**Tema 4: Memoria Virtual**

**Traducción de direcciones**

Sistemas multiusuario o multiprogramado con varios programas ejecutándose concurrentemente:

* Tamaño memoria necesario >> memoria principal.
* Sólo una pequeña porción de la memoria se está utilizando activamente en un instante determinado.

Los programas siempre tienen las mismas direcciones lógicas:

* Reubicación.
* Traducción de direcciones.

Hasta hace “pocos” años, tamaño de un programa > memoria física:

* Overlays.
* Memoria virtual.

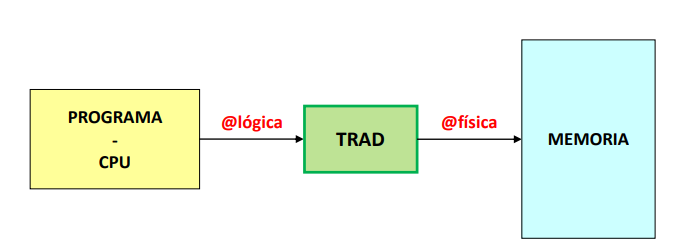
Idea básica:

Diferenciar **espacio lógico** (dirección generada por el procesador) de **espacio físico** (dirección con la que se accede a memoria).

En general son dos espacios diferentes => por lo tanto, necesitamos un mecanismo de traducción de direcciones (que traduzca las memorias lógicas generadas por el programa a su memoria física, ubicada en la memoria “real”).

Esquemas básicos de traducción:

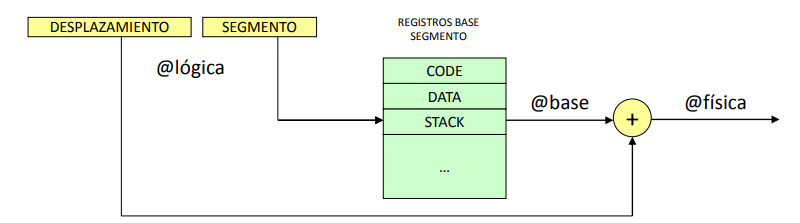
* Segmentación.
* Paginación.



Segmentación:

El programa se descompone en segmentos: código, datos, pila … Cada segmento se identifica por su dirección inicial y tamaño. Los segmentos se almacenan de forma contigua en memoria y de forma disjunta entre segmentos.

El mecanismo de traducción es bastante simple:



Un cambio de contexto (usuario o programa) implica cambiar el contenido de los registros.

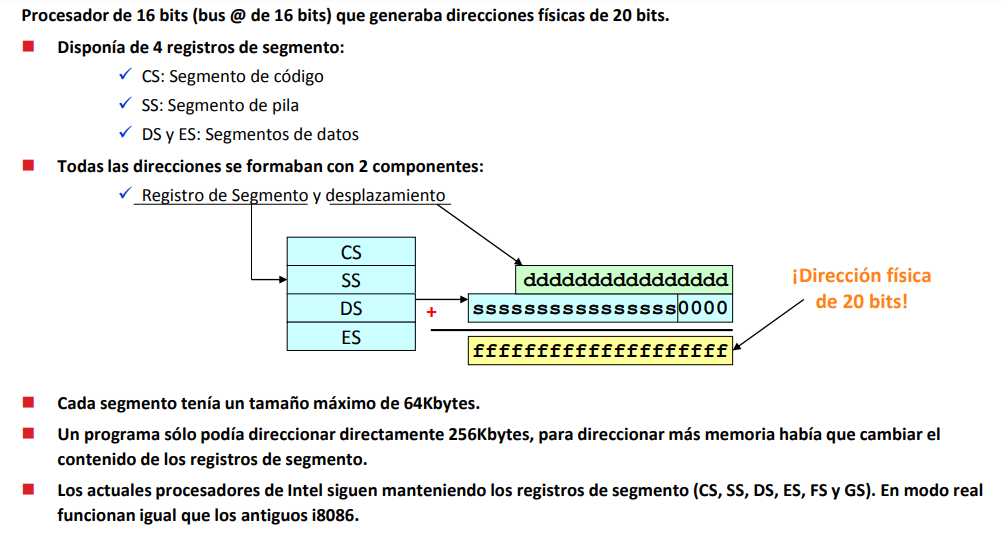
Acceso lento: acceso al banco de registros de segmentos y suma.

Reubicación muy simple.

Fragmentación de la memoria.

Permite protección de los segmentos.

Ejemplo i8086/88:



Paginación:

El espacio lógico se divide en bloques de tamaño fijo => PÁGINAS.

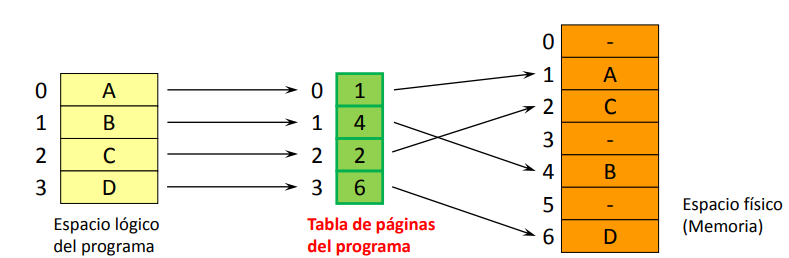
Los sistemas actuales tienen páginas con tamaño entre 4 y 16 KB.

El espacio físico (MP) se divide en marcos de tamaño una página (frames, tramas).

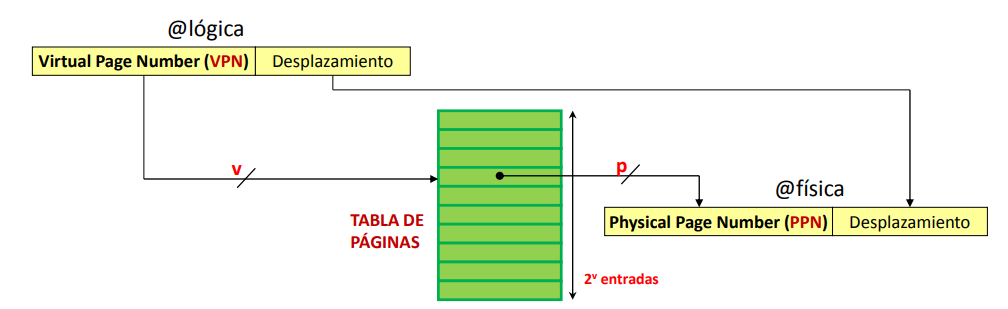
Una página puede colocarse en CUALQUIER marco de página de MP (correspondencia completamente asociativa).

Las páginas se copian desde disco a MP cuando son referenciadas.

Hace falta una estructura de datos para saber qué hay en cada marco de página => TABLA DE PÁGINAS.



Implementación hardware de la paginación:



Cálculo rápido de la dirección (no hay operaciones aritméticas).

Fragmentación (ficheros pequeños ocupan 1 página completa).

Reubicación muy simple.

Permite protección de páginas.

Páginas físicas y virtuales tienen el mismo tamaño.

VPN y PPN pueden tener longitud diferente.

En la mayoría de sistemas se cumple: 2v > 2p.

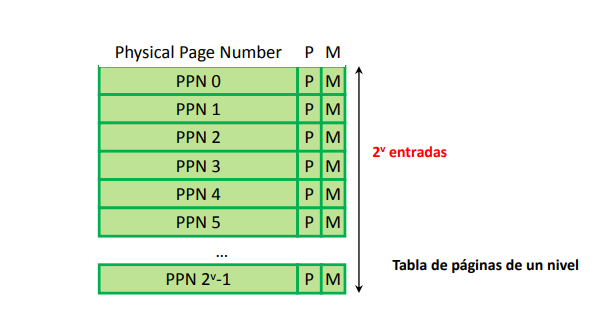
**Tabla de páginas y TLB**

Implementación de la tabla de páginas:

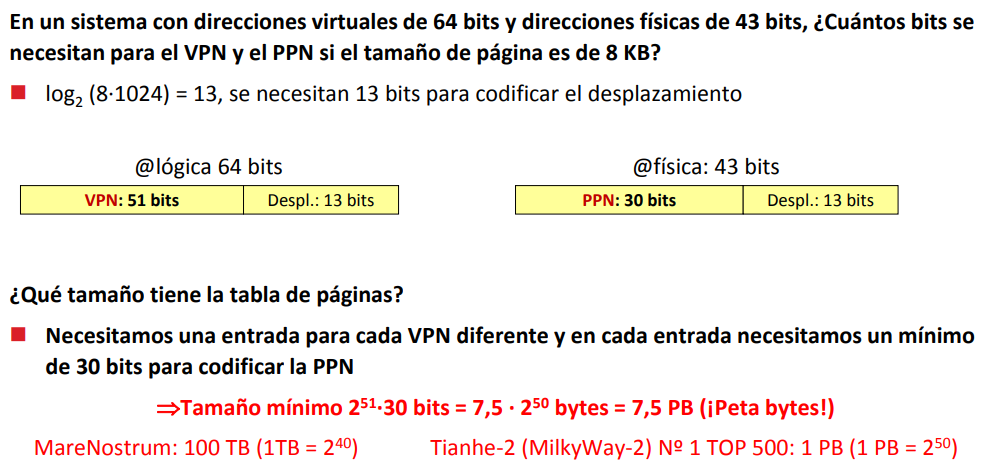
Cada proceso tiene sus propias @ lógicas y físicas y su propia tabla de páginas.

P: bit de presencia (indica si la página está almacenada en MP).

M: bit de modificación (indica si la página ha sido modificada en MP).



Ejemplo práctico de paginación:



Implementación de la tabla de páginas:

En un procesador actual, la TP sería mucho más grande que la memoria principal.

Solución: tablas de paginación de múltiples niveles:

* Sólo una parte de la tabla de páginas está en MP.
* Se requieren varios accesos a la tabla de páginas para conocer la @ física de la página.

En este curso, utilizaremos como modelo una tabla de páginas de un solo nivel almacenada siempre en MP.

La table de páginas es accedida en cada referencia a memoria.

Si la tabla de páginas es de un nivel y se almacena en memoria principal:

* 1 acceso a MP necesita 1 acceso a la tabla de páginas y 1 acceso al dato (MUY LENTO).

Solución: tener una memoria cache “especial” para la tabla de páginas:

* Translation Lookaside Buffer (TLB) – Buffer de traducción anticipada.

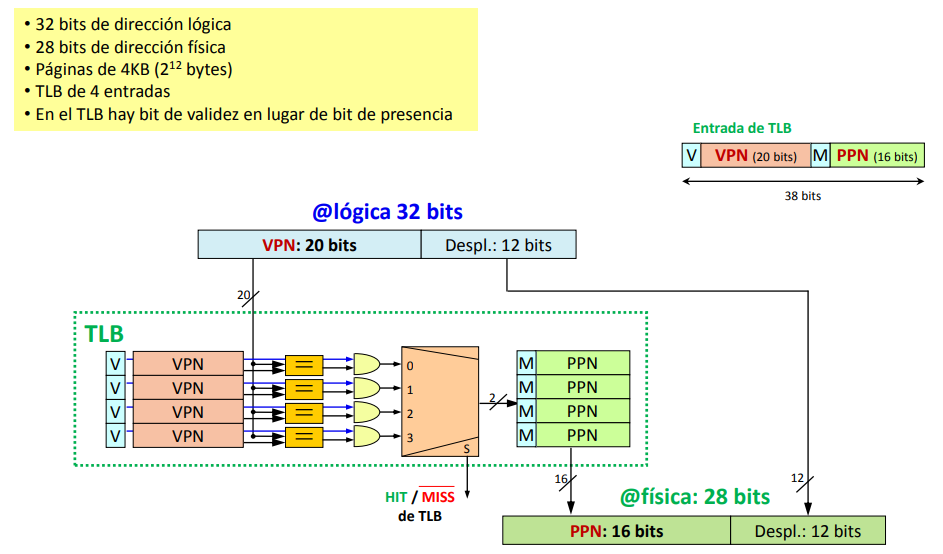
En el TLB no hay datos (o programas), sólo información para acelerar la traducción de direcciones.

Translation Lookaside Buffer (TLB):

* Sirve para acelerar el proceso de traducción de direcciones.
* Tiene una estructura similar a la tabla de páginas.
* Contiene más entradas de página que las páginas que caben en la cache L1 (contiene traducciones de datos residentes en L2 y MP).

Características principales:

* Integrado en el mismo chip que en el procesador.
* Pocas entradas (64-128) (1 entrada por página).
* Completamente asociativo.
* Tasa de fallos muy baja.
* Muy rápido (debido a que tiene pocas entradas de pocos bits).
* Algoritmo de reemplazo (LRU, PesudoLRU, FIFO, Random, …).



Protección (paginación):

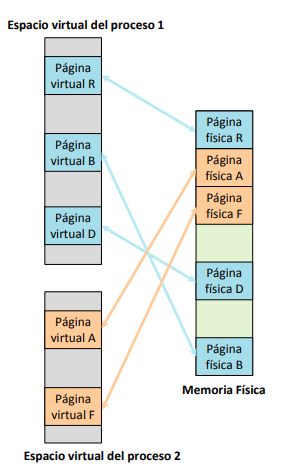
Cada proceso tiene su propia tabla de páginas.

Ventajas:

* Los procesos comparten espacio físico de direcciones, pero tiene espacios virtuales distintos.
* El sistema de traducción de direcciones asegura que las páginas virtuales de cada proceso se mapean en páginas físicas distintas (en MO y en disco).
* Si dos procesos quieren compartir sus datos, algunos SO permiten realizar una petición especifica para que algunas de sus direcciones virtuales se asignen a las mismas direcciones físicas.

Inconvenientes:

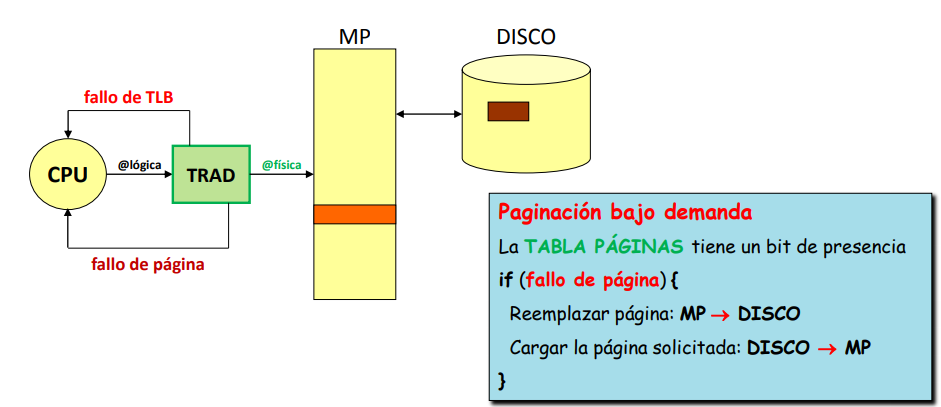
* El mapeo de direcciones virtuales a físicas es parte del estado del proceso.
* Cuando el SO realiza un cambio de contexto, hay que invalidar el TLB.
* Cuando se comienza a ejecutar un proceso hay muchos fallos de TLB.
* Para solventar este problema, algunos sistemas actuales incorporan un ID de proceso en el TLB (coexisten entradas de procesos distintos).



**Memoria Virtual**

La memoria virtual permite:

* Ejecutar un programa con espacio lógico > espacio físico.
* Ejecutar un programa parcialmente cargado en memoria.
* Proteger el espacio de direcciones de los programas de ser accedido por otros programas.



¿Quién gestiona la memoria virtual?

* El sistema operativo (software). La memoria cache es hardware.

¿Cuándo se trae una página de disco a MP?

* Bajo demanda en caso de fallo (hay otros modelos).

¿Dónde se ubica una página en MP?

* En cualquier marco, política totalmente asociativa.

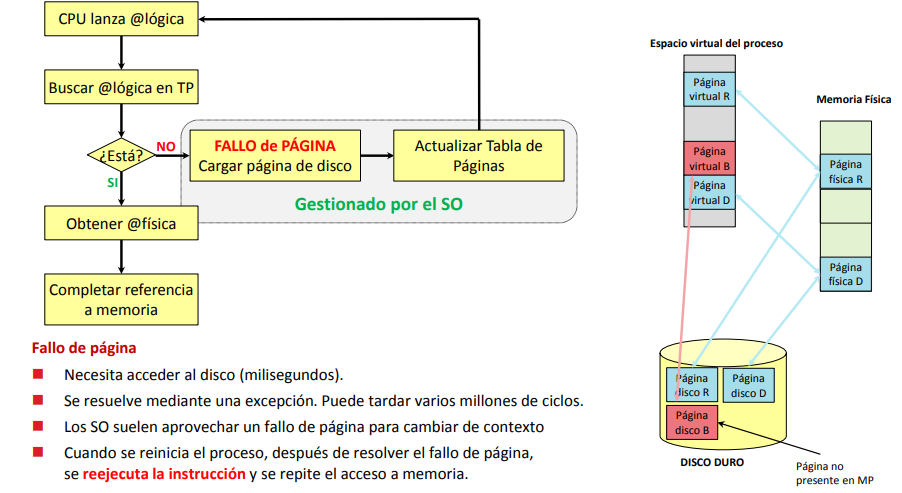
¿Qué página de la MP se substituye en caso de fallo?

* Algoritmos de reemplazo muy sofisticados. La tasa de fallos es muy importante. Un fallo puede costar millones de ciclos porque hay que acceder a disco. La decisión es software y hay mucho tiempo para tomarla.
* Las páginas modificadas hay que escribirlas en disco.
* Tasa de fallos: 0.00001% - 0.001%.

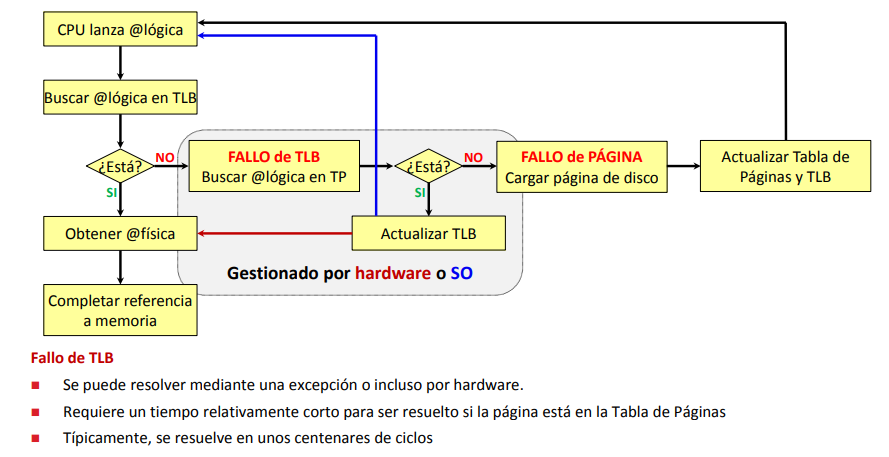
¿Qué se hace con las escrituras?

* COPY BACK + WRITE ALLOCATE.

Paginación bajo demanda (sin TLB):



Paginación bajo demanda (con TLB):



**Juntando memoria virtual y memoria cache**

La traducción de direcciones y la memoria cache son conceptos ortogonales:

* La memoria cache permite acelerar los accesos a memoria.
* La traducción de direcciones permite soportar memoria virtual. (El TLB es solo un mecanismo de aceleración del proceso de traducción).

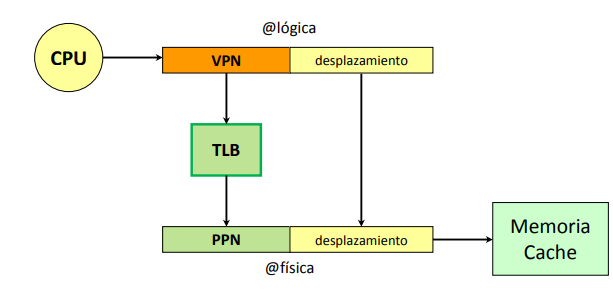
Un sistema puede tener solo memoria cache, solo traducción de direcciones, ambos mecanismos o ninguno de ellos.

Los actuales procesadores de propósito general cuentan con una jerarquía de uno o más niveles de cache y mecanismos de traducción de direcciones con el correspondiente TLB.

En este último caso, ¿cuándo se efectúa la traducción de direcciones lógicas a físicas, antes o después de acceder a la memoria cache? 3 posibilidades:

* Traducción antes de acceder a memoria cache.
* Traducción después de acceder a memoria cache.
* Traducción y acceso a memoria cache simultáneos.

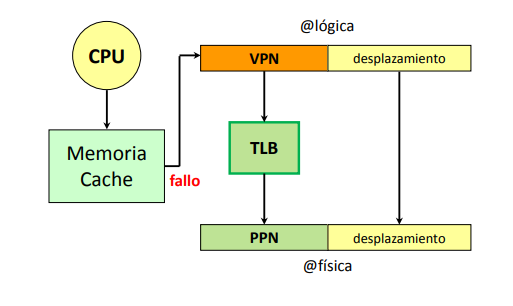
Traducción antes de acceder a memoria cache:



Memoria cache de direcciones físicas.

Lento: un acceso a memoria necesita un acceso TLB + acceso MC.

Traducción después de acceder a memoria cache:

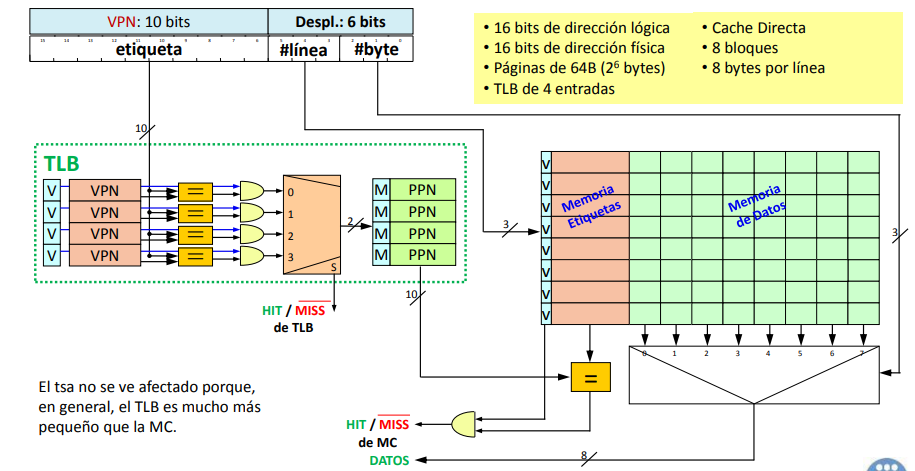


Memoria cache de direcciones lógicas.

Se realiza traducción solo en caso de fallo en MC.

Aumenta el coste de un fallo de MC.

Traducción en TLB y acceso a memoria cache simultáneos:

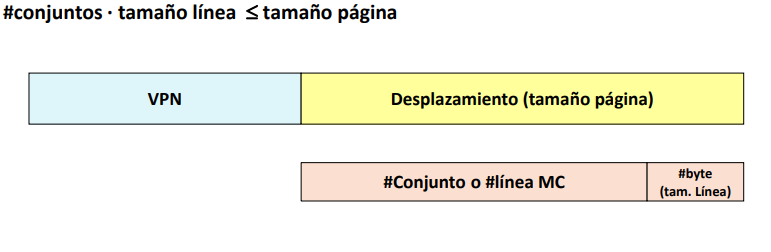


Se busca en la MC con la parte de la dirección que corresponde al desplazamiento (línea y byte de la línea).

La memoria de etiquetas contiene etiquetas físicas.

Se traduce únicamente la página lógica que corresponde a la etiqueta y se comprueba si la línea de la MC es el bloque buscado.

Restringe el tamaño de la memoria cache:



#conjuntos \* tamaño línea = vía