiores en la memoria principal.

Las características técnicas y la organización física de los dispositivos de almacenamiento se abordarán en el capítulo "Dispositivos de entrada/salida".

La razón por la cual ubicamos las memorias secundarias en un último nivel de la jerarquía presentada responde a que uno de los parámetros elegidos para establecer esa jerarquía es, como ya se explicó, el tiempo de acceso a la información que contienen, ya que se consideran muy lentas con respecto a las memorias de semiconductores. En este nivel se utiliza como medida de tiempo el **µs** (**microsegundo** = 10^{-6} *seg*). La mayoría de los parámetros de tiempo que permiten comparar dispositivos de distintos fabricantes se expresa en términos de µs.

Ejercicio 5:

Indique cuántos nanosegundos representan un microsegundo.

Las memorias externas se incluyen en toda computadora debido a su carácter no volátil, o sea que son imprescindibles para retener información cuando el sistema está desconectado de la fuente de alimentación. También se denominan memorias secundarias, nombre utilizado por los autores que llaman memoria principal o interna a la memoria de semiconductores fuera de la CPU, y consideran el término interno como dentro de la computadora, aun cuando la memoria esté fuera de la CPU.

En un principio la CPU obtiene un programa y los datos necesarios desde el almacenamiento secundario. El programa se copia literalmente desde su ubicación en el almacenamiento secundario a la memoria de lectura/escritura RAM de la computadora, lo que le permite ejecutarse con mayor rapidez. Además, la CPU sólo tiene acceso directo a esta última por medio del bus de direcciones. De la misma forma, cualquier dato necesitado por el programa se copia desde el almacenamiento secundario hacia la memoria, y esos datos retornarán al almacenamiento si se los modifica y se quiere guardar los cambios en forma permanente.

Los **dispositivos de E/S** más utilizados para "soportar" estas memorias son el disco rígido y los dispositivos de tecnología óptica, como la unidad de DVD.

Las memorias externas actúan de depósito y toda información que no se necesite para el proceso actual se almacena en ellas hasta que sea demandada. Los parámetros para tener en cuenta a la elección del **soporte** son:

- El costo por bit almacenado,
- el tiempo de acceso,
- el modo de acceso,
- la capacidad de almacenamiento,
- el tipo de tarea que utiliza la información.

Se denomina **soporte** al elemento físico que retiene los bits de información. Los soportes utilizados con mayor frecuencia son los medios magnéticos. En estos soportes el costo por bit está dado por la densidad de grabación medida en bytes por pulgada (en inglés, BPI, sigla de *bytes per inch*). La densidad de grabación depende de la calidad del soporte y la capacidad del transductor para representar o detectar un bit en el medio.

El tiempo de acceso a la información se mide en microsegundos y depende del tipo de unidad accedida. Su lentitud respecto de las memorias de semiconductores se debe fundamentalmente al carácter electromecánico de la unidad o dispositivo de E/S. Para considerar dos parámetros que influyen en el tiempo de acceso, se puede considerar el tiempo de búsqueda en el soporte y el tiempo de transferencia entre el par memoria principal-memoria secundaria.

9.9.1 Archivos

Todo tipo de información se aloja en las memorias externas en una estructura denominada archivo (en inglés, *file*).

Un código ejecutable es un archivo de programa" y los datos de entrada o salida de un proceso constituyen un archivo de datos. Un archivo se diferencia de los demás por el **nombre simbólico** que se le asigna en el momento de su creación.

Por lo general el nombre simbólico hace referencia al tipo de información que contiene. Sin embargo, la forma de nombrar archivos la determina el sistema de archivos del sistema operativo de la computadora; estas pautas están documentadas y se deben respetar, para que el archivo pueda ser reconocido por el sistema.

Todos los sistemas operativos admiten un nombre constituido por caracteres y es posible que se haga diferencia entre un nombre en minúscula y el mismo en mayúscula. Algunos sistemas operativos admiten una segunda parte para el nombre, como en el caso de Windows, que se denomina extensión.

En algunos casos esta extensión es una convención. Por ejemplo, todos los programas fuente de lenguaje C++ deben tener la extensión "**.**C++" para ser compilados. En otros la extensión sólo ayuda al usuario a recordar el tipo de archivo sin que esto sea información específica para el sistema operativo.

9.9.1.1 Estructura de archivos

En principio, los archivos se pueden estructurar de las siguientes maneras:

- Una cadena de bytes, cuyo significado depende del proceso que lo consulta;
- una agrupación de registros lógicos. Los archivos se dividen en unidades de acceso denominadas registros lógicos. Si se los compara con un fichero manual, cada ficha constituye una unidad de acceso y equivale, por lo tanto, a un registro lógico. El registro lógico está constituido por unidades elementales con sentido propio, por ejemplo,

el registro de un alumno en un archivo de datos personales contendrá el número de su DNI, el legajo y su apellido y nombre; estos datos "con sentido propio" se denominan campos. En cada operación de lectura o escritura (en inglés *read/write*) se accede a un registro lógico para recuperar o modificar un campo;

- una agrupación de archivos relacionados entre sí constituye una base de datos.

9.9.1.2 Acceso a archivos

Un proceso que puede leer todos los bytes o registros del archivo en orden, comenzando desde el principio, sin saltear los registros fuera de ese orden, se dice que es un archivo que se accede en forma secuencial.

Se dice que se accede al archivo en forma directa cuando un proceso puede leer bytes o registros de un archivo fuera de orden o puede acceder a los registros considerando uno de sus campos como una clave. Esta técnica involucra un acceso al azar (en inglés, *random*).

9.9.1.3 Atributos de un archivo

El sistema de archivos asocia a cada uno cierta información adicional, por ejemplo, el día y la hora en que se creó y su tamaño. Esta información extra se conoce con el nombre de *atributos del archivo*. La lista de atributos varía en grado considerable de un sistema operativo a otro. Las siguientes son algunas de las posibilidades:

- Protección: Determina quién y de qué manera se puede acceder al archivo;
- Sólo lectura: El archivo puede leerse, pero no modificarse;
- Escondido: El archivo existe pero no aparece en la lista del directorio o lista de archivos existentes;
- Sistema: Es un archivo de uso exclusivo del sistema operativo.

9.9.1.4 Operaciones sobre archivos

Los diferentes sistemas operativos permiten realizar operaciones para almacenar y recuperar la información en los archivos. Estas operaciones se ejecutan mediante llamadas al sistema (en inglés, *system calls*). En cada sistema operativo varía la especificación semántica del comando pero no su función:

- *Create* (crear): Permite crear el archivo sin datos para asignarle alguno de sus atributos.
- *Delete* (borrar): Permite borrar el archivo, y liberar el espacio ocupado en el soporte.
- *Open* (abrir): Todo archivo debe ser abierto por el proceso antes de ser leído o escrito. Esto permite que el sistema operativo busque los atributos y la lista de direcciones de disco. Es usual que durante la ejecución de una tarea esta información se almacene en la memoria principal en una tabla, para un acceso rápido hasta que el archivo se cierre.
- *Close* (cerrar): Todo archivo que deja de utilizarse debe cerrarse para liberar su tabla de acceso en memoria principal.
- *Read* (leer): Permite leer del archivo. La tarea que lee debe especificar cuánta información necesita, así como indicar un buffer, espacio en memoria principal, sobre el que desea colocar la información.
- *Write* (escribir): Permite escribir el archivo. Se debe indicar la posición de comienzo de escritura, teniendo en cuenta que si esa posición es intermedia en el archivo, los datos allí presentes se sobreescriben.

En la Web de apoyo se incorporaron programas fuente en Assembler 80x86, cuya lógica le permitirá comprender mejor todas las actividades necesarias que involucran algunas de las operaciones enunciadas.

- *Append*: Permite escribir el archivo sólo a partir del indicador de final de archivo, esto es, a partir del último byte grabado.
- *Rename* (renombrar): Se utiliza para cambiar el nombre actual del archivo existente.

La densidad de grabación en una unidad de disco rígido depende de cuán pequeña es la distancia del campo magnético (en inglés, *frame*) que representa un bit.

9.9.2 Sistema de archivos en *discos de tecnología magnética*

El sistema de archivos (en inglés, *file system*) es un conjunto de programas del sistema operativo cuyo objetivo principal es hacer transparente al usuario los detalles de la manera en que la información se almacena en archivos y se accede al soporte de información. La necesidad de almacenar información permanente en los soportes se debe a que:

- Todos los procesos en una computadora necesitan almacenar y recuperar información, y es común que ella deba sobrevivir a la terminación del proceso.
- Es necesario almacenar grandes volúmenes de información, por lo tanto, se justifica la necesidad del uso de memorias externas.
- Muchos procesos pueden necesitar el acceso a la misma información en forma simultánea, por lo tanto, cuando un proceso modifica cierta información, ésta debe estar protegida del acceso de otro.

9.9.3 Disco magnético desde el punto de vista lógico

Las características técnicas de estos dispositivos de almacenamiento se abordarán en el capítulo "Dispositivos de entrada/salida". Por el momento sólo nos referiremos a las partes físicas necesarias para comprender su parte lógica, objetivo de este capítulo.

En la actualidad los discos magnéticos están constituidos por platos rígidos, de allí su denominación *disco rígido*, para diferenciarlos de los ya extinguidos discos flexibles. Están dispuestos en unidades de cabezas fijas o móviles, en grupos de uno o más platos. Más allá de todos estos detalles, debemos considerar que un soporte en un disco está constituido por un plato de aluminio, recubierto de material magnetizable con forma de disco que gira alrededor de un eje dispuesto en la unidad, al que se accede por medio de una o más cabezas lectograbadoras.

La forma en que las cabezas recorren la superficie del **plato** describen círculos concéntricos denomindaos pistas (en inglés, *tracks*).

Una **pista** es una división lógica y no física de la superficie del soporte, que se produce por efecto de la acción de rotación del soporte y de la posición fija de la cabeza al momento de grabación o lectura. La cantidad de pistas en la superficie depende del tipo de unidad a la que está asociado el soporte, y del sistema de archivos que lo administre.

Si la unidad graba las dos caras de un plato o cuenta con más de uno, habrá muchas pistas de igual número (a la misma distancia respecto del eje) en las distintas caras. Podemos imaginarnos que por efecto de rotación se "visualizan" tantos **cilindros** concéntricos como pistas haya.

Por cada cara grabable habrá una cabeza lectograbadora. Todas ellas acceden de manera simultánea, como si fueran los dientes de un peine o **brazo actuador**, a un cilindro en particular, cuya identificación está dada por un número relativo que tiene que ver con la distancia del radio entre las cabezas y el eje. Por ejemplo, el cilindro 0 es el conjunto de pistas 0 de todas las caras grabables y es el cilindro más alejado del eje. Cuando la posición del brazo actuador está del lado externo del plato, éste es el cilindro al que más rápido se accede por estar más cerca de la posición inicial del brazo.

El concepto de cilindro (en inglés, *cylinder*) es importante, dado que la información se graba en sentido vertical, o sea que se completa un cilindro y luego se posiciona el brazo en el siguiente. Considere que cada movimiento de cabeza es mecánico y, por lo tanto, resulta entonces óptimo grabar todas las pistas de igual número sin modificar la posición del brazo que sostiene las cabezas.

Cada pista está dividida en sectores que se pueden identificar con un número, en general en las unidades de disco rígido el número de sectores por pista es fijo y, debido a que las pistas externas tienen una circunferencia mayor que las internas y una densidad menor.

El número de sectores que se pueden definir en la pista más interna limita el número de sectores por pista para todo el plato. Muchos discos usan una técnica llamada grabación de zona múltiple, para acomodar más datos en la superficie del disco. Esta técnica permite que el número de sectores por pista se ajuste para que se almacenen más sectores en las pistas externas que en las internas.

Si bien en la mayoría de los casos la unidad de referencia que hace el proceso para acceder a un archivo es el registro lógico, la unidad de acceso al soporte es una agrupación de ellos, denominada bloque. Ésta es la razón por la que los dispositivos de este tipo se conocen como dispositivos orientados al bloque, a diferencia de un teclado, que se denomina dispositivo orientado al carácter.

Como expresamos, en un disco por cada bloque se graban uno o varios sectores; cada sector es una porción de pista, y en términos del sistema operativo Windows un bloque se denomina *cluster*.

Como todos los sectores de una pista almacenan la misma cantidad de bits, y a su vez los sectores de pistas distintas observan el mismo criterio, se deduce que la **densidad** de grabación aumenta en forma sucesiva respecto de la distancia del eje central (cuanto más cerca del eje, mayor densidad de grabación).

La cantidad de bits por pulgada depende de las cabezas lectograbadoras, la calidad del medio de grabación y las revoluciones por minuto a las que gira el disco. La cantidad de pistas por pulgada depende de las cabezas lectograbadoras, la precisión mecánica con la que se pueden posicionar las cabezas y la aptitud del disco de girar formando un círculo muy definido.

Si indicamos que la cantidad de bits por sector se mantiene constante, resulta, entonces, que la cantidad de bits de un bloque, también. La determinación de la cantidad de información por sector o por bloque tiene relación con la capacidad total del volumen.

El sistema de archivos tiene la responsabilidad de aportar una estructura lógica que permita un reconocimiento ágil de la ubicación de la información a acceder. Estos programas se dedican a organizar y administrar la gestión de almacenamiento, permitiendo la captura de bits desde el disco. Podemos indicar que dar formato a un volumen permite delimitar el territorio sobre el soporte. Originalmente, un disco es un territorio sin límites. Se dice que un programa que registra en el soporte una estructura productiva realiza un proceso de "dar forma", y es conocido como formateo o inicialización del soporte.

Los parámetros que determinan la ubicación de una entidad para su acceso son el número de pista o cilindro, el número de cara y el número de sector. El formateo (en inglés, *format*) es el nombre con que se denomina al comando que ejecuta el programa de inicialización de un disco y que genera la grabación de marcas (en inglés, *labels*), que identifican estos parámetros. De este modo se puede localizar el lugar donde se graban los archivos. El proceso de formateo es, entonces, imprescindible para trabajar con el soporte.

Caso ejemplo: una forma de gestionar el acceso a un archivo

Cuando se ejecuta el comando que implica abrir un archivo existente, se determina dónde se encuentra éste en el disco. Para esto es necesaria la identificación de cilindro, cabeza y sector. Luego se transfiere esta información a la **controladora** de disco, que lleva al **motor del actuador**, y, por lo tanto, al **brazo actuador**, a posicionar las cabezas sobre la pista adecuada. A medida que el disco rota, la cabeza apropiada lee la dirección de cada sector en la pista. Cuando aparece el sector deseado bajo la cabeza lectograbadora, se lee todo el contenido del sector en forma secuencial. Esta lectura vuelca los datos necesarios en una memoria ultrarrápida especial, denominada **caché de disco**. Entonces, el controlador que actúa de interfaz de la unidad de disco envía la información necesaria a la memoria principal de la computadora. Para almacenar datos en una unidad de disco rígido, el procedimiento es similar a recuperarlos, sólo que inverso.

9.9.3.1 Particiones

Un disco rígido es una unidad física y, por lo tanto, escapa al alcance de este capítulo. Sin embargo, los sistemas operativos las consideran unidades lógicas. En un mismo disco físico se pueden crear dos o más unidades lógicas, a las que se denomina **partición**. Desde el punto de vista del software, cada partición es un disco lógico o unidad lógica diferente. La partición se crea cuando se da formato al disco, su capacidad es fija y está constituida por un grupo de cilindros contiguos. Una de las razones por las cuales se crea más de una partición, es para que en cada una se instale un sistema operativo distinto; de esta manera se crean dos entornos de trabajo diferentes para una misma computadora. De más está decir que el sistema se arranca desde una única partición por vez. En un mismo disco puede haber particiones primarias y secundarias. En una partición primaria se instala un sistema operativo, sólo una de ellas está activa por vez. Una partición secundaria se utiliza para almacenar archivos que en algunos casos pueden ser accedidos por distintos sistemas operativos, pero no contienen los programas de instalación.

Caso ejemplo: organización lógica de una partición

Consideremos la descripción de una estructura lógica en disco que nos permita visualizar los conceptos vigentes indicados hasta ahora. Sin embargo, como las computadoras pueden administrarse bajo distintos sistemas operativos, se mantuvieron algunas formas estándar comunes a todos ellos que serán las descriptas.

Es común que en una partición primaria de un disco se organice un territorio denominado **área de sistema**, cuya información se utiliza para la administración del soporte, y un **área de datos** para los archivos.

En el área de sistema se puede encontrar:

- Un sector de arranque (en inglés, *boot*).
- Una tabla índice que señala la locación inicial de cada archivo grabado, por ejemplo, una *file allocation table* o FAT.
- Una tabla de declaración de archivos, denominada directorio.

Cada sector se referencia internamente con la dirección formada por su referencia física: número de cilindro, cara y sector; no obstante, el sistema operativo identifica estos parámetros bajo un mismo número. Por ejemplo, cuando un *cluster* contiene un único sector, se puede decir que ese sector es el "*cluster* número N", que es lo mismo que identificar a un alumno por su número de documento o indicar que es el primero de la fila, o sea que se puede lograr una identificación con distintas referencias.

Un sector de arranque (en inglés, *boot sector*) es una unidad lógica donde se almacena una rutina de carga, cuya ejecución produce la carga en memoria principal o interna de otros programas del sistema operativo.

En el momento de formateo se indica si una partición puede inicializar el sistema operativo o no.

Además de la rutina de arranque, la información que se puede encontrar en el sector de arranque es la que sigue:

- Número de bytes por sector.
- Cantidad de sectores por *cluster*, la mayoría de las veces contiene más de uno.
- Cantidad de sectores reservados.
- Cantidad de FATS y de sectores que ocupa (para el caso de sistemas que organicen el espacio para el área de archivos de datos con esta técnica).
- Cantidad de entradas al nodo raíz, si se utiliza una organización de archivos tipo árbol.
- Cantidad total de sectores en la partición del disco.
- Técnica utilizada para la grabación de la información.
- Cantidad de sectores por pista.
- Cantidad de cabezas lectograbadoras.

Caso ejemplo: el sector de arranque de una computadora personal

Al encender la computadora, el sistema básico de E/S (programa alojado en memoria ROM) lee el "*master boot record*" situado en el cilindro 0, cara 0, sector *1* del disco rígido que contiene la rutina de arranque y la descripción de las particiones, en este caso son dos, una primaria y otra secundaria.

Una vez que se detecta la partición activa, el sistema básico de entrada/salida le transfiere el control a esa partición para que se ejecute la rutina, que en este caso carga los programas de Windows.

El sector maestro de arranque está constituido por dos partes básicas: el código de la rutina de arranque y 4 entradas de dieciséis bytes para la descripción de las cuatro particiones que se pueden crear como máximo.

Cada entrada, de *16 bytes*, describe características de una partición. El primer byte indica si la partición es activa o no, los tres bytes siguientes se utilizan para indicar el comienzo de la partición, un byte indica el sistema operativo que está instalado en la partición, tres bytes indican el final de la partición, cuatro bytes, el número de sectores anteriores al comienzo de la partición, y los últimos cuatro, el número de sectores totales.

Ejercicio 9:

Si los tres bytes que indican comienzo o fin de partición se interpretan de la siguiente manera: *1 byte* indica el número de cara; *2 bits* del segundo byte, el número de cilindro que se completa con los *8 bits* del último byte (en total diez bits), y los seis bits restantes del segundo byte indican el número de sector. Asimismo, considerando que la numeración de caras y cilindros comienza en 0 y la de los sectores en *1,* determine cómo se deduce dónde comienza la partición (cuántas caras, cilindros y sectores), si la combinación hexadecimal que obtiene es "*3F BF B0*"

9.9.3.2 FAT, file allocation system

Es una tabla de asignación de espacio para los archivos del área de datos. En cada entrada de la tabla se representa un *cluster* del área de datos. Así, la entrada *3* identifica al *cluster 3*, y esta entrada a su vez contiene un número que actúa de puntero a otro *cluster*. Si la entrada *3* contiene un *6*, significa que el archivo ocupa el *3* y continúa en el *6*. En este sistema un archivo está asociado a un *cluster* de comienzo, indicado en el directorio, como veremos más adelante.

Para mayor claridad, supongamos que en el directorio dice que el archivo comienza en el *cluster 2*; con ese número se accede a la entrada *2* de la FAT. La entrada señalada contiene el número de cluster donde continúa el archivo, o sea que el archivo continúa en el cluster *3*. Consultando la entrada *3,* se deduce que el archivo continúa en el cluster *6*. Así, las sucesivas entradas se encadenan hasta que el *cluster* final del archivo se marca con una entrada de la FAT que contiene todos unos. Si un *cluster* de datos está vacío, su entrada correspondiente en la FAT contiene el número 0.

3	*6*	*0000000*	*0000000*	*7*	*1111111*	*0000000*
Entrada 2	Entrada 3	Entrada 4	Entrada 5	Entrada 6	Entrada 7	Entrada 8

En realidad, éste es un esquema muy simplificado de la organización de la FAT. La función de la FAT es mostrar la cadena de *clusters* en la que se aloja el archivo. Inicialmente los archivos se almacenan en *clusters* sucesivos, pero cuando el usuario elimina ciertos archivos quedan huecos que podrán ser utilizados por otro nuevo. Es común que el hueco no tenga el tamaño suficiente como para alojar el archivo nuevo y, por lo tanto, se produzca un salto al próximo *cluster* vacío, de manera que permita alojar el resto de los datos que se desea almacenar.

Este procedimiento se repite una y otra vez, y los archivos quedan fraccionados en el soporte. El ejemplo de tabla muestra que el archivo que comienza en el *2*, continúa en el *3,* del *3* salta a *6* y finaliza en el *7*; o sea que ocupó *4 clusters*, y en el momento de su creación los *clusters 4* y *5* estarían ocupados, razón por la que del *3* salta al *6*. En este momento ambas entradas figuran con 0, por lo que el archivo que las ocupaba dejó el espacio libre. Si el próximo archivo a grabar ocupa *3 clusters* de dato, entonces la entrada *4* se completará con un *5* y la entrada *5* con un *8*, luego la *8* con 1, que indican fin de archivo.

Si bien es lógico y deseable que los archivos aprovechen los huecos vacíos del soporte, recuperar un archivo fraccionado en *clusters* ubicados en cilindros diferentes insume más tiempo, porque provoca el movimiento mecánico del brazo, por lo que es mejor que un archivo se aloje en *clusters* sucesivos. En este tipo de organizaciones se produce una reestructuración de la información que se conoce con el nombre de desfragmentación (en inglés, *defrag*).

9.9.3.3 El directorio raíz y los subdirectorios

El directorio es un archivo que se actualiza en relación con las operaciones que se realizan con los archivos. Es una tabla con información propia de los archivos y a cada uno se asigna una entrada que básicamente indica su nombre y atributos.

El área de archivos o área de datos está dividida en clusters de igual tamaño (que varía según el tamaño del disco).

Para acceder a un archivo guardado en un disco, se accede al directorio y se lo ubica por su nombre de archivo en el disco y un número del cluster de comienzo que permita su ubicación en el soporte y el acceso a la primera entrada de la FAT, para conseguir su cadena de locaciones.

Además de archivos, un directorio puede contener otros directorios, que dependen del primero; así se establece una organización jerárquica que se puede identificar como un árbol invertido. De este modo, se pueden separar archivos utilizados en aplicaciones distintas en directorios diferentes.

La información que se puede encontrar en una entrada se resume así:

- Nombre del archivo.
- Byte de atributos.
- Hora de última modificación.
- Fecha de última modificación.
- Número de *cluster* donde comienza el archivo.
- Tamaño del archivo.

Es importante detenerse en los atributos de archivo. Los archivos admiten operaciones como abrir, cerrar, leer, escribir, crear, borrar (en inglés, *open, close, read, write, create, delete*, respectivamente). Estas operaciones pueden ser reguladas por los atributos asociados al archivo. El estado de un atributo puede representarse con un bit, por ejemplo:

- *Read-Only*: El archivo puede abrirse, cerrarse y leerse, pero no puede borrarse ni escribirse.
- *Hidden:* El archivo existe pero no se muestra en las búsquedas normales dentro del directorio, y está oculto para búsquedas normales.
- *System:* El archivo pertenece al sistema operativo.
- *Volume:* Indica que este archivo identifica el soporte, o sea que es una cadena de caracteres que identifica la unidad lógica con un nombre simbólico.

El procedimiento de acceso requiere ciertas operaciones. Citemos, por ejemplo, la creación, una lectura o una grabación, o su eliminación del soporte.

Creación

Se accede al directorio comparando el nombre del archivo con cada una de sus entradas. Si el archivo existe, no podrá crearse, como consecuencia deberá existir la opción de reemplazo de uno por el otro, previa consulta al usuario. Si el archivo no existe, se le asigna una entrada al directorio y se busca en la FAT. La primer entrada con contenido 0 se asigna como primer cluster, se graba y se busca otra entrada vacia, asi se arma una cadena que puede estar físicamente discontigua hasta la finalización del archivo.

Lectura/grabación

Se accede al directorio comparando el nombre del archivo a leer o grabar. Si se quiere leer y no existe, provoca un error archivo no encontrado (en inglés, *file not found*); si existe, se accede a los *clusters* de datos consultando alternativamente la FAT.

Eliminación de un archivo

Se accede al directorio comparando el nombre del archivo con cada una de sus entradas. Si no se produce equiparamiento, se genera un mensaje del error; en caso contrario se marca la entrada del directorio indicando que ésta queda libre; a su vez se liberan las entradas de la FAT que le estaban asignadas. Es importante que le quede claro el concepto de que el archivo no se borra físicamente del soporte, sino que los territorios ocupados quedan liberados gracias a la actualización del directorio y las entradas de la FAT regrabadas con el número 0.

Lo más lógico en todos los soportes es que el área de sistema se aloje en el cilindro que permita el menor tiempo de acceso a las cabezas lectograbadoras, dado que éstas siempre son consultadas para acceder al soporte. Éste es el cilindro cuya distancia física de las cabezas es menor, mientras que los *clusters* de datos ocupan el resto de los cilindros.

Es importante recalcar que lo explicado hasta ahora es un ejemplo representativo de la organización lógica de un sistema de archivos específico. Otros soportes del mismo tipo se pueden organizar de manera distinta. El siguiente es otro ejemplo de organización del área de datos.

Los metadatos son datos estructurados que describen información. Esto es, son datos "acerca de otros datos". Un ejemplo puede ser: su identificación (área en la que está incluido), su calidad (nivel de precisión), su referencia (nivel de actualización).

9.9.3.4 Sistema de archivos NTFS

Como su nombre lo indica (en inglés, *new technology file system*), es un sistema de organización de archivos que ordena la información almacenada en el soporte de manera distinta a los que utilizan FAT, y es más eficiente para particiones de gran tamaño. Las características que los diferencian son:

- Compatibilidad mejorada con los metadatos.
- Permite la estructura de datos denominada árboles-B.
- Aprovecha mejor el espacio en disco.
- Reduce la fragmentación de archivos típica del sistema FAT.

Los metadatos son archivos creados en forma automática por el sistema cuando formatea un volumen de NTFS. Se localizan utilizando una tabla de archivo maestra (en inglés, *master file table*). Esta tabla es el punto de arranque de cualquier unidad bajo NTFS, y se la puede analogar a la FAT, pero es mucho más que una lista de *clusters* vacíos.

Por cada archivo o directorio que se crea en el disco, se genera un registro en la MFT, que puede alcanzar un tamaño de *2 KB*. Su función es almacenar información de los archivos o directorios con la forma de atributos, que dan la característica de "datos acerca de datos". Si el archivo es tan pequeño que los datos y los atributos de los datos entran en un registro, queda almacenado en la MFT y no requiere espacio adicional en el área de datos.

Cuando se agrega un registro más al MTF, que ocupa un lugar denominado zona de MTF y que se crea cuando se da formato a la partición, se calcula un 10% más de la capacidad total del disco, a sólo efecto de agregar nuevos registros. Si este porcentaje no alcanza, se puede fragmentar la MTF y continuar su estructura en otra zona de disco.

La descripción de los registros de la tabla nos ayudará a comprender su funcionalidad.

Nombre en inglés del archivo Metadata	*Registro de la MFT*	*Descripción*
Master File Table (MFT)	0	El primer registro contiene datos acerca de la propia MTF.
Master File Table 2 (MFT2) or Master File Table Mirror	1	Es una copia de seguridad de la MTF que se almacena generalmente al final de la partición o en el medio.
Log File	2	Un registro de transacciones producidas en el volumen para que pueda ser recuperado en caso de error.
Volume Descriptor	3	Contiene la información propia de la partición .
Attribute Definition Table	4	Contiene el significado de cada atributo.
Root Directory/Folder	5	Éste es el puntero al directorio raíz del volumen.
Cluster Allocation Bitmap	6	Contiene una tabla de clusters ocupados y vacíos.
Volume Boot Code	7	Contiene una copia de la rutina de arranque.
Bad Cluster File	8	Contiene la tabla de clusters dañados para que no se utilicen.
Quota Table	9	Tabla de cuotas asignadas. Una cuota es una cantidad de espacio en disco que se le puede asignar a un peticionario.
Upper Case Table	10	Tabla que contiene la información para convertir los nombres de archivos a código UNICODE para que sean compatibles a nivel internacional.

9.9.4 Buffers y cachés de disco

Se denomina buffer de disco a una memoria de almacenamiento temporal en la que se aloja un bloque de información del disco. En un proceso de lectura, una vez que la cabeza recorre un sector, lo aloja en el *buffer* del controlador y luego lo envía a un *buffer* dispuesto en la memoria principal, donde concluye la tarea del controlador. En un proceso de escritura la información se envía de memoria principal al *buffer* del controlador para luego grabarlo en disco. En la memoria principal se pueden alojar varios sectores, por lo tanto, es factible mantener tantos *buffers* de memoria principal como hayan sido definidos en la configuración del sistema. Esto agiliza los tiempos de búsqueda, ya que es más rápido buscar en memoria principal que acceder a disco de manera continua. Para optimizar aún más los accesos a disco, se puede asociar al sistema un caché de disco. Esto es, un *buffer* especial dotado de un algoritmo que permite determinar qué sectores son los solicitados con mayor frecuencia, para retenerlos y agilizar los tiempos de búsqueda. El caché puede estar alojado en el controlador de la unidad de disco (caché hardware), o bien, controlado por un programa de gestión del sistema operativo en memoria principal (caché software).

Ejercicio 10:

En su explorador puede encontrar la información relacionada con "guardar" lo más recientemente utilizado. Búsquelo e indique qué dice cuando consulta la ayuda por el contenido de "caché de página Web".

9.9.5 Discos virtuales

Ram disk o *virtual disk* son términos equivalentes que pueden llegar a confundirlo con otros conceptos que incluyan las palabras RAM o virtual. Esta facilidad no tiene la envergadura suficiente como para tratar el tema con mucho detalle y es, más bien, del ámbito de estudio de algunas extensiones de memoria en las computadoras personales. Si se considera que un acceso a disco puede tardar algunos microsegundos y que, a veces, un proceso depende de él para continuar su ejecución, sería interesante crear un disco ficticio en una parte de memoria RAM, de manera que se emule su actividad con la consecuente ganancia de tiempo. Se debe proveer un programa que haga posible la creación y el mantenimiento de este disco virtual. El programa debe generar el área de arranque y el directorio, así como la organización lógica que asumen los discos; o sea que se debe simular su funcionamiento. Se puede simular un disco duro usando como medio de almacenamiento la memoria interna; éste debe residir en memoria para atender los requerimientos que una aplicación pretende de este supuesto disco, interpretando los comandos típicos que se utilizan en la memoria externa.

Se puede simular un disco duro empleando un servicio de almacenamiento *online* que ofrece varios GB como si se tratara de un disco duro local.

9.9.6 Sistema de archivos en discos de tecnología óptica

El primer sistema de archivos creado para medios ópticos denominados CD-ROM, esto es, que no aceptaban reescritura, fue especificado en la norma *ISO 9660*. Se trataba del primero que soportaba archivos para diferentes sistemas operativos. En tecnología óptica ésta es la característica más buscada, puesto que en general se utiliza el medio no sólo en distintas computadoras con sistemas operativos diferentes sino también en múltiples dispositivos digitales. En la actualidad el formato más utilizado es el UDF, que es la implementación de la norma ISO/*IEC 13346*, que mejora la anterior, ya que permite la grabación en medios ópticos. Sus siglas significan *formato de discos universal*, y permite el acceso a los soportes ópticos desde diferentes sistemas operativos. Es pues obligatorio para DVD y admitido por los CD. Como característica principal podemos indicar que admite la grabación por paquetes, aprovecha mejor el espacio de almacenamiento del disco compacto y mantiene sus atributos originales.

Resumen

En este capítulo hemos visto la memoria de una computadora desde el punto de vista de los componentes y desde el de la administración del sistema operativo. En cuanto a las distintas jerarquías en el subsistema de memoria, hemos pasado revista a las tecnologías disponibles para la implementación física, tanto de memorias volátiles como de memorias no volátiles, considerando como parámetros para establecer los distintos niveles, los tiempos de respuesta, la capacidad de almacenamiento y/o el costo relativo del almacenamiento de un bit. Hemos abordado las razones que justifican el uso generalizado de la memoria dinámica, como la actual tecnología de preferencia de tipo RAM en la memoria principal y la importancia de tener niveles de memoria *caché* para disminuir los accesos a ella. Esto sugiere que la velocidad con la que el procesador disponga de instrucciones y datos es de suma importancia, pues el factor disponibilidad incide en su rendimiento en cuanto a velocidad de ejecución. Respecto del tiempo de acceso es razo-

nable pensar que una memoria más veloz siempre mejora el rendimiento global, sin embargo en un sistema real, el aumento o disminución de la prestación no siempre se da en la misma proporción que el aumento de velocidad de las memorias que lo componen. Las innumerables tablas de comparación de tiempos de acceso de los fabricantes de *chips* hacen referencia a la disponibilidad de información, desde que el componente recibe una orden de lectura hasta que se estabiliza su contenido en el *bus* de datos, y no a otros factores que inciden en su relación con los demás componentes del sistema. En el tiempo de acceso a un *chip* DRAM no se contempla el tráfico del *bus*, que depende de otras condiciones del sistema, y que se relacionan con el acceso a la memoria principal, pues influye en la dinámica de obtener bits. Considere que el procesador no es el único que está accediendo a memoria principal y sus accesos están limitados por el uso del bus en los ciclos de E/S. Estos ciclos se utilizan para transferir información entre la memoria principal y los dispositivos externos, por eso, el manejo de las interrupciones hardware así como el detalle de cómo se lleva a cabo una transferencia influye en el tráfico del bus que asocia ambas unidades. Estas particularidades serán descriptas en el capítulo "Dispositivos de Entrada/Salida", que considera sólo medios de almacenamiento masivo, y en el capítulo "Transferencia de Información" en el cual no sólo se describen los distintos modos de transferencia sino también las interfaces hardware que se utilizan para la conexión de componentes.