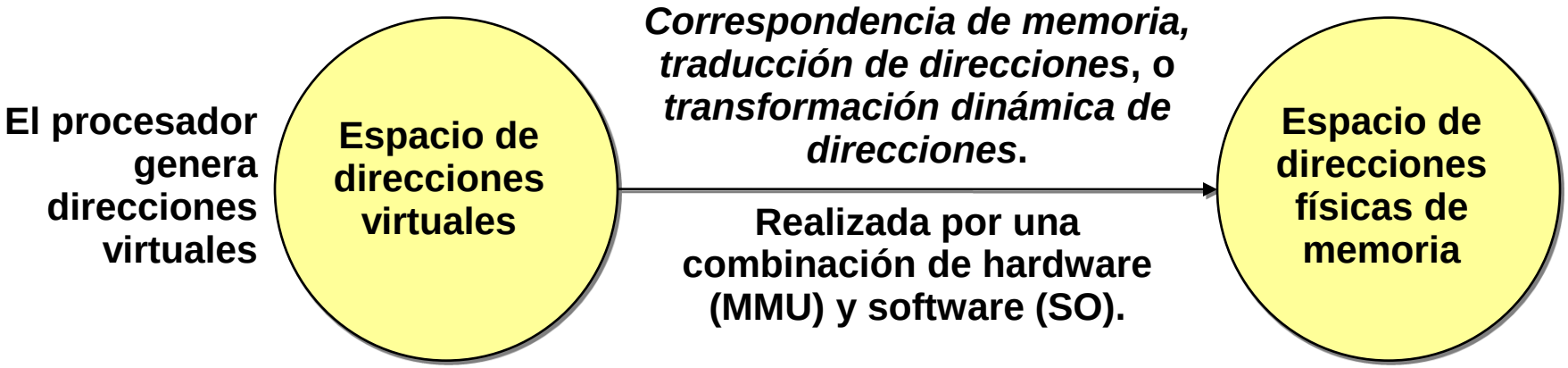
Concepto de memoria virtual

- **Problema: necesidad de programas mayores que la memoria física.**
- **1ª solución: OVERLAYS *(solapamientos, superposiciones)***
  - El programador divide los programas en partes que caben en la memoria, y son cargadas y descargadas durante la ejecución (sigue usándose en sistemas empotrados, sin memoria virtual).

  ⇒Gran esfuerzo de programación.
- **2ª solución: MEMORIA VIRTUAL**
  - Permite el acceso a un espacio de memoria mayor que el real.



Concepto de memoria virtual

- La memoria virtual corresponde a dos niveles de la jerarquía de memoria: MP (DRAM) y discos magnéticos.
  - Recordemos que la caché involucra los niveles de caché (SRAM) y MP (DRAM).
- Diferencias entre caché y memoria virtual como partes de la jerarquía de memoria:

Rangos típicos de parámetros de la jerarquía de memoria para caché

Tamaño de bloque (línea)	4-128 bytes
Tiempo de acierto	1-4 ciclos de reloj (normalmente 1)
Penalización de fallo	8-32 ciclos de reloj
(tiempo de acceso)	(6-10 ciclos de reloj)
(tiempo de transferencia)	(2-22 ciclos de reloj)
Frecuencia de fallo	1 % - 20 %
Tamaño de cache	1 KB - 256 KB

Rangos típicos de parámetros de la jerarquía de memoria para memoria virtual

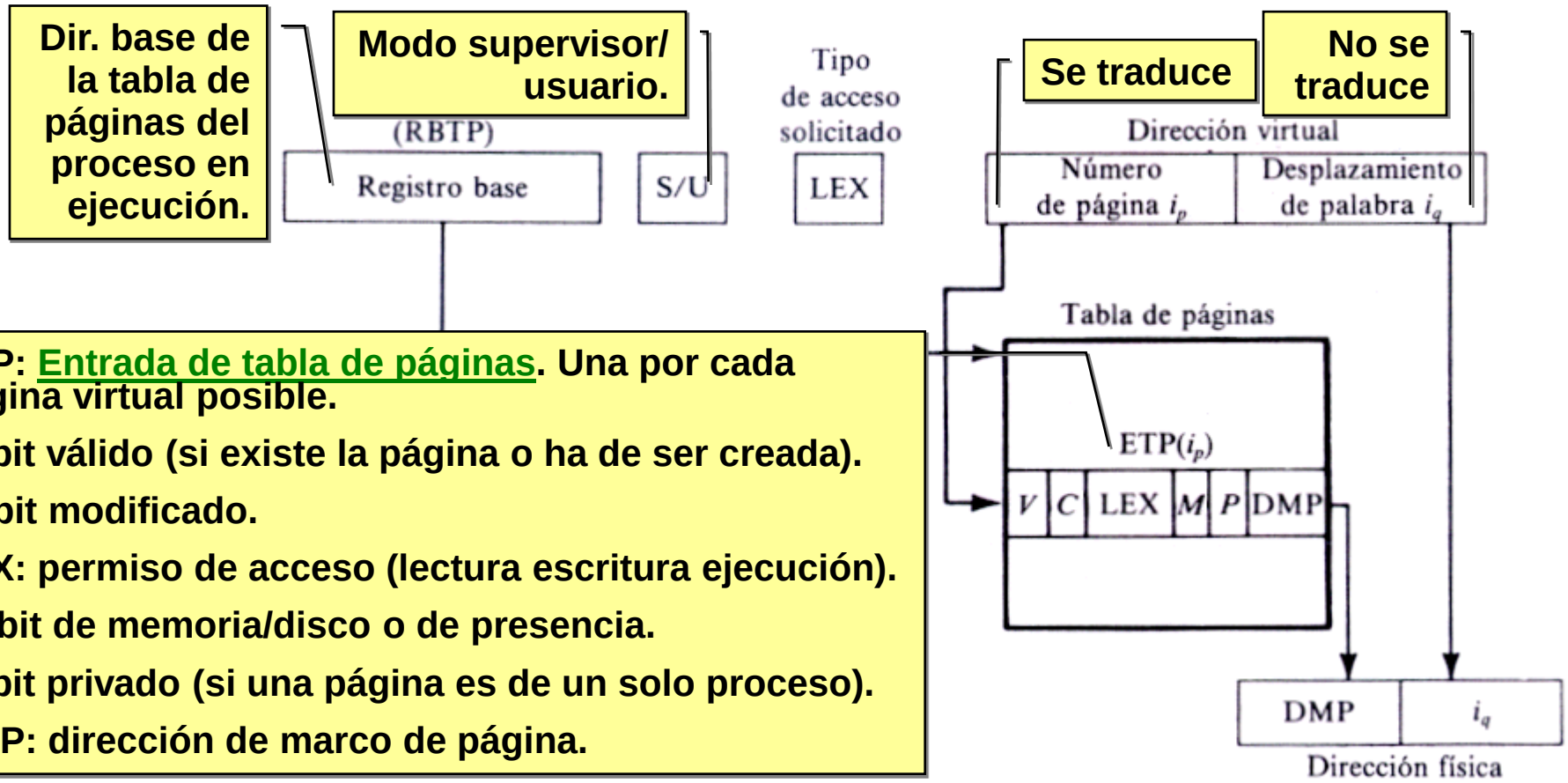
Tamaño bloque (página)	512-8192 bytes
Tiempo de acierto	1-10 ciclos de reloj
Penalización de fallos	100 000-600 000 ciclos de reloj
(tiempo de acceso)	(100 000-500 000 ciclos de reloj)
(tiempo de transferencia)	(10 000-100 000 ciclos de reloj)
Frecuencia de fallos	0,00001 %-0,001 %
Tamaño de memoria principal	4 MB-2048 MB

Concepto de memoria virtual

- **Esquemas de traducción de dirección virtual a física:**
  - Memoria paginada
    - El espacio virtual se divide en **páginas de tamaño fijo**, que pueden residir en bloques de igual tamaño (marcos de página) en la memoria física. (*Similar a caché*)
  - Memoria segmentada
    - Los programas se estructuran en **segmentos o módulos de tamaño variable**, cada uno con un espacio de direcciones propio y **cierta entidad lógica**.
  - Memoria con segmentos paginados
    - **Cada segmento se divide en páginas**. El nº de páginas por segmento puede variar, pero el nº de palabras por página permanece fijo.

Paginación

- **Memoria paginada**
  - Mecanismo de correspondencia entre direcciones virtuales y físicas: **tabla de páginas** (una por proceso).



ETP: Entrada de tabla de páginas. Una por cada página virtual posible.

***V*: bit válido (si existe la página o ha de ser creada).**

***C*: bit modificado.**

LEX: permiso de acceso (lectura escritura ejecución).

***M*: bit de memoria/disco o de presencia.**

***P*: bit privado (si una página es de un solo proceso).**

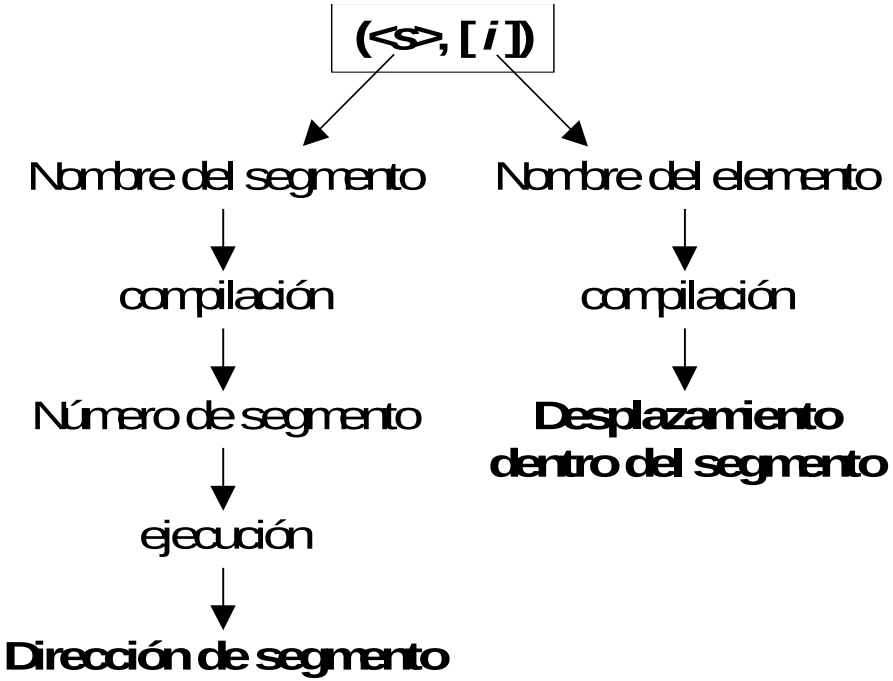
DMP: dirección de marco de página.

Paginación

- Ventajas e inconvenientes:
  - ✓ Reemplazo de bloque sencillo (todos los bloques tienen el mismo tamaño).
  - ✓ Tráfico de disco eficiente (se ajusta el tamaño de página para equilibrar tiempo de acceso y tiempo de transferencia).
  - ✓ El programador no ha de ser consciente de que usa esta técnica.
  - ✗ Fragmentación interna (porciones inutilizadas de páginas).

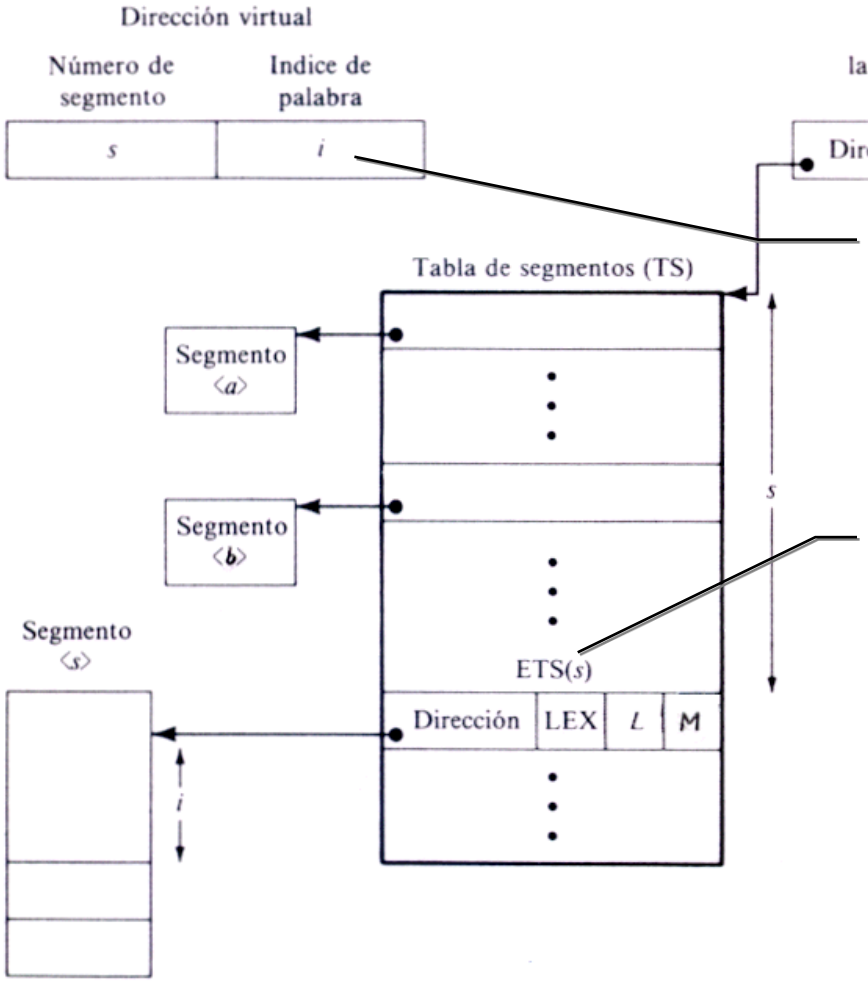
Segmentación

- **Memoria segmentada**
  - Segmento: conjunto de datos o instrucciones contiguos y relacionados lógicamente (pila, subrutina, matriz, datos, ...).
  - Un elemento de un segmento se referencia por:



Segmentación

- Mecanismo de correspondencia entre direcciones virtuales y físicas: **tabla de segmentos**.



Si $i \geq L \Rightarrow$ dirección fuera de rango $\Rightarrow$ violación de acceso.

ETS: Descriptor de segmento.

Dirección: dirección de segmento en memoria o disco

LEX: permiso de acceso (lectura escritura ejecución)

***L*: Longitud del segmento <s>**

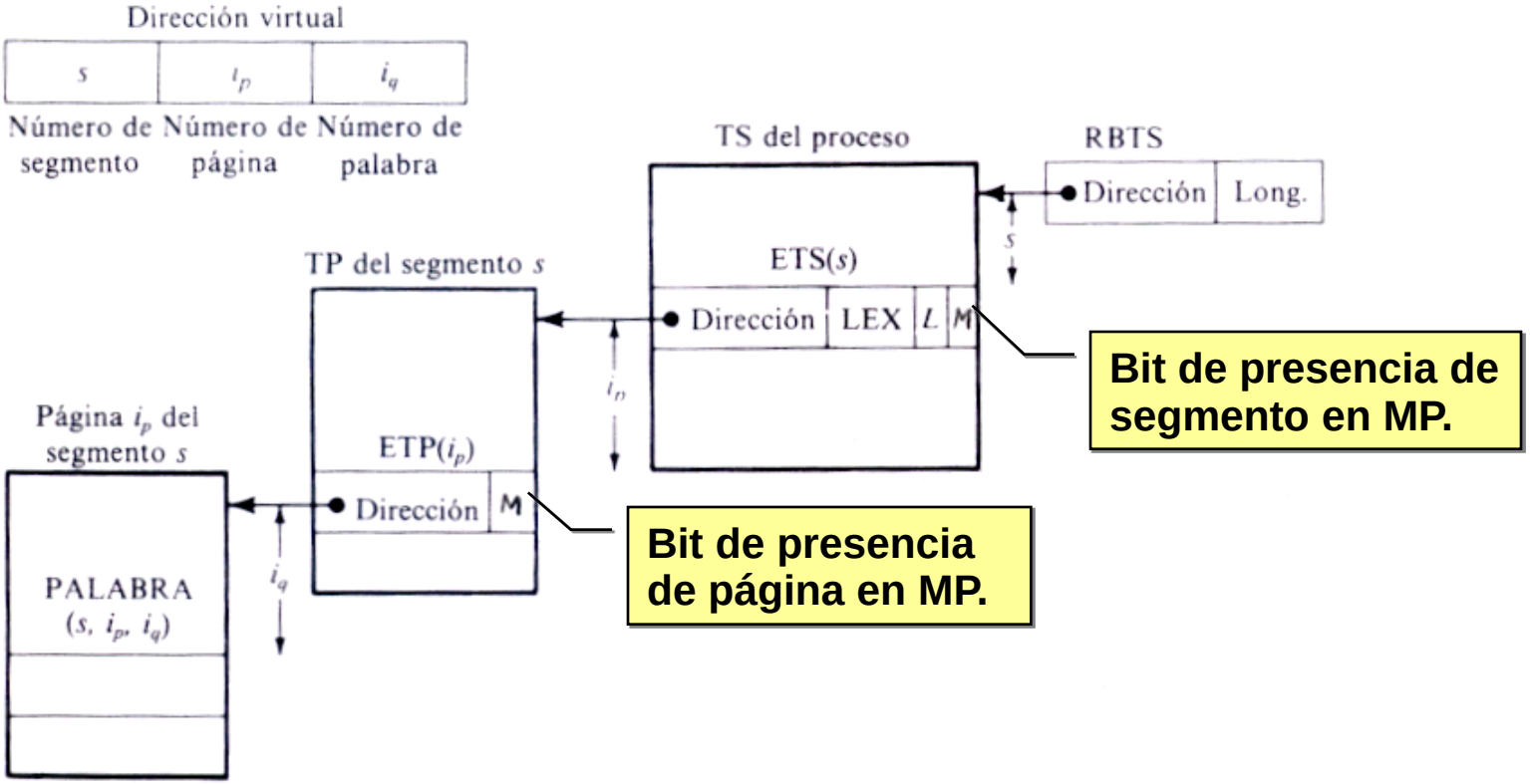
***M*: bit de memoria/disco o de presencia**

Segmentación

- Cuando una dirección virtual se traduce a física, puede generarse alguno de los siguientes *traps* (excepciones) ⇒ el control se transfiere al SO.
  - Falta de segmento.
  - Violación del espacio de direcciones.
  - Violación de la protección.
- Ventajas e inconvenientes:
  - ✓ Facilita la estructuración, la compartición y la protección.
  - × Reemplazo de bloque difícil (el SO ha de encontrar una parte no utilizada contigua de MP).
  - × Fragmentación externa (partes no usadas de MP) ⇒ necesidad de compactación.

Segmentación paginada

- **Memoria con segmentación paginada**
  - Mecanismo de correspondencia entre direcciones virtuales y físicas: **tabla de segmentos**; un segmento se divide en páginas y es accedido a través de una **tabla de páginas** (en el ejemplo de la figura **una para cada segmento**)

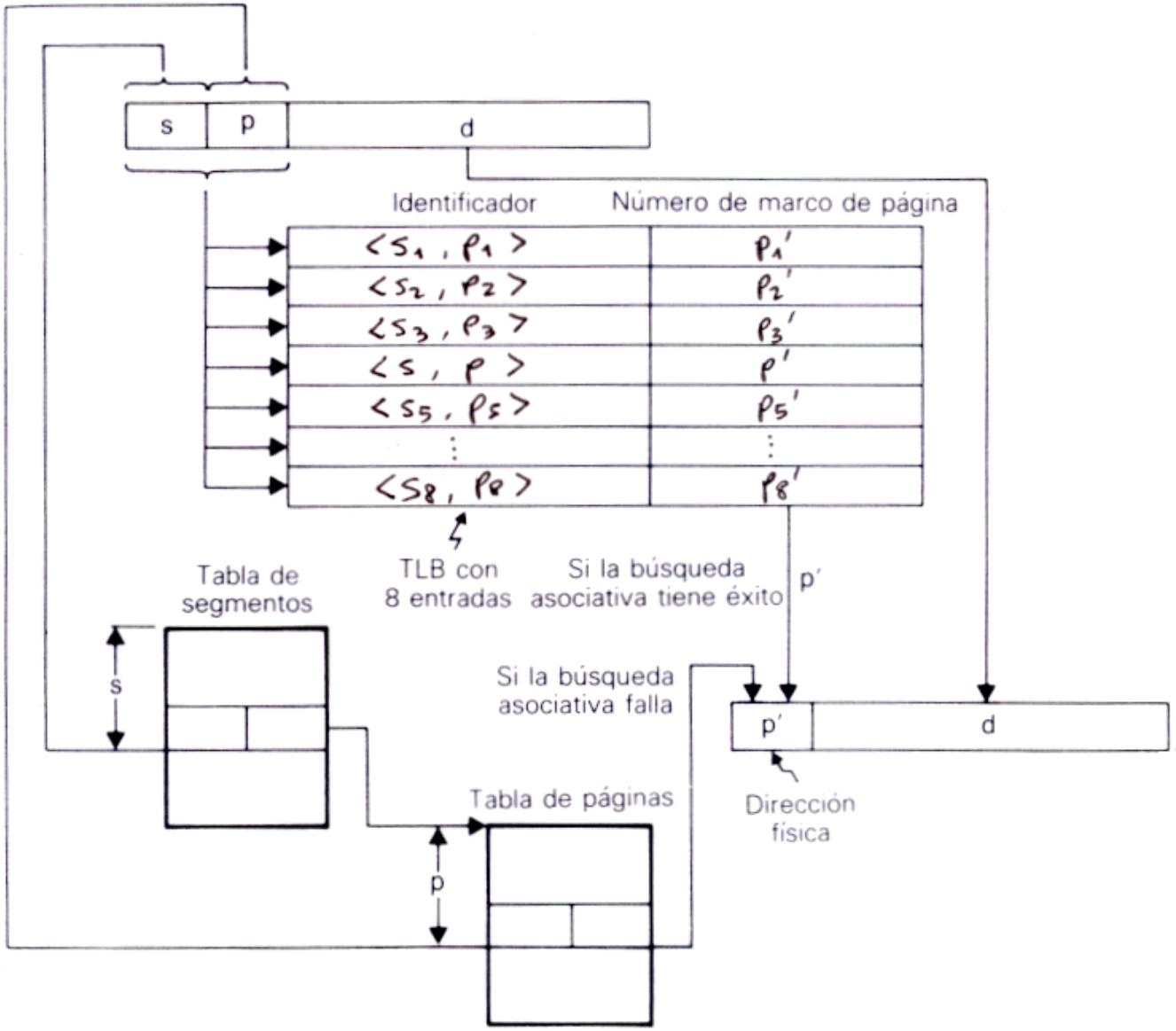


Segmentación paginada

- **Problema:**
  - Los tres métodos son ineficaces por requerir más de un acceso a memoria por dato accedido
- **Solución (basada en la localidad de las referencias):**
  - Búfer de traducción anticipada

    (TLB, *Translation Look-aside Buffer*)
    - Pequeña memoria caché que almacena la información relativa a las últimas direcciones de memoria accedidas
  - ¿Por qué hay mejora?
    - Porque el TLB usa una rápida memoria asociativa
    - Porque la tasa de aciertos es alta

Segmentación paginada



Administración de la memoria virtual

- **Estrategias de administración de memoria virtual**
  - **Política de colocación**
    - ¿Dónde puede ubicarse un bloque en MP?
      - Elección entre:
        - » Reducir la frecuencia de fallos
        - » Algoritmo de ubicación sencillo
      - Debido al enorme coste de un fallo, se elige reducir su frecuencia ⇒ los bloques se pueden colocar en cualquier posición de MP (correspondencia totalmente asociativa)

Administración de la memoria virtual

- Selección del **tamaño de página**
  - Páginas mayores:
    - Se ahorra memoria en la tabla de páginas.
    - Es más eficiente transferir páginas a o desde la memoria secundaria (se aprovecha más la localidad espacial).
  - Páginas menores:
    - Se desperdicia menos memoria debido a la fragmentación interna.
    - Se emplea menor tiempo en cada fallo de página.
- Estrategias para **posicionar nuevos segmentos** en los huecos libres de MP:
  - Primer ajuste: Fácil de implementar.
  - Mejor ajuste: Genera huecos muy pequeños.
  - Peor ajuste: Evita que se generen huecos pequeños.

Administración de la memoria virtual

- **Política de reemplazo**
  - ¿Qué página debería sustituirse en un fallo de página si la MP está llena?
    - Algoritmo óptimo de Belady o MIN: sustituir la página que no va a necesitarse en el más largo período de tiempo
      - » No es implementable
      - » Se utiliza en simulación para comparar con otros algoritmos y determinar la eficiencia de éstos
    - LRU: reemplazar la página menos recientemente usada
    - LFU: reemplazar la página menos frecuentemente usada
    - FIFO: reemplazar la página que lleva más tiempo en memoria
    - RAND: reemplazar una página escogida aleatoriamente

Administración de la memoria virtual

- **Políticas de precaptación de páginas** (*prefetching*)
  - Intentan cargar las páginas que se prevé que se van a utilizar en el futuro
  - Algoritmo de anticipación de bloque:
    - Cuando hay una falta de página, se precapta una página de más, adyacente a la que ha generado la falta. Será la primera en reemplazarse si no es accedida antes.

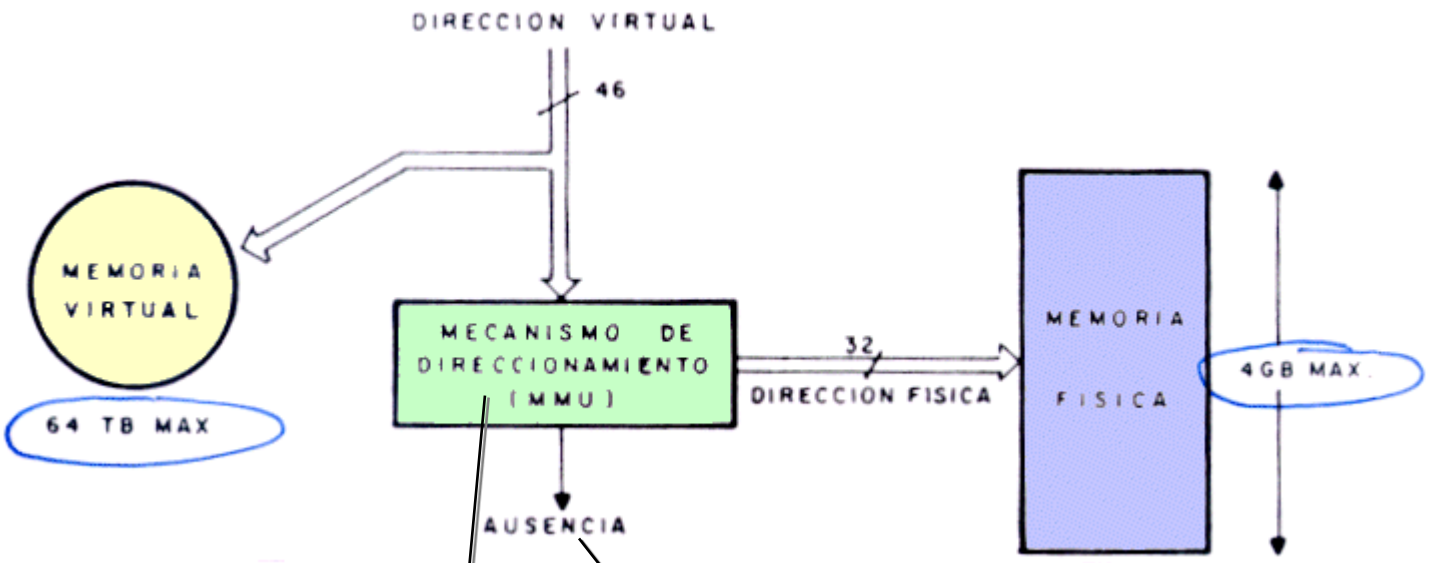
- **Actualización de la memoria virtual**
  - Discos muy lentos relativamente

  $\Rightarrow$ La estrategia de escritura siempre es post-escritura, incluyendo un bit de modificaciones (*dirty*), de manera que sólo los bloques alterados se escriban en disco.

Memoria virtual en x86

- **Modo protegido**
- **Se pueden distinguir tres espacios de direcciones:**
  - **Espacio virtual** o lógico
    - Abarca toda la memoria virtual y es el que maneja el programador de aplicaciones
  - **Espacio lineal**
    - Las direcciones virtuales hacen referencia a segmentos
    - Al situarse los segmentos sobre la memoria física, tienen dispuestas todas sus posiciones en un orden consecutivo o lineal
  - **Espacio físico**

Memoria virtual en x86

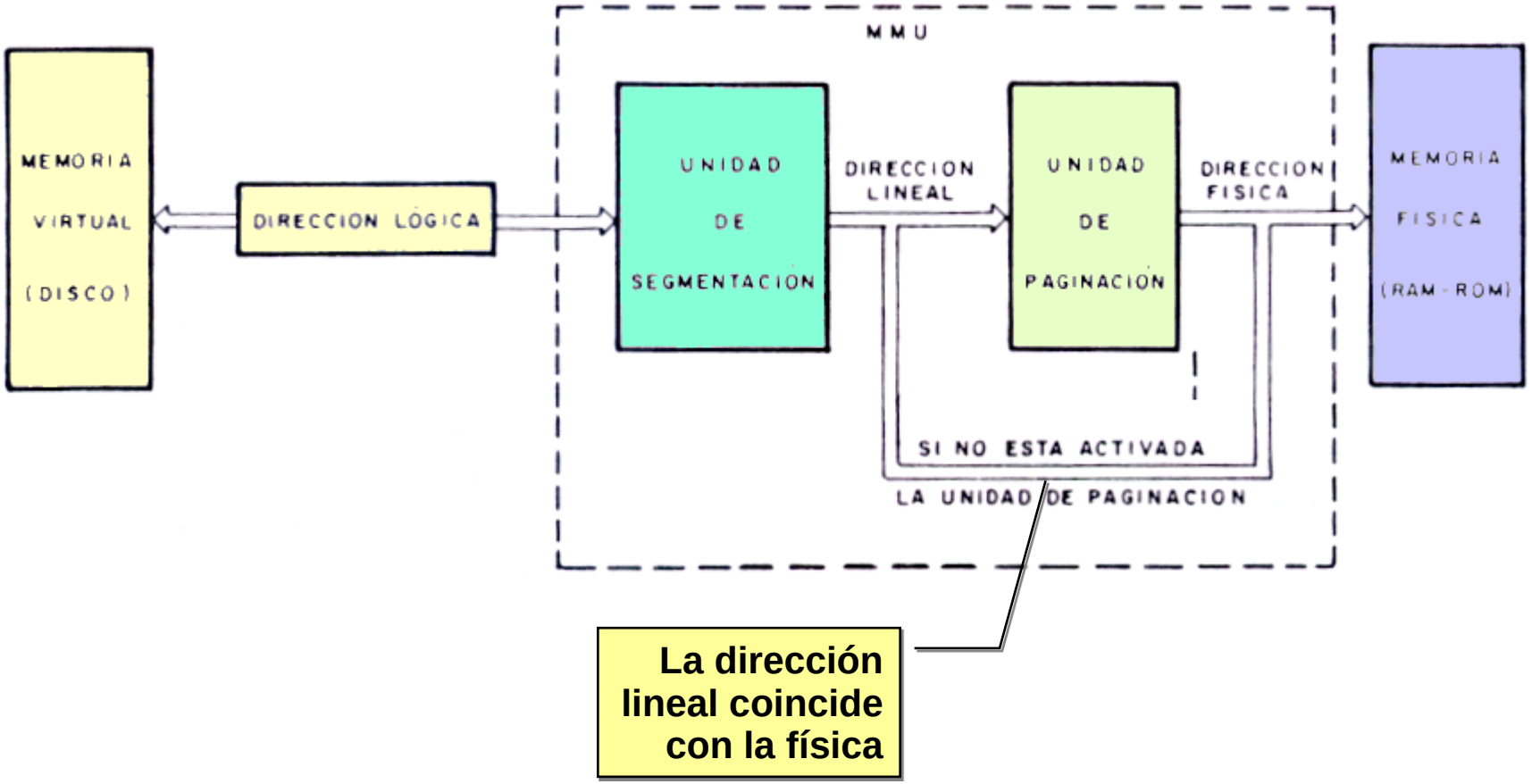


MMU: unidad de segmentación + unidad de paginación

Generación de una excepción del SO encargada de hacer la transferencia oportuna.

Memoria virtual en x86

- Traducción de la dirección lógica en direcciones lineal y física:



Memoria virtual en x86

- **Segmentación**
  - Espacio virtual o lógico
    - La dirección consta de dos partes:

SELECTOR (16 bits): Selecciona un segmento del espacio virtual

DESPLAZAMIENTO (32 bits): Posición dentro del segmento. Tamaño máximo del segmento = 2^{32} Bytes = 4GB.

ÍNDICE (13 bits): direcciona descriptores de segmento

TI (Indicador de tabla):
TI = 0 ⇒ Tabla global de descriptores
TI = 1 ⇒ Tabla local de descriptores

14 bits ⇒ máx. 16 K descriptores de segmento

RPL: nivel de privilegio del que pide el segmento (0=máx., 3=mín.).

14 + 32 = 46 bits ⇒ máx. espacio virtual = 2^{46} bytes = 64 TB.

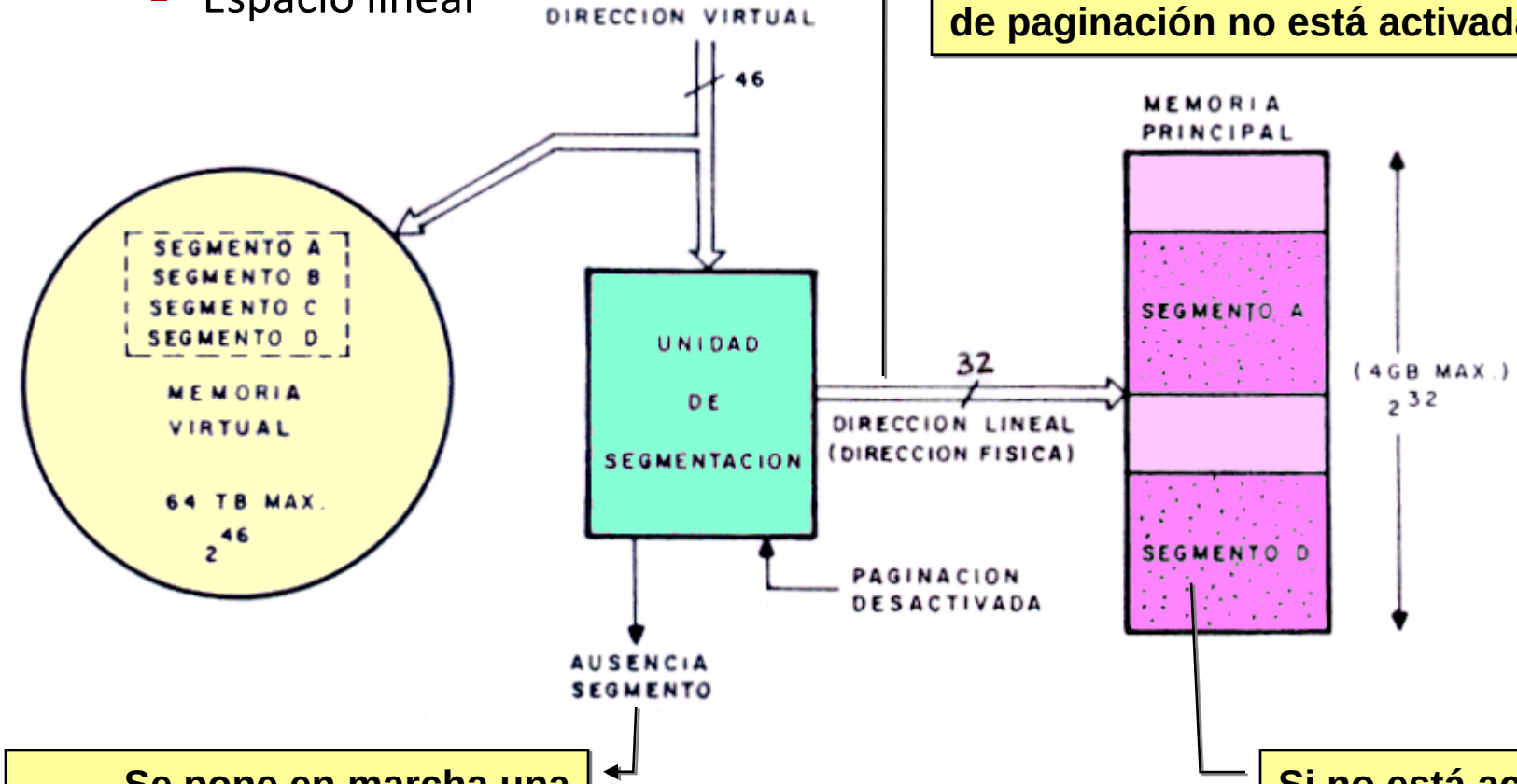
Memoria virtual en x86

- Espacios direccionados:

Selector	Desplazamiento	Tipo de segmento
CS	EIP	Código
SS	ESP	Pila
DS, ES, FS, o GS	Se calcula de acuerdo con el modo de direccionamiento de la instrucción.	Datos

Memoria virtual en x86

- Espacio lineal



Dirección lineal (32 bits). Coincide con la dirección física en MP si la unidad de paginación no está activada.

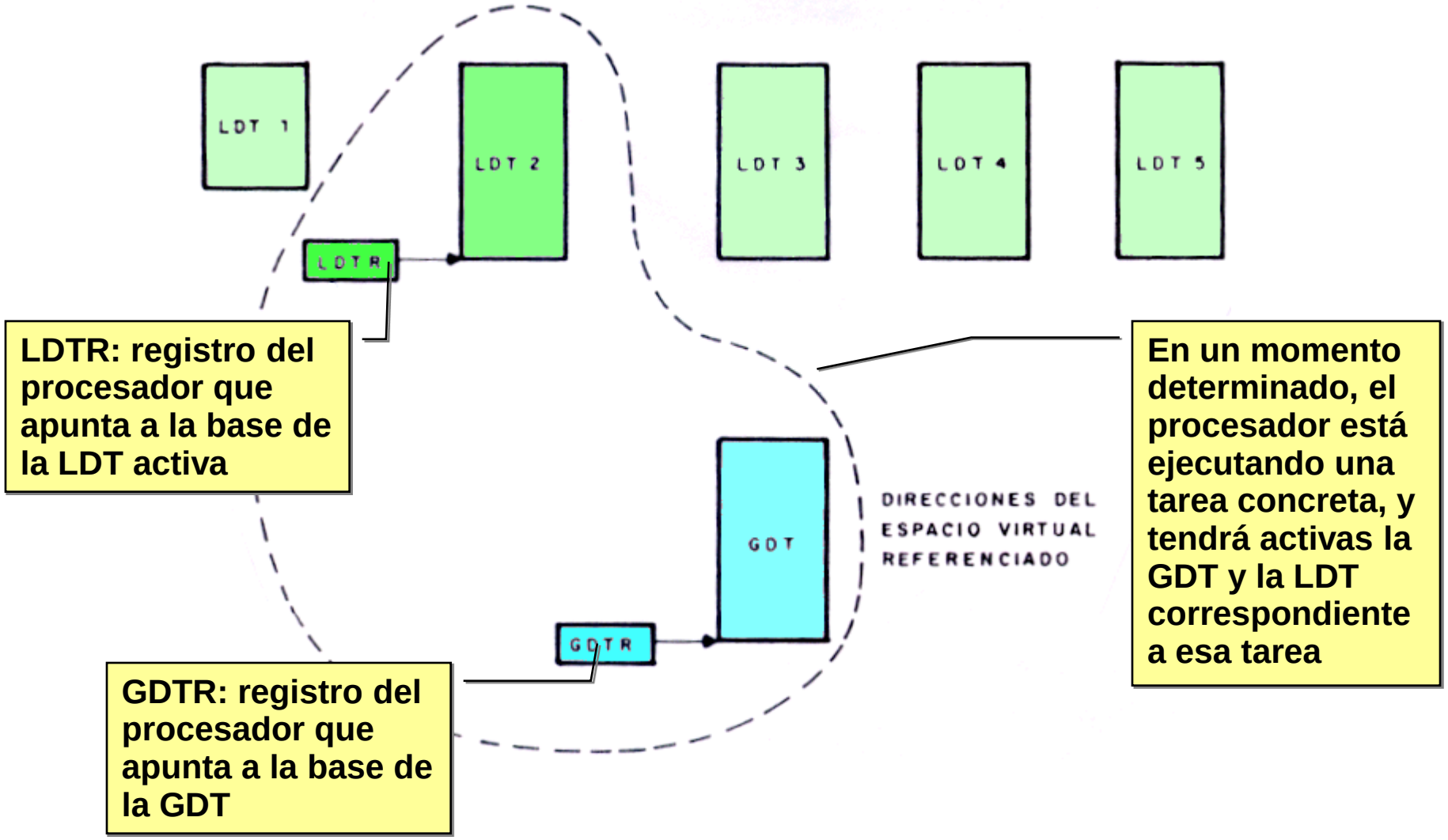
Se pone en marcha una rutina del SO que traslada el segmento de la memoria secundaria a la física

Si no está activa la paginación ⇒ la MP contiene segmentos completos

Memoria virtual en x86

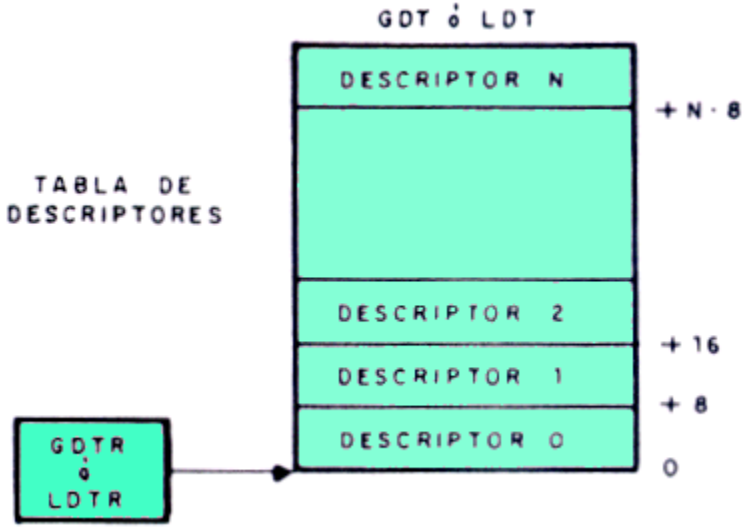
- Tablas de descriptores de segmento
  - Contienen los descriptores de todos los segmentos que usa el procesador.
  - Un **descriptor** es una estructura de datos (8 bytes) que especifica todos los parámetros que definen un segmento (base, límite y atributos).
  - Tenemos un sistema multitarea compuesto por:
    - Área global:
      - » Hay una **tabla global de descriptores (GDT)**.
      - » En ella residen los segmentos comunes a todas las tareas.
    - Áreas locales:
      - » Hay *n* **tablas locales de descriptores (LDT)**.
      - » Una para cada tarea.

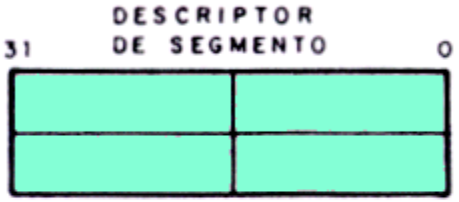
Memoria virtual en x86



Memoria virtual en x86

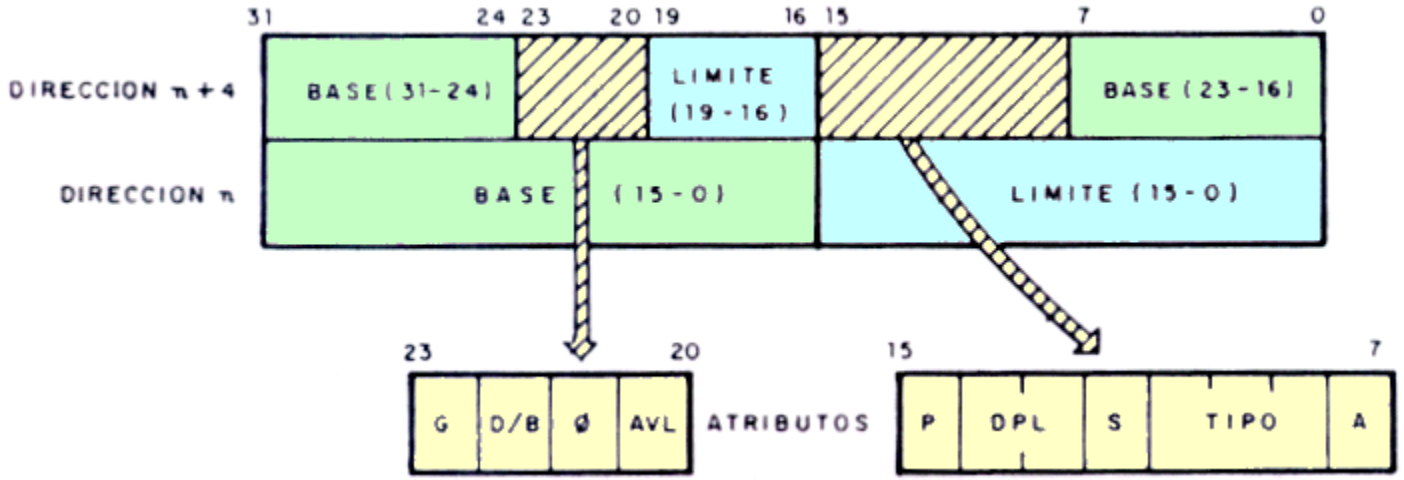
- Estructura de las tablas de los descriptores:





Memoria virtual en x86

- Descriptores de segmento:
  - Estructura de datos de 8 bytes que contiene los parámetros que definen el segmento referenciado (base, límite, y derechos de acceso o atributos):



  - **BASE** (32 bits): Dirección donde comienza el segmento.
  - **LÍMITE** (20 bits): Tamaño del segmento en bytes (si G = 0) o en páginas de 4 KB (si G = 1).
  - **ATRIBUTOS** o derechos de acceso (12 bits).

Memoria virtual en x86

- ATRIBUTOS:
  - G (Granularidad):
    - G = 0 ⇒ Tamaño del segmento (límite) en bytes.
    - G = 1 ⇒ Tamaño del segmento (límite) en páginas de 4 KB.
  - D/B (Defecto/Grande):
    - Permite manejar conjuntamente segmentos del 286 con otros del 386 o superior.
  - AVL (Disponible):
    - Bit a disposición del programador.
  - P (Presencia):
    - P = 0 ⇒ Segmento ausente de MP.
    - P = 1 ⇒ Segmento presente en MP.

Memoria virtual en x86

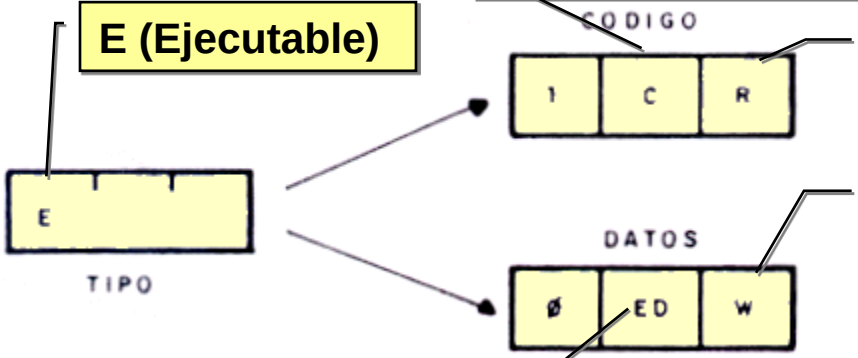
- DPL (Nivel de privilegio):
  - Nivel de privilegio del segmento al que hace referencia el descriptor (de 0 a 3).
- S (Tipo de segmento):
  - S = 0 $\Rightarrow$ Referencia un recurso especial del sistema.
  - S = 1 $\Rightarrow$ Segmento normal (código, datos o pila).
- A (Accedido):
  - Se pone a 1 cada vez que el procesador accede al segmento.
  - El SO lee y borra este bit periódicamente para implementar el algoritmo LRU.

Memoria virtual en x86

- TIPO: En los segmentos normales...
  - distingue si se trata de uno de código, de datos o de pila.
  - determina el acceso permitido (lectura / escritura / ejecución).

C (Ajustable):
C = 0 ⇒ Al ser accedido, CPL no cambia.
C = 1 ⇒ CPL toma el valor de DPL de este segmento (permite llamadas a rutinas de niveles inferiores).

E (Ejecutable)



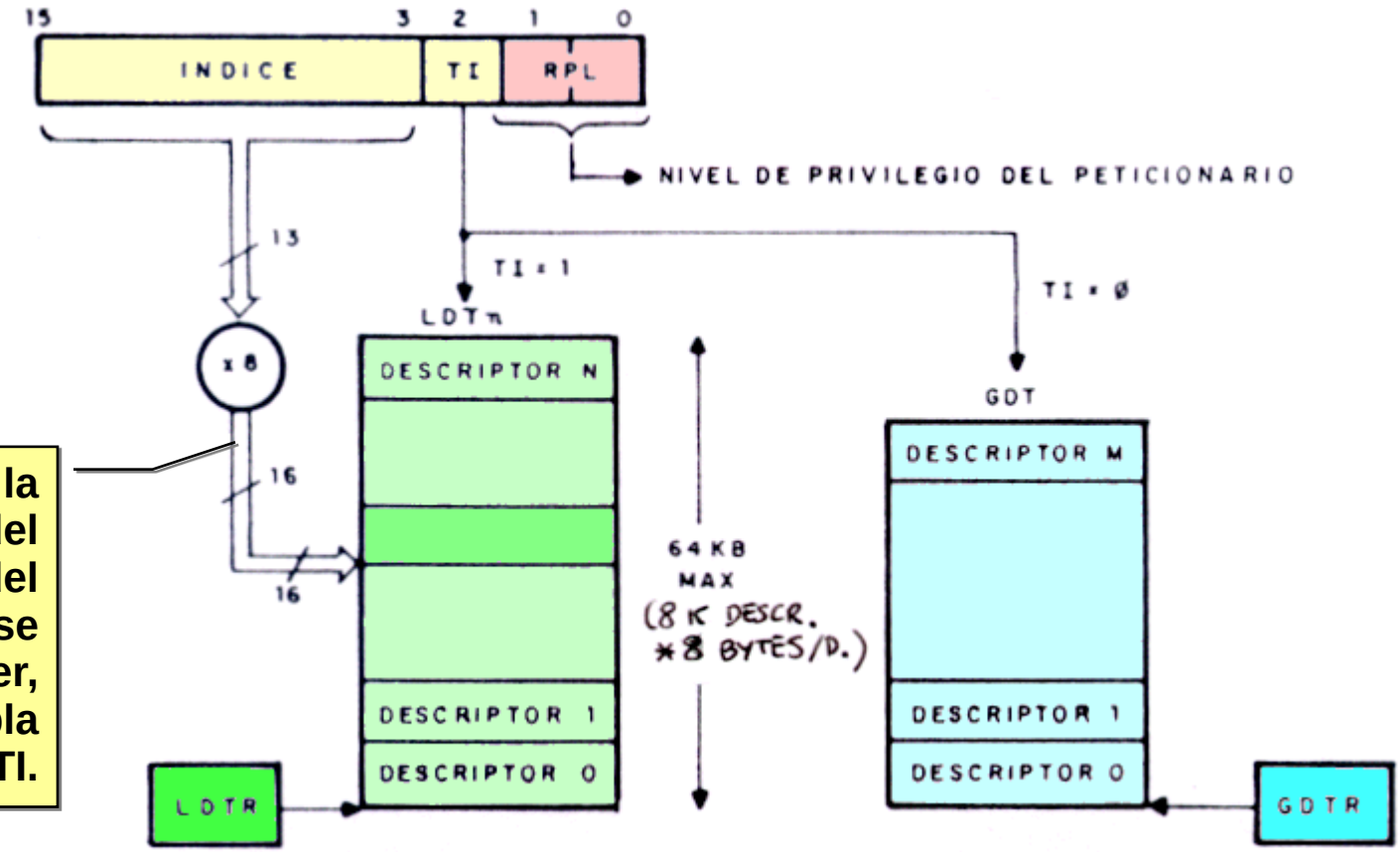
R (Legible):
R = 1 ⇒ El segmento de código puede leerse.

W (Escribible):
W = 0 ⇒ Sólo puede leerse.
W = 1 ⇒ Puede leerse y escribirse.

ED (Expansión decreciente):
ED = 0 ⇒ Segmento de datos normal (crece hacia arriba).
ED = 1 ⇒ Segmento de pila (crece hacia abajo).

Memoria virtual en x86

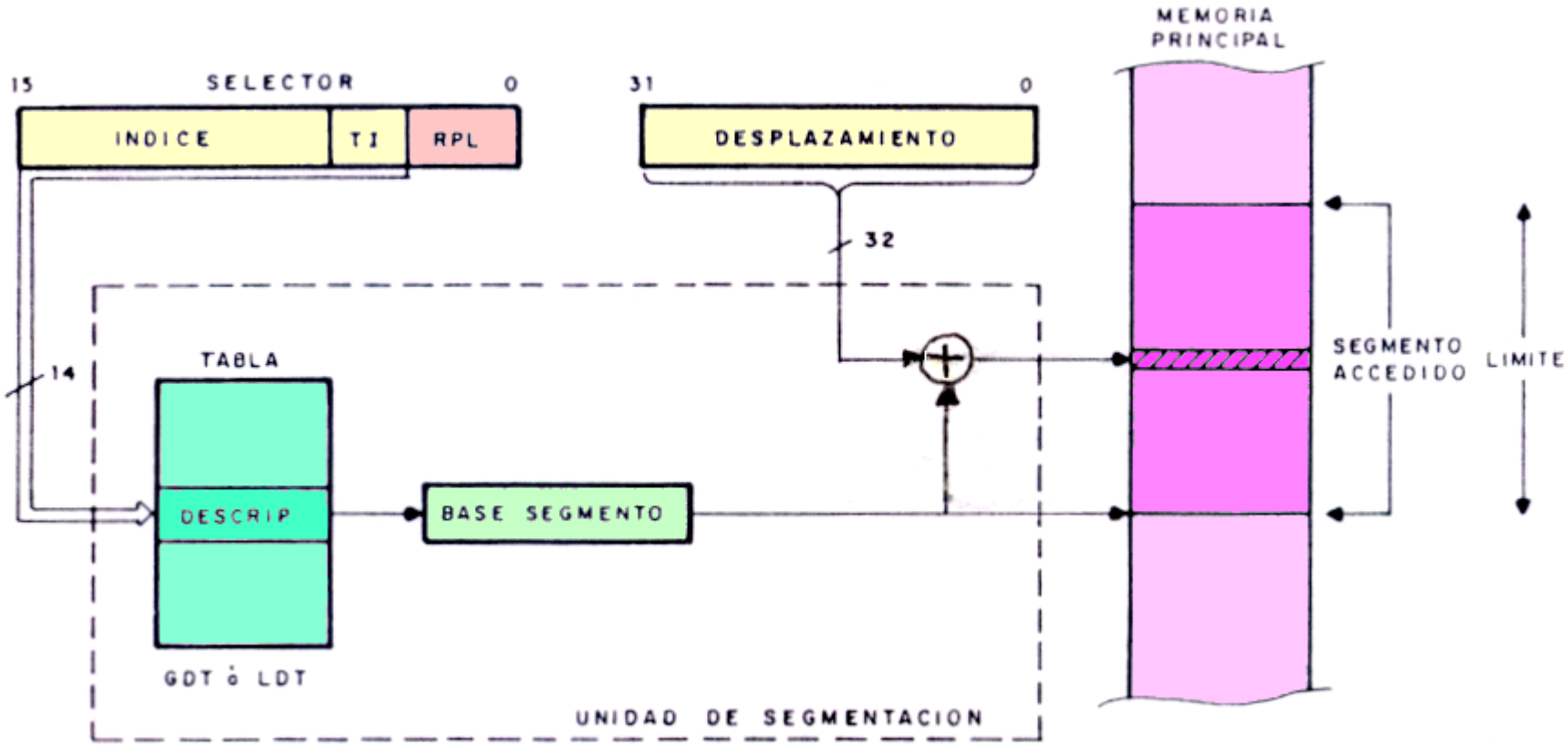
- Obtención de la dirección lineal o física por parte de la unidad de segmentación cuando está inhibida la paginación.



Seleccionan la dirección del descriptor del segmento al que se desea acceder, dentro de la tabla seleccionada por TI.

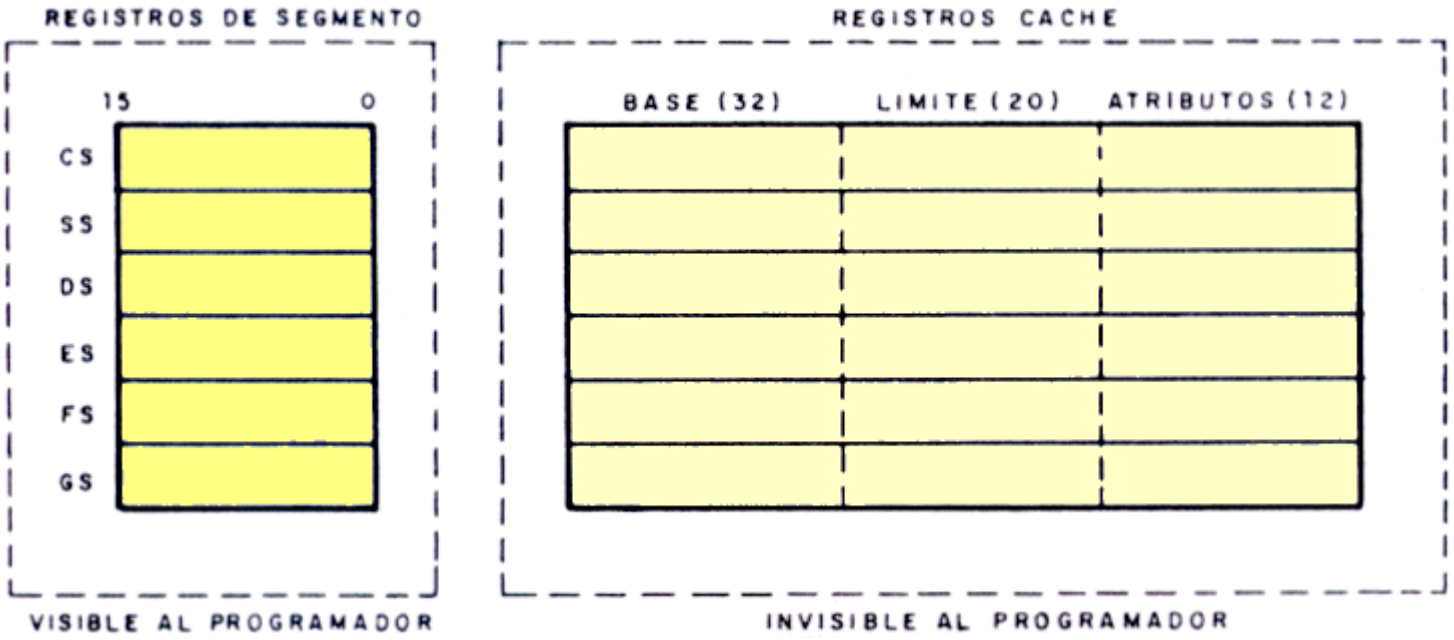
Memoria virtual en x86

- A partir del selector, y a través de las tablas de descriptores, la unidad de segmentación localiza la base del segmento, a la que suma el desplazamiento para obtener la dirección lineal:



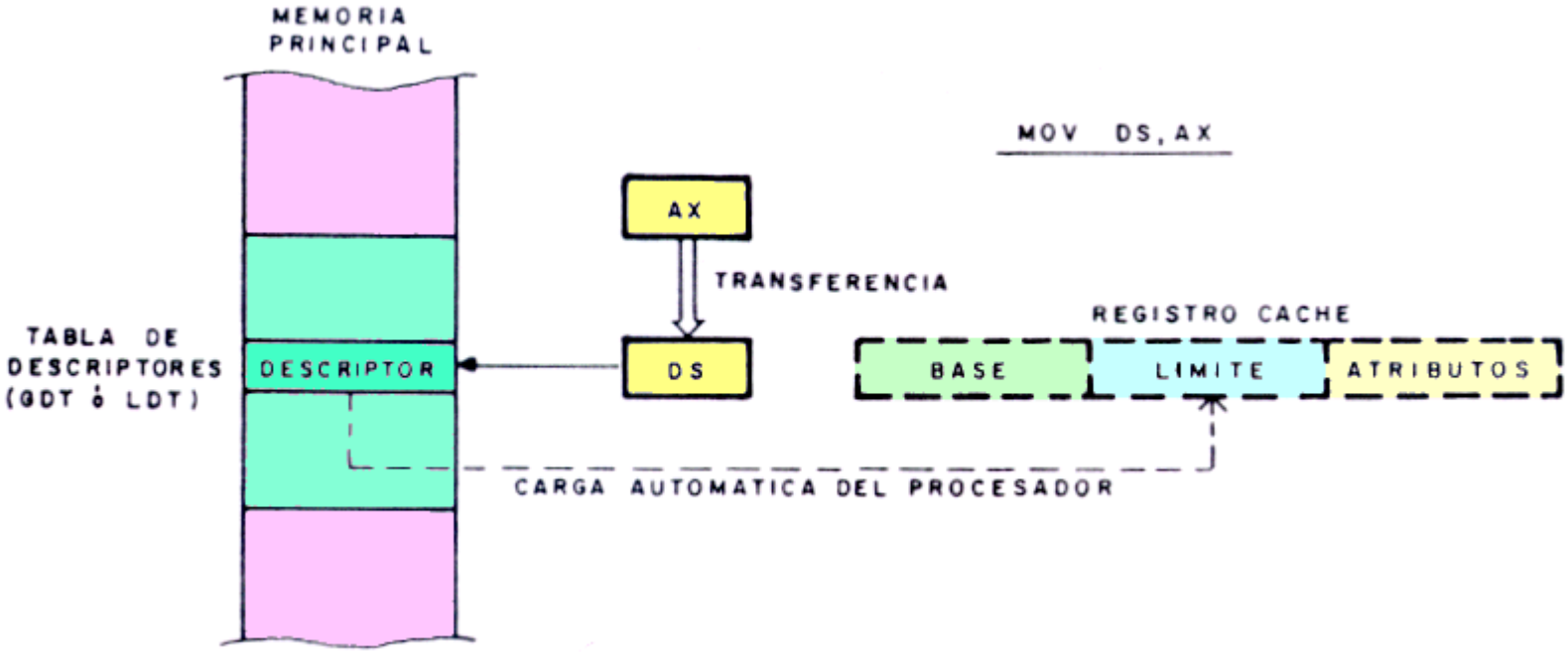
Memoria virtual en x86

- Registros de segmento y registros caché.
  - CS, SS, DS, ES, FS y GS contienen el campo “selector” de la dirección virtual.
  - Cada uno funciona asociado a un **registro caché (64 bits)** de alta velocidad de acceso.



Memoria virtual en x86

- Cuando **se carga un registro de segmento**, el contenido del descriptor al que hace referencia se lee de GDT o LDT y se almacena en el registro caché asociado.
- Mientras no se modifique un registro de segmento, **el procesador accede al segmento a través del registro caché ⇒ velocidad ↑** (sólo hay que acceder a la tabla de descriptores una vez para buscar los parámetros que definen el segmento).

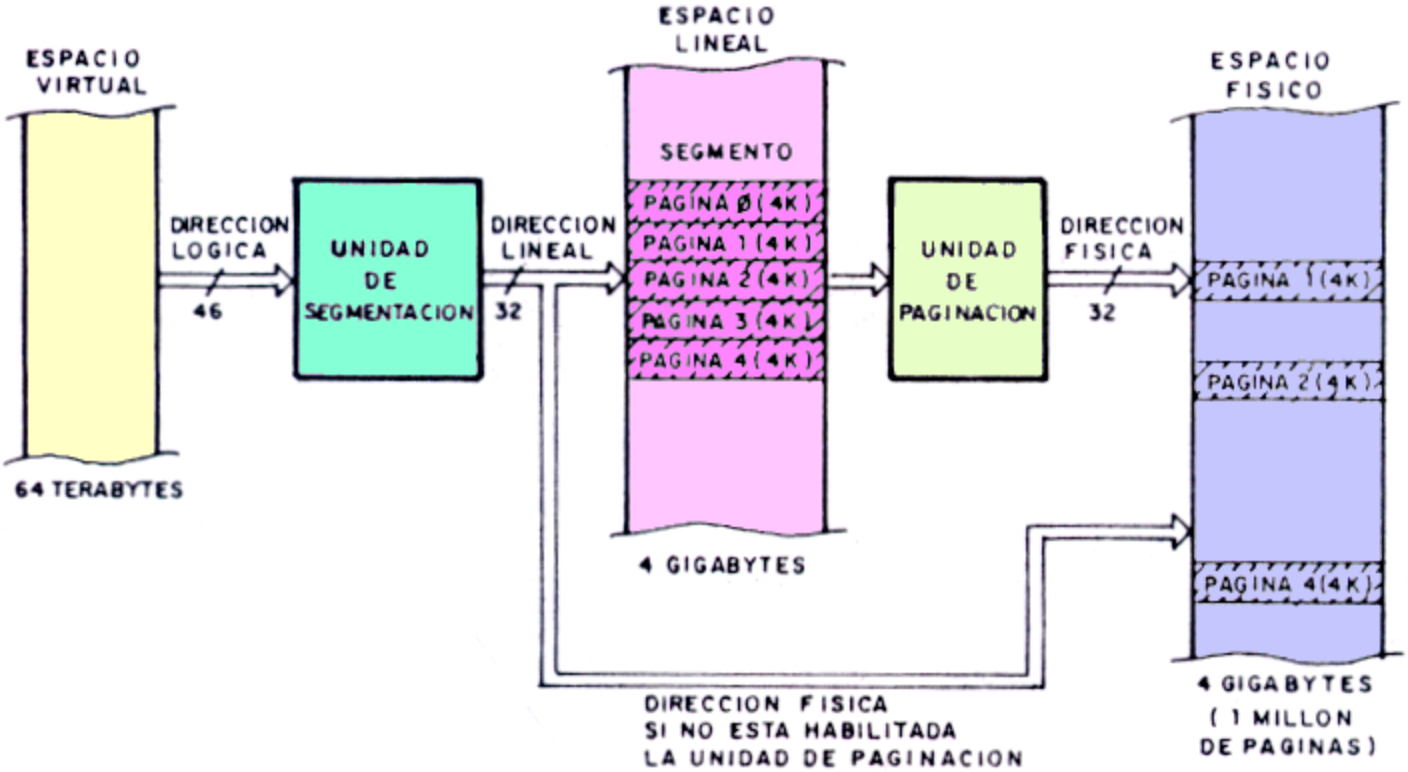


Memoria virtual en x86

- **Paginación**
  - El funcionamiento de la paginación es **optativo**.
    - Habilitación: poner a 1 el **bit PG** del registro de control CR0.
  - Se divide cada segmento del espacio lineal en páginas.
  - La unidad de paginación traduce de dirección lineal a dirección física, distribuyendo en la memoria física las páginas que se precisan en cada momento.
  - Tamaño máximo del espacio físico: 4 GB.
  - Tamaño de página: 4 KB.

} Hasta 1 M páginas.

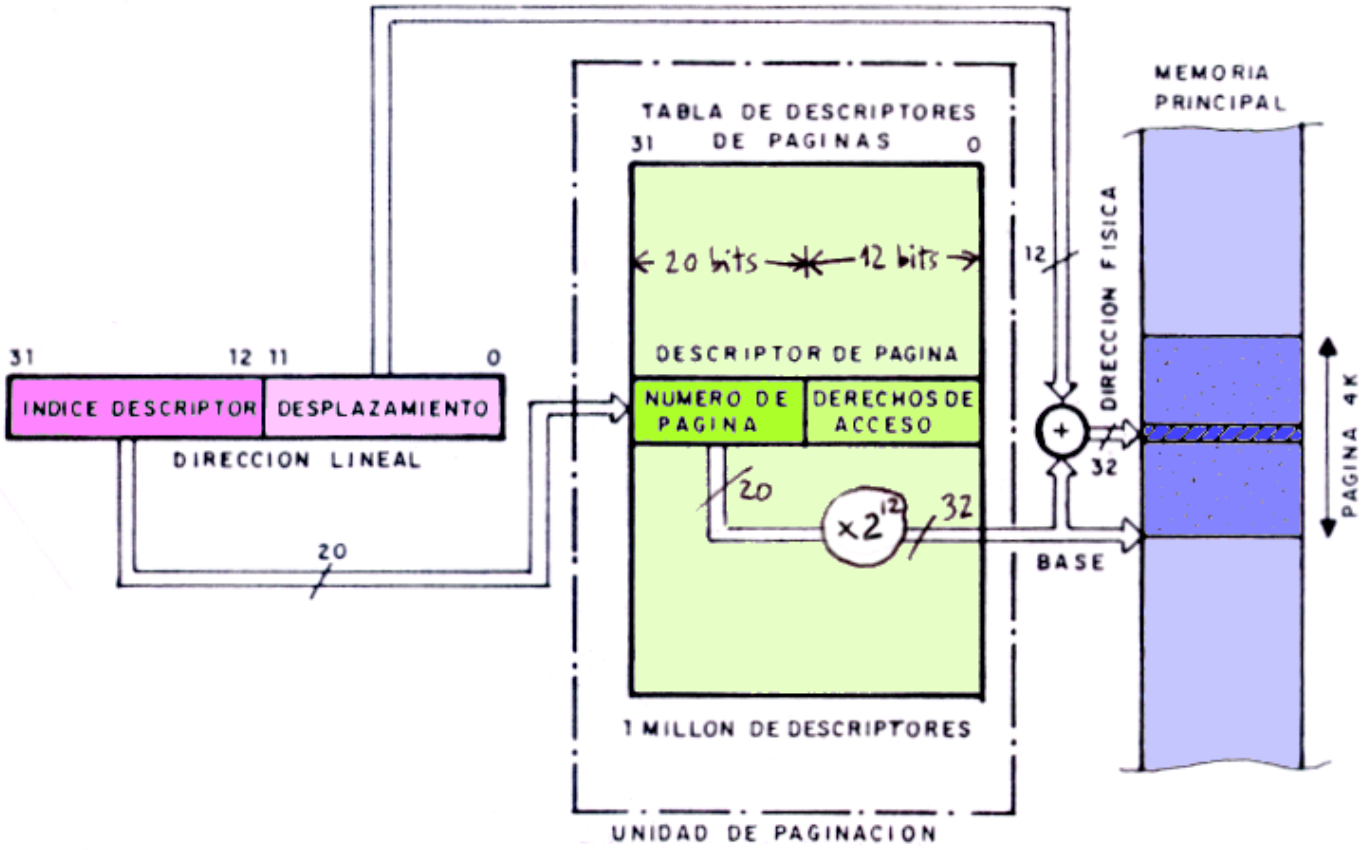
Memoria virtual en x86



Memoria virtual en x86

- Aparentemente, la unidad de paginación se comporta como una **tabla con 2^{20} descriptores de páginas**, que traduce de dirección lineal a física.
- Cada descriptor de página: 32 bits.

¡La tabla de páginas ocuparía 4 MB!

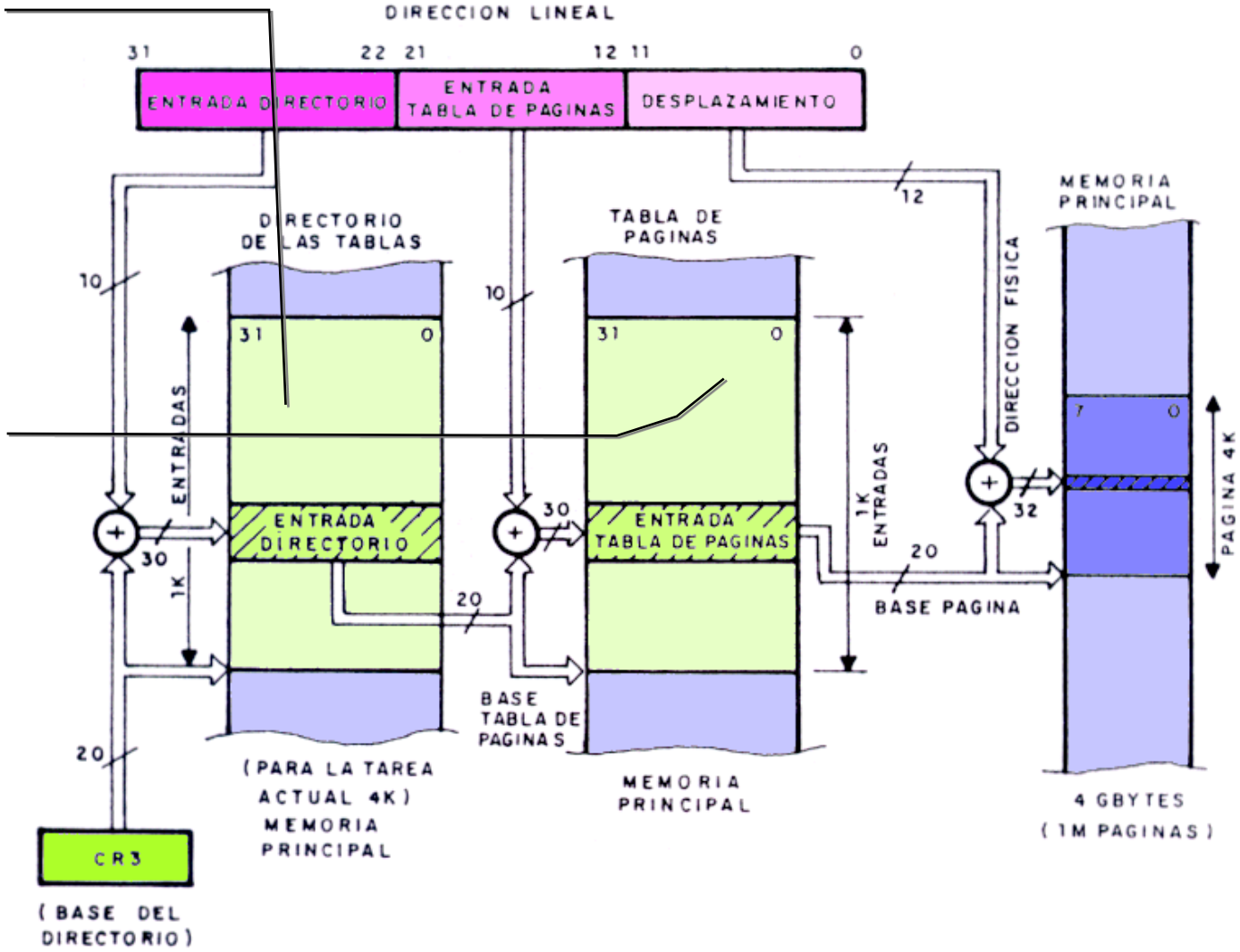


Memoria virtual en x86

- Traducción de direcciones a dos niveles:

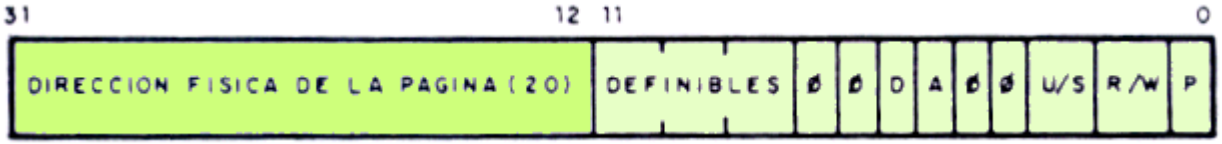
1. **Para cada tarea hay una tabla de 4 KB (1 K entradas de 32 bits) llamada directorio de tablas de páginas. Su base está cargada en el registro de control CR3.**

2. **Se selecciona una entrada del directorio, que contiene la dirección de la base de una página, que actúa como una segunda tabla de páginas (1 K entradas de 32 bits).**



Memoria virtual en x86

- Formato de las entradas del directorio y de las tablas de páginas:



  - **DIRECCIÓN FÍSICA** (20 bits):
    - 20 bits más significativos de la dirección base de la página de la siguiente estructura (los 12 bits de menor peso de esa dirección son 0).
  - **ATRIBUTOS** (12 bits).

Memoria virtual en x86

- ATRIBUTOS:
  - D (Sucio):
    - D = 0 ⇒ Página no modificada ⇒ se puede sobrescribir.
    - D = 1 ⇒ Se ha escrito en la página ⇒ actualizarla en MP antes de sobrescribirla.
  - A (Accedido): Se pone a 1 cada vez que se accede a la página. El SO usa este bit para el algoritmo de sustitución LRU.
  - U/S (Usuario/Supervisor):
    - U/S = 1 ⇒ Nivel supervisor. En la página puede haber todo tipo de instrucciones.
  - R/W (Lectura/Escritura):
    - R/W = 0 ⇒ Sólo se puede leer.
    - R/W = 1 ⇒ Se puede leer y escribir.
  - P (Presencia):
    - P = 0 ⇒ Fallo de página ⇒ se activa una rutina del SO que trae la página de disco a MP.
    - P = 1 ⇒ La página está cargada en la memoria física.

Memoria virtual en x86

- **Tabla de traducción de direcciones lineales (TLB).**
  - El mecanismo de traducción de direcciones en la paginación es lento, ya que se requieren dos accesos adicionales a memoria.
  - Solución: uso de un TLB, que guarda la traducción de direcciones lineales a físicas correspondientes a las 32 últimas páginas accedidas.
  - Se han comprobado tasas de acierto de más del 97%.

Memoria virtual en x86