

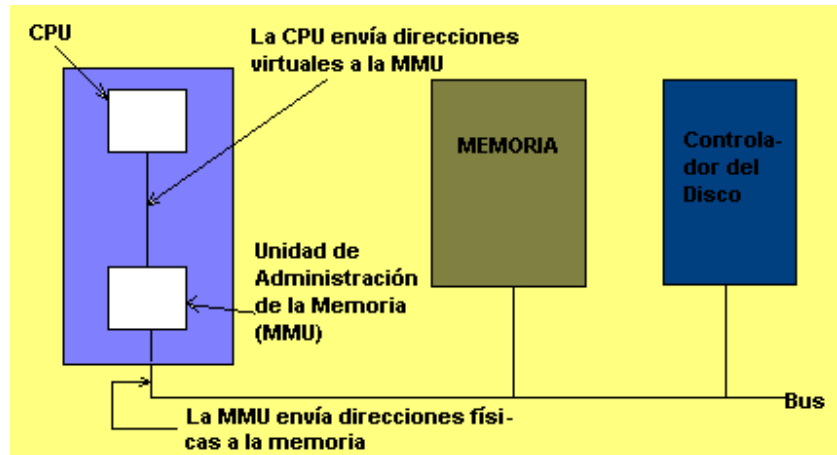
Memoria Virtual

Generalidades

En tiempos pasados cuando un programa era más grande que el tamaño de la memoria, este debía dividirse en módulos o partes e irse cargando a medida que se iban necesitando (recubrimientos). Este trabajo correspondía al programador y esto además de ser un trabajo engorroso implicaba pérdida de tiempo. Posteriormente se creó un mecanismo para encargarle este trabajo en su totalidad al sistema.

El método diseñado se conoce como **MEMORIA VIRTUAL**. La idea fundamental detrás de la memoria virtual es que el tamaño combinado del programa, los datos y lapila de ejecución puede exceder la cantidad de memoria real disponible para él. El sistema operativo mantiene aquellas partes del programa que se están utilizando en cada momento en la memoria principal y el resto permanece en el disco. En la medida en que se vayan necesitando nuevas partes estas se intercambian con las residentes en la memoria principal.

En los sistemas que utilizan memoria virtual, todas las direcciones son virtuales y el conjunto de todas ellas conforman el espacio (hueco) de direcciones virtuales. En los computadores donde no hay memoria virtual, la dirección se coloca directamente sobre el bus de la memoria, lo que permite que se pueda acceder a la palabra de la memoria física que tenga tal dirección. Al utilizar memoria virtual, las direcciones no pasan directamente al bus de memoria, sino que van a una **unidad de administración de la memoria (MMU)**, un conjunto de chips, que asocian las direcciones virtuales con las direcciones de memoria física.



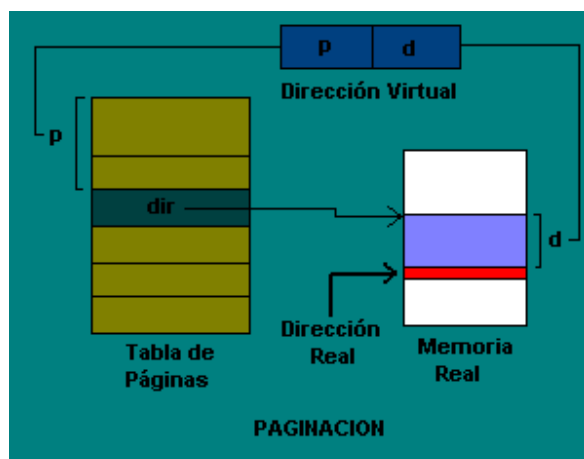
La idea central de la memoria virtual es que dirección es diferente de localización física, por ello se hace necesario tomar las direcciones virtuales y convertirlas a reales para poder tener acceso a la posición de memoria correspondiente.

Mecanismos de Traducción de Direcciones.

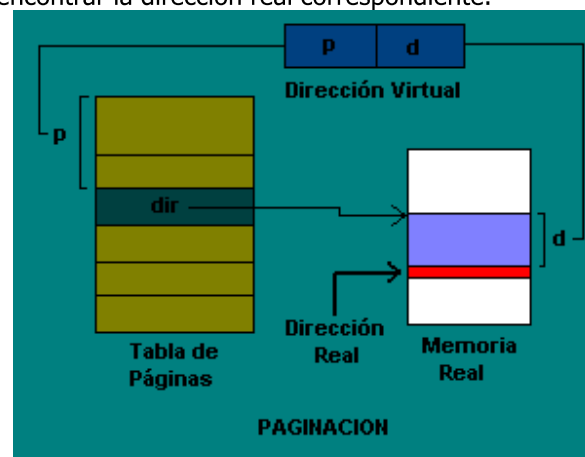
Para implantar la memoria virtual se hace necesario contar con algún método que permita traducir direcciones virtuales a direcciones reales. Estos métodos definen la forma como se implementará la memoria virtual y entre ellos tenemos:

<http://doblev.wordpress.com>

Paginación

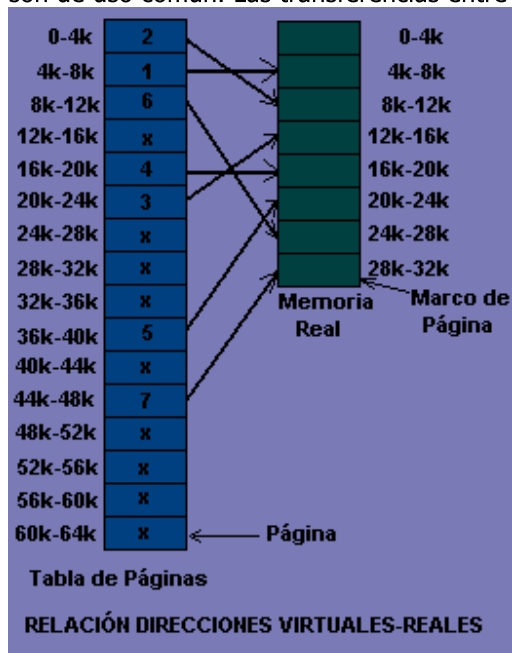


En este método la memoria se divide en bloques de igual longitud denominados páginas. La dirección virtual se compone de dos partes: una página y un desplazamiento. La primera sirve para encontrar la entrada correspondiente en la tabla de páginas, en donde se encuentra la dirección del marco de página correspondiente. A esta se le suma el desplazamiento para encontrar la dirección real correspondiente.



Como ejemplo, suponga que un computador puede generar direcciones virtuales de 16 bits que van desde 0 hasta 640K. Suponga además que la máquina sólo tiene 32K de memoria real. El espacio de direcciones virtuales se divide

en unidades denominadas páginas y las unidades correspondientes en memoria real, marcos de página. Las páginas y los marcos siempre tienen el mismo tamaño. En el ejemplo son de 4K, pero tamaños desde 512 bytes hasta 8K son de uso común. Las transferencias entre la memoria real y el disco son siempre unidades de página.



Por ejemplo, si el programa referencia la dirección 0 virtual, esta se envía a la MMU. La MMU ve que esta dirección cae en la página 0 (0-4095), la cual de acuerdo a la tabla de páginas está en el marco 2 (8192-12287). Transforma entonces la dirección en 8192 y manda esta al bus. La memoria no sabe que acerca de la MMU y sólo ve una solicitud de acceso a la posición 8192, la que atiende.

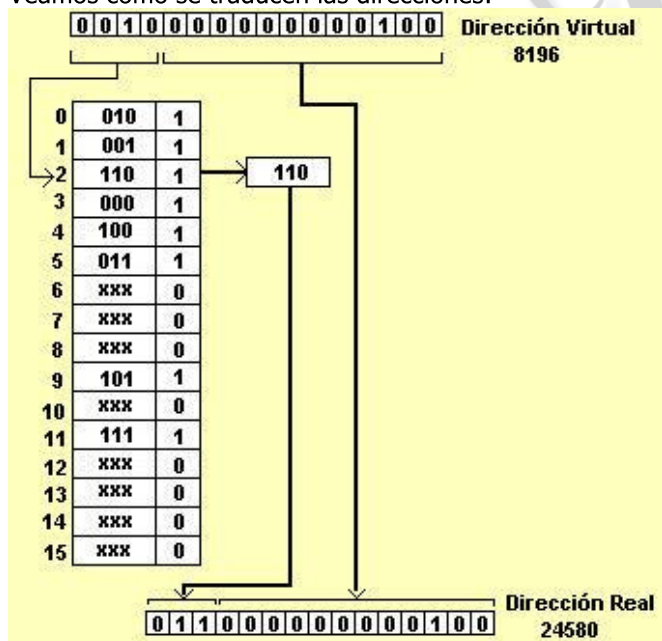
La dirección virtual 21500, que está a 20 bytes del inicio de la página virtual 5 (20480-24575) está asociada a la dirección real $12288+20=12308$.

Dado que la memoria virtual tiene 16 páginas y la memoria real 8 marcos de página, en un momento dado sólo se podrán albergar en ésta última 8 páginas. Es por ello que 8 entradas en la tabla de páginas están marcadas con x, indicando que ellas no residen en la memoria real.

Si se referencia la dirección virtual 32780 que está en la página 8 (12 bytes abajo de 32768). Como se ve en la gráfica, la entrada correspondiente está marcada con x, lo que indica que ella no reside en la memoria real. En este caso se presenta una interrupción denominada **defecto o fallo de página**, entonces el sistema operativo toma el control, elige un marco de página y lo "baja" de la memoria real. Después trae la página correspondiente a la dirección referenciada y la ubica en el marco liberado, modifica las entradas correspondientes en la tabla de páginas y reinicia el proceso.

Si en el ejemplo, el sistema operativo decidió liberar el marco 1, carga la página virtual 8 en la dirección física 4K y hará dos modificaciones a la tabla de página. En la entrada 1 quedará una x, indicando que está página ya no reside en la memoria real y en la entrada 8 quedará un 1 indicando que está página reside en tal marco. De esta forma la dirección virtual 32780, será traducida como 4108 real.

Veamos cómo se traducen las direcciones:



La dirección virtual 8196 (00100000000100 en binario), está dividida en 2 partes: los 4 bits más significativos es el número de la entrada correspondiente a esa página en la tabla (0010, 2 en decimal, para el ejemplo). Con este valor se indiza en la tabla de páginas, allí observa que el bit de presencia está encendido, lo que indica que la página reside en memoria principal. La dirección del marco es 011 (ver figura), que conjuntamente con el desplazamiento (00000000100) dan el valor de la dirección real 01100000000100, 24580 en decimal.

La dirección virtual, entonces, está compuesta de dos partes: un valor correspondiente a la entrada en la tabla de páginas y un desplazamiento.

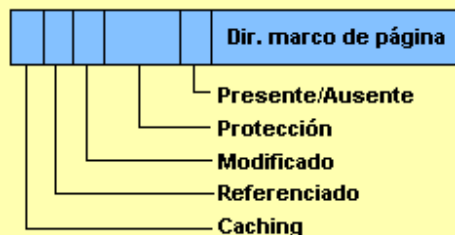
La finalidad de la tabla de páginas es asociar páginas virtuales con marcos de página.

La entrada típica en una tabla de páginas es la siguiente:

La dirección del marco de página, guarda la dirección en memoria real donde se encuentra la página.

El bit presente/ausente, indica si la página reside en la memoria real o no. Si la página no está presente (0) y se referencia, se producirá un [defecto de página](#).

Los bits de protección indican el tipo de acceso permitido. Estos generalmente se conocen como **rwX** y dependiendo de si están encendidos, la página se podrá leer (**r**), escribir (**w**) o ejecutar (**x**).



ENTRADA TÍPICA EN UNA TABLA DE PAGINAS

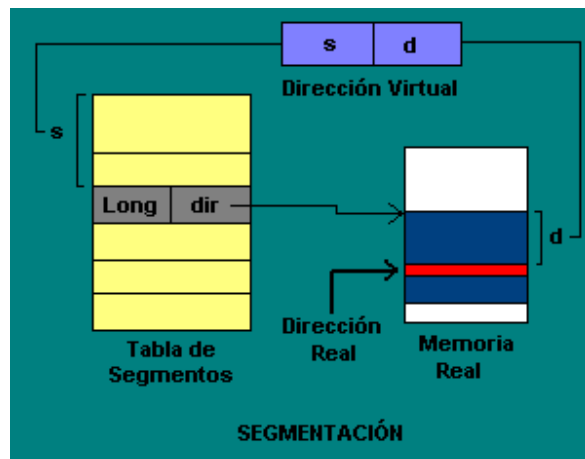
El bit de modificación se enciende cada que la página sufre algún cambio en memoria real. Se utiliza cuando la página es seleccionada para salir de la memoria real, cuando se presenta un [defecto de página](#). Si el bit está encendido, entonces la página se escribirá sobre el disco. En caso contrario sólo se desecha pues la imagen en disco es igual a la existente en la memoria real.

El bit de referencia como su nombre lo indica se enciende cada vez que la página es referenciada, bien sea para lectura o escritura. Sirve para apoyar al sistema operativo, para seleccionar la página a salir cuando se presenta un [defecto de página](#). Las páginas no referenciadas son unas muy buenas candidatas a salir. [Ver algoritmos de reemplazo](#).

El bit de caching. Si está desactivado, cuando se produzca una operación de E/S, el sistema operativo busca la información en el hardware y no una copia antigua en el caché.

Segmentación

En este método la memoria se divide en bloques de longitud variable denominados **segmentos**. La dirección virtual se compone de dos partes: un segmento y un desplazamiento. La primera sirve para encontrar la entrada correspondiente en la **tabla de segmentos**, en donde se encuentra la dirección del marco de segmento correspondiente. A esta se le suma el desplazamiento para encontrar la dirección real correspondiente.



Dentro de cada entrada de la tabla de segmentos se encuentra un campo (Long) que indica la longitud del segmento. Este se utiliza para verificar que no se hagan referencias fuera del segmento. Si esto llegase a ocurrir se generaría una [interrupción](#).

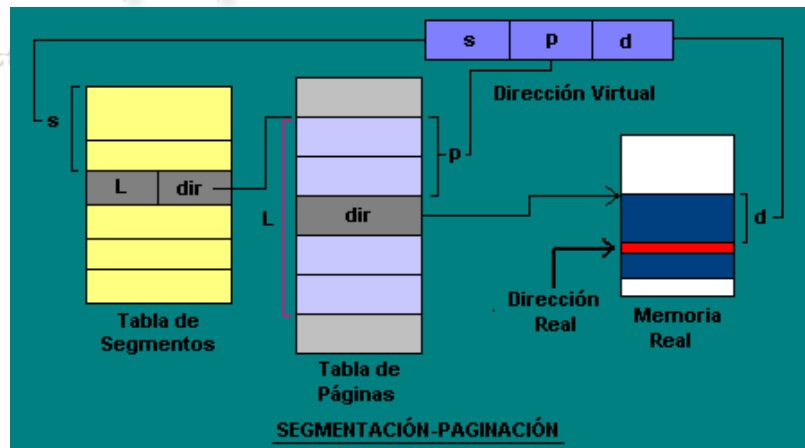
Las ventajas de la segmentación es que evita la [fragmentación interna](#), ya que los segmentos se definen del tamaño que se requiera y que facilita la protección de la memoria.

La principal desventaja es que conduce a [fragmentación externa](#), que es un fenómeno que presenta cuando se manejan

[bloques de longitud variable](#).

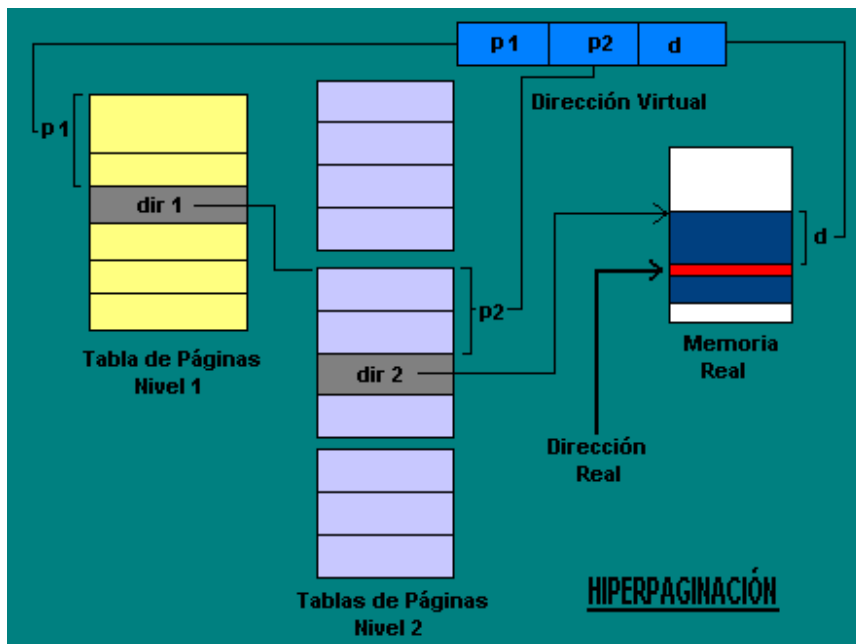
Segmentación-Paginación

En este método las direcciones virtuales se componen de tres partes: un segmento, una página y un desplazamiento (offset). La primera sirve para encontrar la entrada en la tabla de segmentos en donde se encuentra la dirección donde comienza la página, con la segunda componente se encuentra la entrada correspondiente a la página y allí la dirección del marco de página que junto con el desplazamiento dan la dirección real. Una ventaja de este método es que permite mantener una estructura de segmentación sin conducir a [fragmentación externa](#). Además permite mantener las tablas de páginas en memoria auxiliar.



Paginación de varios niveles o Hiperpaginación.

Para evitar el problema de tener unas tablas de páginas muy grandes en la memoria todo el tiempo, algunos sistemas utilizan tablas con varios niveles o hiperpaginación.



La idea central de las tablas de páginas de varios niveles consiste en evitar mantener todas las tablas en la memoria todo el tiempo. Aquellas que no sean necesarias no deben mantenerse allí.

La dirección virtual ahora se compone de tres campos: dos direcciones de páginas y un desplazamiento. Cuando una dirección virtual llega a la MMU, esta extrae primero el componente **p1** y lo utiliza como índice en la tabla de páginas de nivel 1. Allí ubica la dirección donde se encuentra la dirección de inicio de otra página en la tabla de páginas de nivel 2 y con la componente **p2** de la dirección encuentra la dirección del marco que contiene la página respectiva. Luego con la tercera componente de la dirección, el desplazamiento **d**, encuentra la dirección real.

El sistema de dos tablas, puede aumentarse a tres o más, pero ello implica que la complejidad del sistema se eleve de manera considerable, por lo que no se acostumbra más de tres niveles.

Memoria Asociativa

La memoria asociativa se basa en el **Principio de Localidad** que establece que la mayoría de programas tiende a referenciar un porcentaje reducido de sus páginas durante períodos relativamente largos de tiempo. Con base en este principio se equipa a los computadores de un dispositivo de hardware para asociar las direcciones virtuales con las direcciones reales (físicas) sin tener que recurrir a la tabla de páginas. Este se encuentra en la **MMU** y consta de un número pequeño de entradas (8, 16, 32 son cantidades típicas), cada una de las cuales tiene la estructura similar a una entrada en una tabla de páginas convencional.

La memoria asociativa funciona de la siguiente forma: cuando se presenta una dirección virtual a la **MMU** para su traducción, se verifica primero si su número de página virtual se encuentra en la memoria asociativa, al comparar todos los registros que la componen en paralelo. Si se encuentra allí, la dirección del marco de página se selecciona de allí directamente sin ir a la tabla de páginas y se continúa con el proceso de traducción normal. Cuando el número de página no está en la memoria asociativa, se recurre a la tabla de páginas. Una vez localizada la entrada correspondiente, se extrae una entrada de la memoria asociativa y se reemplaza con el dato determinado en la tabla de páginas. Así, si esta página vuelve a ser referenciada, la segunda vez será encontrada con rapidez. Con esto la **proporción de encuentros**, es decir, la proporción de referencias a la memoria que pueden ser satisfechas a partir de la memoria asociativa, se incrementa considerablemente, permitiendo una mayor eficiencia en el proceso de traducción de direcciones.

Espacios Múltiples y Espacio Único

Cuando se tiene un **Único Espacio Virtual** para todos los usuarios. En este caso se requiere una sola tabla de páginas, además los accesos entre usuarios son más sencillos de realizar pues se direcciona un mismo espacio. Sin embargo tiene el inconveniente de seguridad entre usuarios y el sistema operativo, lo mismo que asignación, la cual se puede realizar por los métodos mencionados anteriormente: **zonas de longitud fija** y **zonas de longitud variable**.

Cuando se tienen **Espacios Múltiples** cada proceso tiene asociado un espacio virtual y por lo tanto una tabla de páginas. Esto implica más espacio para guardar las tablas de páginas y dificulta la comunicación entre procesos, pero las labores de protección son casi que automáticas.

Si el sistema operativo no aparece en cada uno de los espacios virtuales, cada uno de ellos puede considerarse una extensión del hardware pues es equivalente a contar con una máquina abstracta pero sin sistema operacional. La mayoría de sistemas recurre a colocar parte del sistema operativo en cada uno de los espacios del usuario.

Algoritmos de Reemplazo

Cuando ocurre un [defecto de página](#), el sistema operativo debe seleccionar una página para retirarla de la memoria real y así dejar un marco vacío que contendrá a la página referenciada. Si el marco seleccionado contiene una página que haya sido modificada, esta deberá ser escrita de nuevo al disco. Por el contrario, si no se ha modificado, la página a cargar reemplazará a la existente en memoria real sin necesidad de reescribirla, pues su imagen en disco es idéntica a la existente en memoria principal.

A pesar de que se puede elegir una página al azar, es mejor seleccionar una de poco uso para favorecer el [principio de localidad](#).

El Principio de Optimalidad dice que el mejor algoritmo de reemplazo será aquel que seleccione la página que será referenciada más tarde en el tiempo. A pesar de ser muy fácil de enunciar, este principio es imposible de implementar pues para ello se requeriría saber de antemano, el orden en el que van a ser referenciadas las páginas. Dentro de los algoritmos realizables, los más conocidos son los siguientes:

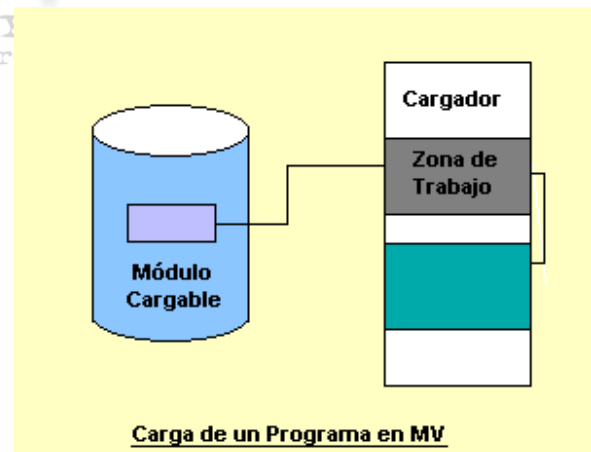
- **Aleatorio o al azar:** No respeta el [principio de localidad](#).
- **FIFO (First Input First Output):** reemplaza las páginas en el orden de llegada. No respeta el [principio de localidad](#).
- **LFU (Least Frequently Used):** Elige la página menos frecuentemente usada.
- **MRU (Most Recently Used):** Selecciona la página más recientemente referenciada. A pesar de no respetar el [principio de localidad](#), puede comportarse bien en sistemas interactivos, en donde existen períodos largos de tiempo mientras el usuario responde al sistema.
- **LRU (Least Recently Used):** Elige la página menos recientemente usada. Se basa en el [principio de localidad](#).

Este último es uno de los más utilizados a pesar de que para su implementación se requieren mecanismos sofisticados, ya que para poder llevarlo a cabo, se requeriría guardar para cada página la hora exacta en la cual ocurrió la última referencia, lo cual implicaría ampliar el tamaño de la entrada en la tabla de páginas, lo que resulta muy costoso. Por esta razón, se recurre a una aproximación que consiste en que cada vez que la página es referenciada, el [bit de referencia](#) se prende en la entrada correspondiente en la tabla de páginas. Periódicamente, un proceso demonio (un proceso background del sistema operativo), se encarga de apagar tal bit en aquellas páginas que lo tengan prendido. Cuando se necesita un marco, se selecciona una de aquellos que contengan páginas que tengan el bit apagado

Carga de un programa en Memoria Virtual

Para cargar un programa en memoria virtual este primero se mueve a una zona de módulos cargables en el espacio auxiliar y creando las correspondientes entradas en la tabla de páginas, las cuales deben quedar con el bit presente/ausente apagado, lo que implica que las páginas aún no están en la memoria real, a pesar de que el programa ya está cargado en la memoria virtual. Después de esto se deja todo el trabajo al mecanismo de tratamiento de los [defectos de página](#) para que este se encargue de traer a memoria real las páginas que vayan siendo referenciadas. De esta forma se carga un programa sin necesidad de realizar [entradas/salidas](#).

Este mecanismo recibe el nombre de acoplamiento y también se utiliza para realizar operaciones de E/S.



Control del Rendimiento del Sistema

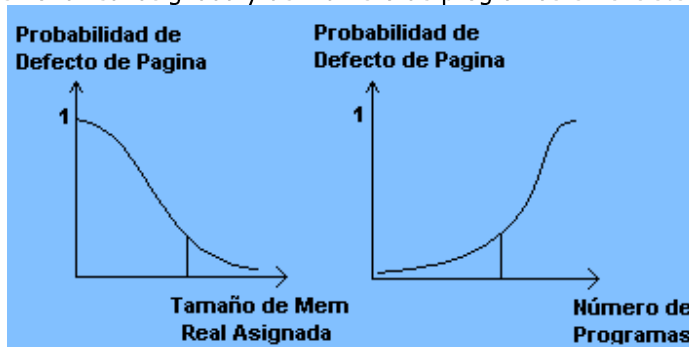
Uno de los aspectos más importantes en un sistema con memoria virtual es el control del rendimiento y de su tratamiento depende en gran medida su **desempeño**. Este último puede verse perjudicado por dos razones:

- El aumento considerable del número de transferencias de página, lo que conlleva un aumento significativo de la sobrecarga (overhead) del sistema.
- Se puede presentar el caso de que se le retiren páginas a un programa mientras espera por otra.

Estos fenómenos se pueden ver gráficamente:



Si el número de marcos de página asignadas a un proceso es pequeño, la cantidad de [defectos de página](#) se incrementa considerablemente. Existe además, un número crítico de página, marcado con **a** en la gráfica, a partir del cual, si se asignan menos marcos de página, [los defectos](#) se incrementan considerablemente. En las siguientes gráficas se ilustra el fenómeno, mediante la relación de la probabilidad de tener un defecto de página y el tamaño de la memoria real asignada y del número de programas en el sistema.



En los gráficos anteriores existe un punto a partir del cual el desempeño del sistema se degrada considerablemente. Este punto corresponde a una cierta cantidad de memoria real asignada al proceso. Si se le asigna menos, se generará una gran cantidad de [defectos de página](#), en cuyo caso se dice que el sistema está en **trashing**.

El fenómeno de trashing se puede representar analíticamente mediante el porcentaje de utilización del procesador (**PUP**).

Tiempo de Ejecución

$$PUP = \frac{\text{Tiempo de Ejecución}}{\text{Tiempo de Ejecución} + \text{Tiempo de Espera}}$$

Tiempo de Ejecución + Tiempo de Espera

El Tiempo de Espera (**TE**), en promedio es igual al número promedio de defectos de página (**PDP**) multiplicado por tiempo de transferencia de una página (**t**).

El valor de **PDP** puede calcularse multiplicando el tiempo de ejecución (**TEjec**) por la probabilidad de que ocurra un defecto de página (**p**). En resumen se tiene:

$$PUP = \frac{TE_{\text{jec}}}{TE_{\text{jec}} + TE}$$

$$PUP = \frac{TE_{\text{jec}}}{TE_{\text{jec}} + TE_{\text{jec}} * p * t}$$

$$PUP = \frac{1}{1 + p * t}$$

Según lo anterior el desempeño del sistema, depende de dos factores: el tiempo de transferencia de una página y la probabilidad de que ocurra un defecto de página. Para lograr que el porcentaje de utilización del procesador (**PUP**), llegue al 100% se debe procurar que la fracción $p \cdot t$, tienda a cero y para ello se debe procurar que ambos términos sean lo más pequeños posibles.

Para disminuir el valor de t (tiempo de transferencia de una página), se puede recurrir a obtener dispositivos periféricos más rápidos.

Para disminuir la probabilidad de que ocurra un defecto de página, se debe asignar más memoria real a cada proceso (ver gráfico) y para ello:

- Se debe aumentar la capacidad de la memoria real, o
- Se debe disminuir el número de procesos en el sistema

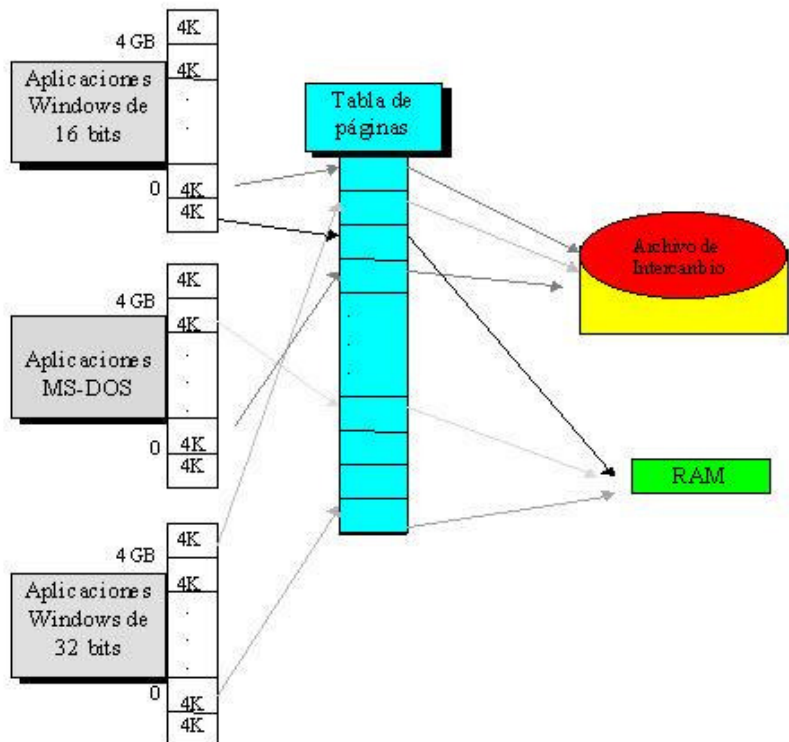
Gestión de la memoria virtual en Windows 95.

Generalidades

En Windows 95 es posible hacer que la memoria disponible en la PC sea mayor que la RAM física que tiene instalada.

El procesador 386 introdujo la posibilidad de asignar cualquier ubicación de la memoria a cualquier dirección. Windows 95 virtualiza la memoria de la misma forma que virtualiza las máquinas y los controladores. Esta virtualización de la memoria permite que las aplicaciones se comporten como si cada una tuviera su propia RAM física.

A cada aplicación se le asigna un espacio de direcciones virtuales exclusivo, que es el conjunto de direcciones que la aplicación puede utilizar. A cada aplicación de 32 bits y a cada aplicación basada en MS-DOS se les asigna su propio espacio de direcciones virtuales. Las aplicaciones acceden a la memoria a través de las direcciones virtuales, que el administrador de la memoria de Windows 95 asigna a direcciones físicas. Estas direcciones físicas pueden señalar ubicaciones de la RAM o del disco duro.



Direcciones de memoria asignadas entre ubicaciones físicas y virtuales

El código del programa y los datos que se encuentran en la memoria física pueden trasladarse a un archivo de intercambio creado para ese fin en el disco duro.

El archivo de intercambio cambia de tamaño según sean las necesidades del sistema. Si el espacio en la unidad de disco duro empieza a escasear, se disminuye el tamaño del archivo de intercambio. Si hay espacio disponible en el disco duro y se necesita más memoria, se aumenta el tamaño del archivo de intercambio, como se muestra en la Figura.

Cada espacio de direcciones está dividido en 1.048.576 (220) páginas y cada página tiene un tamaño de 4KB. La memoria física se asigna página a página. Una página puede indicar memoria que se encuentra en algún lugar de la RAM (el sistema puede cambiar la ubicación real) o en un archivo de intercambio de un disco, o bien puede estar marcada como no utilizada. En este último caso, la dirección existe, pero no se le ha asignado memoria física.

El Administrador de memoria virtual asigna las direcciones virtuales del espacio de direcciones del proceso a las páginas físicas de la memoria o del archivo de intercambio del equipo, con lo cual oculta a la aplicación la

organización física de la memoria. De esta forma se asegura que las [hebras](#) puedan tener acceso a la memoria del proceso al que pertenecen a medida que la necesitan, pero no a la memoria de otros procesos.

El administrador de memoria virtual de Windows controla todo el proceso de intercambio que tiene lugar en el disco. Lleva a cabo la [paginación](#) y mantiene una lista de las páginas de 4KB que se encuentran en ese momento en la memoria física accediendo a una tabla de páginas. Dicha tabla indica a Windows qué páginas se han trasladado al disco, cuáles pertenecen a cada proceso, etc.

Windows 95 establece varias nuevas exigencias en el gestor de memoria virtual:

- El nuevo tipo de aplicaciones Win32 con muchas nuevas funciones de la API que admiten varios tipos diferentes de memoria de aplicación compartida y asignada dinámicamente.
- Componentes del sistema [cargables dinámicamente](#).

Todas estas demandas requieren cambios en el gestor de memoria virtual de 32 bits en modo protegido, aunque no se requieren cambios para las antiguas aplicaciones Win16.

Direcciones virtuales en Windows 95

Windows 95 asigna a cada espacio de memoria virtual 4 GB de direcciones de memoria. Se trata sólo de espacio de direcciones; no se necesitan 4 GB de memoria física. En la Figura se muestra la forma cómo se realiza la asignación de espacio de dirección de memoria virtual .

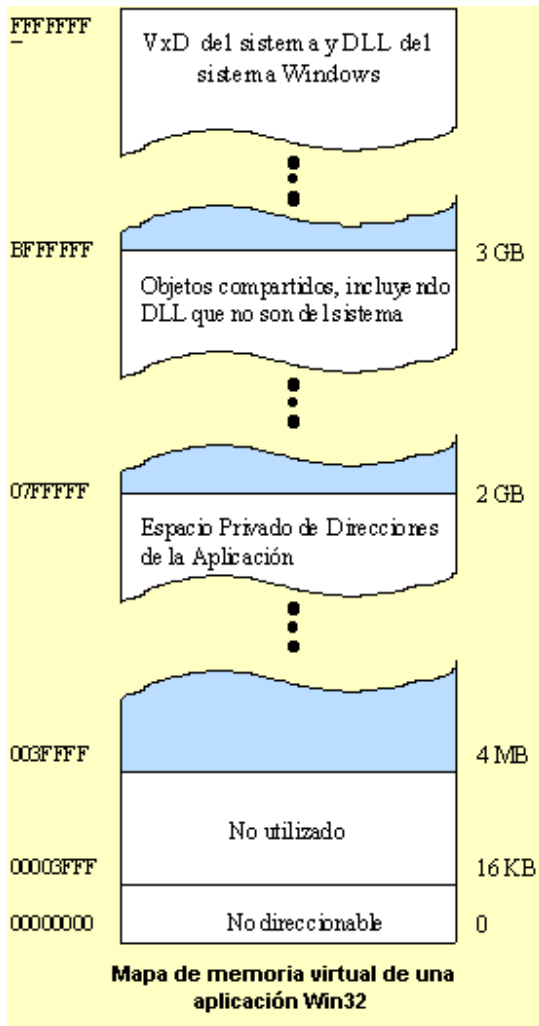
4 GB	Sistema Operativo	Núcleo, GDI Software del anillo 0	Sistema
3 GB	Usuario	DLL y otros Objetos compartidos	Ubicaciones de memoria compartidas
2 GB		Aplicaciones Windows de 32 bits y 16 bits	Privado de las aplicaciones
4 MB		Aplicaciones Windows de 16 bits	
1 MB	Aplicaciones MS-DOS y Windows de 16bits		
0			

ASIGNACIÓN DEL ESPACIO DE MEMORIA VIRTUAL

Las direcciones de memoria se asignan de la siguiente forma:

- **0-1 MB:** en una memoria virtual de MS-DOS, estas direcciones las utilizan las aplicaciones basadas en MS-DOS. Si no se trata de una memoria virtual de MS-DOS, las aplicaciones no utilizan las direcciones sino que éstas quedan disponibles para cualquier controlador de dispositivo de modo real que esté cargado.
- **1-4 MB:** normalmente no se utilizan. Windows NT se carga por encima de esta dirección y, para mantener la compatibilidad, Windows 95 no utiliza estos espacios de direcciones, como tampoco lo hacen las aplicaciones de 32 bits basadas en Windows. No obstante, estos espacios de direcciones están disponibles para que los utilicen las aplicaciones de 16 bits basadas en Windows.

- **4MB-2GB:** las utilizan las aplicaciones de 32 bits basadas en Windows (y algunas de 16 bits).
- **2-3 GB:** las utilizan las DLL y otros objetos compartidos. Por ejemplo, los cuadros de diálogo estándar que utilizan todas las aplicaciones se encuentran en las DLL de diálogos comunes, COMMDLG.DLL y COMDLG32.DLL. Estos archivos se cargan en este espacio de direcciones.
- **3-4 GB:** están reservadas para uso del sistema operativo (todos los componentes del anillo cero se asignan aquí). Por ejemplo, los controladores de vídeo y otros controladores virtuales se cargan en este espacio de direcciones.



Como se puede ver en la Figura, Windows 95 permite que una aplicación Win32 consuma un espacio virtual de direcciones enorme -y hay bastantes características nuevas disponibles para los programas Win32 para asegurar el consumo de todo ese espacio, incluyendo verdadera memoria compartida y diversas capacidades nuevas de asignación de memoria dinámica. El SO base asigna todas las regiones de memoria virtual privada de las aplicaciones Win32 dentro de los 2 GB inferiores del espacio virtual de direcciones. Todos los objetos de memoria compartidos - por ejemplo, las regiones de memoria compartida creadas por la aplicación - residen dentro de la región que va de los 2 GB a los 3 GB.

Se puede ver que una aplicación Win32 tiene un verdadero espacio de direcciones de 4 GB. Las llamadas a las DLL del sistema son llamadas rectas sin transición de anillo y sin [cambio de contexto](#). La ventaja de esta mejora está en la velocidad -no existe sobrecarga más allá de la sobrecarga de la propia llamada de la función. La desventaja es que una aplicación puede obtener un puntero del espacio de direcciones del sistema y empezar a acceder - posiblemente sin ningún efecto.

Los servicios de gestión de memoria del SO base deben admitir la creación de muchos nuevos tipos diferentes de objetos de memoria dentro del espacio virtual de direcciones de la aplicación.

Muchas funciones de gestión de memoria virtual requieren que el sistema haga una copia de seguridad de la memoria virtual mediante la asignación de memoria física en algún momento (aunque existen funciones que simplemente reservan regiones del espacio virtual de direcciones que nunca se utilizarán). Sin embargo, la asignación de memoria física real (es decir, la asignación de RAM) puede que no ocurra inmediatamente, ya que no hay necesidad de hacer una copia de seguridad de la memoria virtual hasta que la aplicación alcance la página de memoria. Pero el sistema tiene que seguir unos pasos para asegurarse de que hay espacio disponible en el [archivo de intercambio](#).

<http://intercambio.dpress.com>

Gestión de memoria para aplicaciones de Windows 95.

Montón se refiere a la región de memoria utilizada para satisfacer las peticiones de asignación de memoria por parte de la aplicación. Montón local se refiere al espacio de direcciones de la aplicación. Montón global que pertenece al sistema.

Windows 95 ofrece funciones que proporcionan soporte a montones privados, ya que una aplicación puede reservar una parte de la memoria de su propio espacio de direcciones. Una aplicación puede crear y utilizar tantos montones privados como desee y puede hacer que el sistema satisfaga las llamadas de asignación de memoria subsiguientes para un montón privado concreto. Una aplicación podría utilizar las funciones del montón local para crear zonas de memoria diferentes, conteniendo cada una estructuras de datos del mismo tipo y tamaño.

- Windows 95 proporciona funciones que permiten a una aplicación reservar una región específica de su propio espacio de direcciones virtuales que, una vez reservada, no se utilizará para satisfacer otras peticiones de asignación de memoria dinámica. En una aplicación [multihebra](#), el puntero de 32 bit a esta región reservada es una forma sencilla de ofrecer a cada [hebra](#) acceso a la misma memoria.

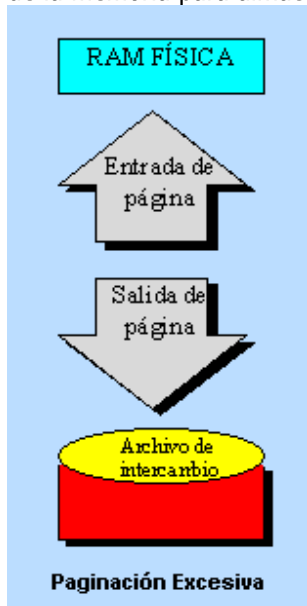
- Los archivos mapeados en memoria permiten que aplicaciones diferentes compartan datos. Una aplicación puede abrir un archivo identificado y hacer corresponder una región del archivo en su espacio virtual de direcciones. Los datos del archivo son después directamente direccionables por medio de una dirección de memoria de 32 bits única. Otras aplicaciones pueden abrir el mismo archivo, hacerlo corresponder con una sección de su espacio privado de direcciones y referenciar los mismos datos con un único puntero.

Paginación excesiva en Windows 95. (Ver [Control del Rendimiento del Sistema](#))

En determinadas situaciones, el sistema dedica más tiempo a intercambiar datos entre la RAM y el archivo de intercambio que a procesar datos, en la Figura se ilustra el intercambio de páginas.

Cuando una dirección virtual señala una página que no se encuentra en la memoria física (es decir, la página a la que se hace referencia está almacenada en el disco duro), el procesador genera un [fallo de página](#). Dicho fallo

indica al administrador de memoria virtual que cargue en la memoria la página que se encuentra en el disco. Si no hay suficiente memoria disponible para cargar la página, el administrador de memoria virtual debe sacar una página de la memoria para almacenarla en el disco.



El sistema está diseñado para permitir una cierta capacidad de paginación. Sin embargo, si se pagina frecuentemente, puede llegar un momento en que el sistema pagine tanto que apenas quede tiempo para hacer trabajo útil. La paginación excesiva (o la pérdida de control del disco) degrada el rendimiento global del sistema y puede ocasionar una avería prematura del disco duro. Entre los síntomas de una paginación excesiva se incluyen:

- **Utilización muy elevada de la CPU.**
- **Respuesta del sistema extremadamente lenta.**
- **Alta actividad del disco duro (casi constante).**

Por lo general, la solución a la paginación excesiva consiste en agregar más RAM al equipo o en trabajar con menos aplicaciones simultáneamente.

Archivos proyectados en memoria en Windows 95

En Windows la nueva característica de la gestión de memoria es el soporte para operaciones de memoria compartida a través de los archivos proyectados en memoria. Este es el modo recomendado de asignación y de utilización de regiones de memoria compartida. Las aplicaciones utilizarán esta característica para permitir el acceso a grandes estructuras de datos residentes en memoria. Para acceder a un archivo proyectado en memoria, una aplicación debe obtener un manejador a un objeto de proyección de archivo mediante la utilización de la función `CreateFileMapping()` de la API.

Gestión de la memoria del sistema .

La asignación de bloques de memoria en tiempo de ejecución, utilizan una referencia a un bloque para manipularlo, y en último caso la devolución del bloque al sistema para su reutilización, es la forma en la cual los programadores de Windows han tratado siempre con los requisitos de memoria dinámica.

Windows 95 no es diferente, lo que ha cambiado es la forma en la que el sistema capta las peticiones de memoria dinámica de la aplicación. La API de Windows sólo manipula el espacio virtual de direcciones de la aplicación. Esto significa que la solicitud de un bloque de memoria por parte de una aplicación hace que se adapte el mapa de direcciones virtuales de la aplicación, pero no hace absolutamente con la memoria física del sistema. Recuérdese que el 386 trata con la memoria física por medio de páginas de 4K. El mapa del espacio virtual de direcciones de cada aplicación de Windows refleja este formato de página. Por ejemplo, si una aplicación solicita 100K de memoria, su espacio virtual de direcciones tendrá 25 páginas más de memoria. El sistema también adaptará los datos a sus propias estructuras de control para reflejar el nuevo mapa de memoria de la aplicación.

En el momento de la asignación, Windows no hará nada con la memoria física del sistema. Sólo en el momento en que la aplicación comienza a utilizar la memoria es cuando el administrador de la memoria del sistema subyacente entra en juego y asigna páginas de la memoria física para hacerlas corresponder con las referencias de memoria virtual hechas por la aplicación. Si la aplicación asigna, pero no referencia una región de su espacio virtual de memoria, puede que el sistema nunca asigne memoria física para hacerla corresponder con la memoria virtual. La capacidad del 386 de permitir que las páginas de memoria física se utilicen en diferentes momentos dentro de distintos espacios virtuales de direcciones es la base de las capacidades de memoria virtual del sistema operativo.

Dentro del sistema se encuentran varias primitivas de gestión de memoria a disposición de los [controladores de dispositivos](#) y otros componentes del sistema que unas veces tratan con la memoria virtual y otras fuerzan al sistema a disponer páginas reales de memoria física. Pero esas primitivas son específicas del sistema operativo base. Ni las aplicaciones ni el subsistema Windows conocen o se preocupan de la memoria física. Las aplicaciones fuerzan al sistema a que asigne memoria física tan sólo mediante el uso real de la memoria: concretamente, mediante lecturas y escrituras en las posiciones de una página. La separación de la gestión de memoria de Windows entre los niveles virtual y físico en un espacio clave del sistema. Las aplicaciones y los subsistemas de Windows tratan con API definidas y espacios virtuales de direcciones. El sistema base lo hace con la memoria física, al igual que con los espacios virtuales de direcciones. Aunque la memoria física es transparente para una aplicación, la conducta de éste puede afectar el rendimiento del sistema en forma radical.