Taller de Sistemas Operativos

Direccionamiento de Memoria

Agenda

- Arquitectura de memoria en x86 (IA-32).
- Direccionamiento en Linux.

Arquitectura de memoria en x86

- Sistema de administración de memoria de IA-32.
  - Segmentación: provee un mecanismo para separar el código, datos y el *stack.*
  - Paginación: provee un mecanismo para implementar un sistema de memoria virtual de paginación bajo demanda.
- En x86 segmentación está siempre disponible, mientras que paginación debe ser activado en el modo protegido.
- Existen tres espacios de direcciones.
  - Lógico.
  - Lineal.
  - Físico.

Espacios de direcciones

- Direccionamiento Lógico.
  - Las direcciones se componen de un segmento (*segment*) y un desplazamiento (*offset*).
- Direccionamiento Lineal (direccionamiento virtual).
  - Permite referenciar a 2^{32} = 4GB de memoria.
- Direccionamiento Físico.
  - Utilizado para acceder a las celdas de memoria RAM del sistema.
- Traducción de direcciones.

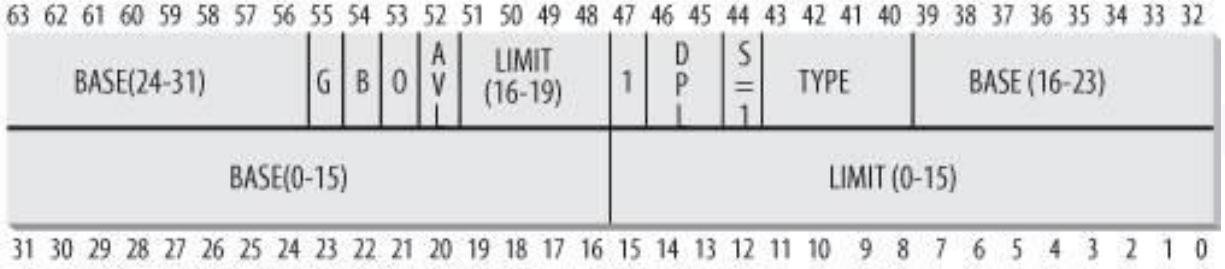
Unidad de Segmentación

- El direccionamiento en el espacio lógico es realizado a través de un selector de segmento y un desplazamiento.
- El selector está compuesto de 3 campos (16 bits):
  - Índice (13 bits): es el número de entrada en una tabla de descriptores de segmentos.
  - Indicador de tabla (1 bit): define si es sobre la tabla global (*GDT – Global Descriptor Table*) o local (*LDT – Local Descriptor Table*).
  - Nivel de privilegio (2 bits): real, protegido, usuario.
- Las tablas pueden contener hasta 8192 (2^{13}) entradas de descriptores.
- Existen dos registros que indican el comienzo de las tablas (`gdtr` y `ldtr`).

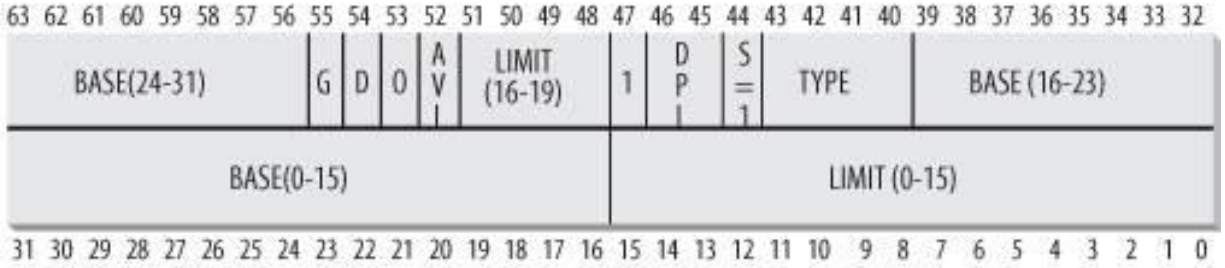
Descriptores de segmentos

- Hay diferentes tipos de segmentos.
  - Datos (DSD – Data) (GDT o LDT).
  - Código (CSD – Code) (GDT o LDT).
  - Estado de la tarea (*TSSD – Task State*) (GDT).
- Contienen la dirección base y el límite del segmento en el espacio lógico.

Data Segment Descriptor

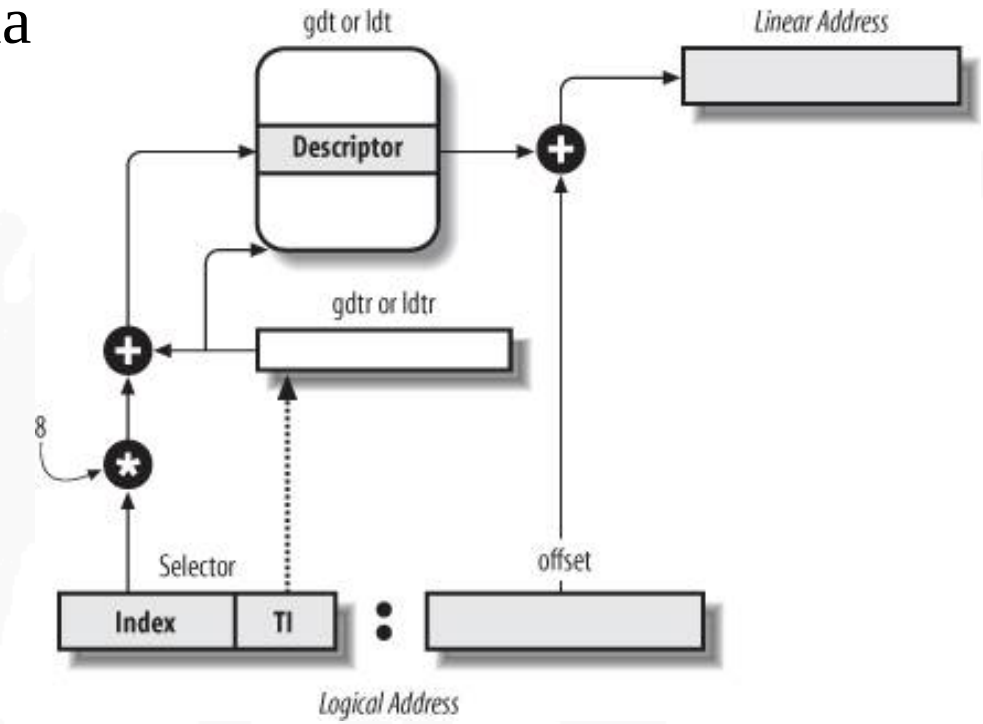


Code Segment Descriptor



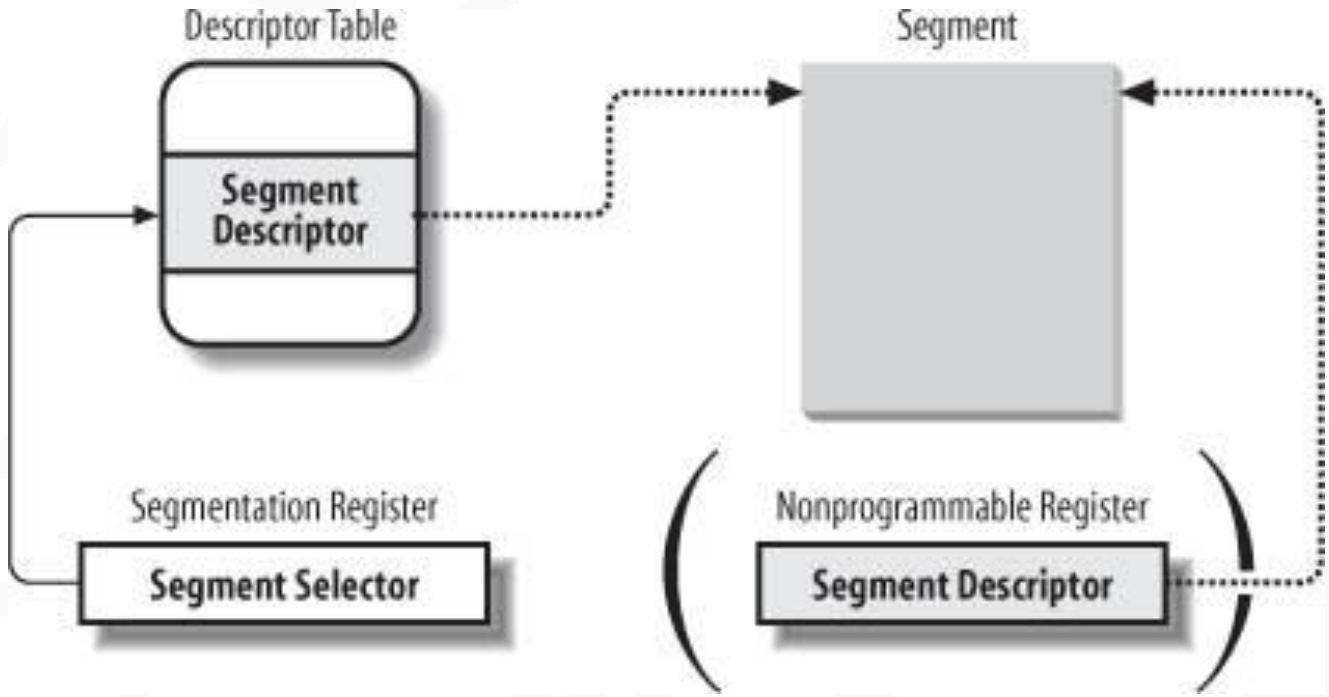
Traducción de lógicas a lineales

- El selector determina el tipo de tabla (`gdtr` o `ldtr`).
- Se selecciona el descriptor correspondiente al índice.
- A partir del descriptor se toma la dirección base y se le suma el desplazamiento.
- Es necesario 2 accesos a memoria



Traducción de lógicas a lineales

- Intel agrega registros no programables (cargados por la arquitectura) que contienen los descriptores de segmentos.
- Esto permite que la traducción se haga en un solo acceso.



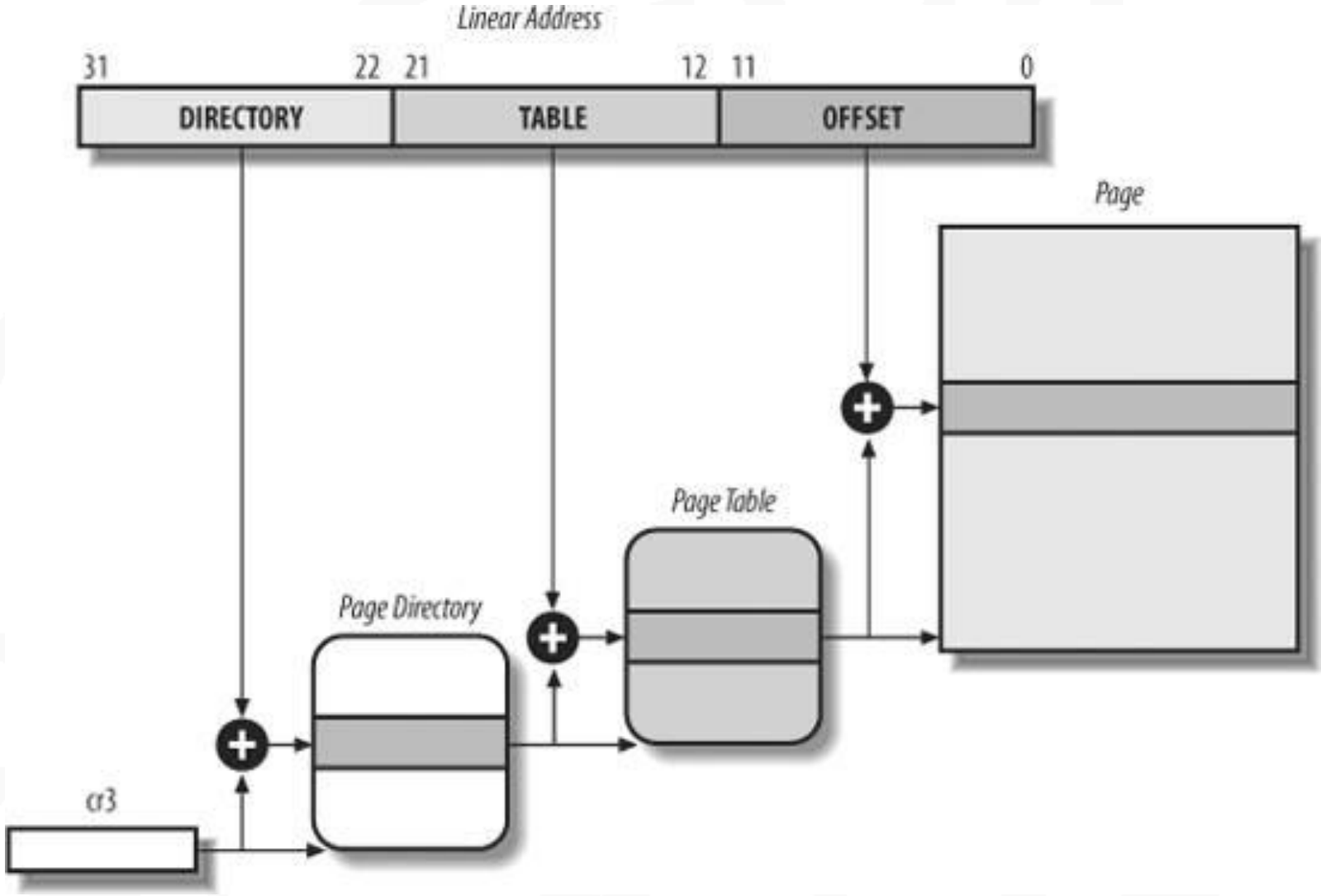
Unidad de Paginación

- Traduce direcciones lineales (virtuales) a físicas.
- Por defecto no está habilitado (si el sistema operativo lo desea, debe setear la bandera PG en el registro `cr0)`.
- El espacio de direcciones lineales está dividido en intervalos de largo fijo llamados páginas.
- La unidad de paginación abstrae a la memoria física como fraccionada en intervalos de largo fijo y del mismo tamaño que la página (marcos - *frames*).
- En IA32 las páginas son de 4KB de largo.
- Las direcciones se componen de 3 campos:
  - Selector de directorio (10 bits).
  - Selector de tabla (10 bits).
  - Desplazamiento (12 bits).

Traducción de lineales a físicas

- La traducción es llevada a cabo en dos etapas:
  - *Page Directory* (1024 entradas).
  - *Page Table* (1024 entradas).
- En total se puede direccionar:
  - `1024 * 1024 * 2^12 = 4GB`
- El registro base de la tabla de directorio (*Page Directory*) es el `cr3.`
- Los dos niveles permiten aprovechar mejor el espacio de memoria que tener una tabla global.
- Las tablas deben ser mantenidas por el sistema operativo.

Traducción de lineales a físicas



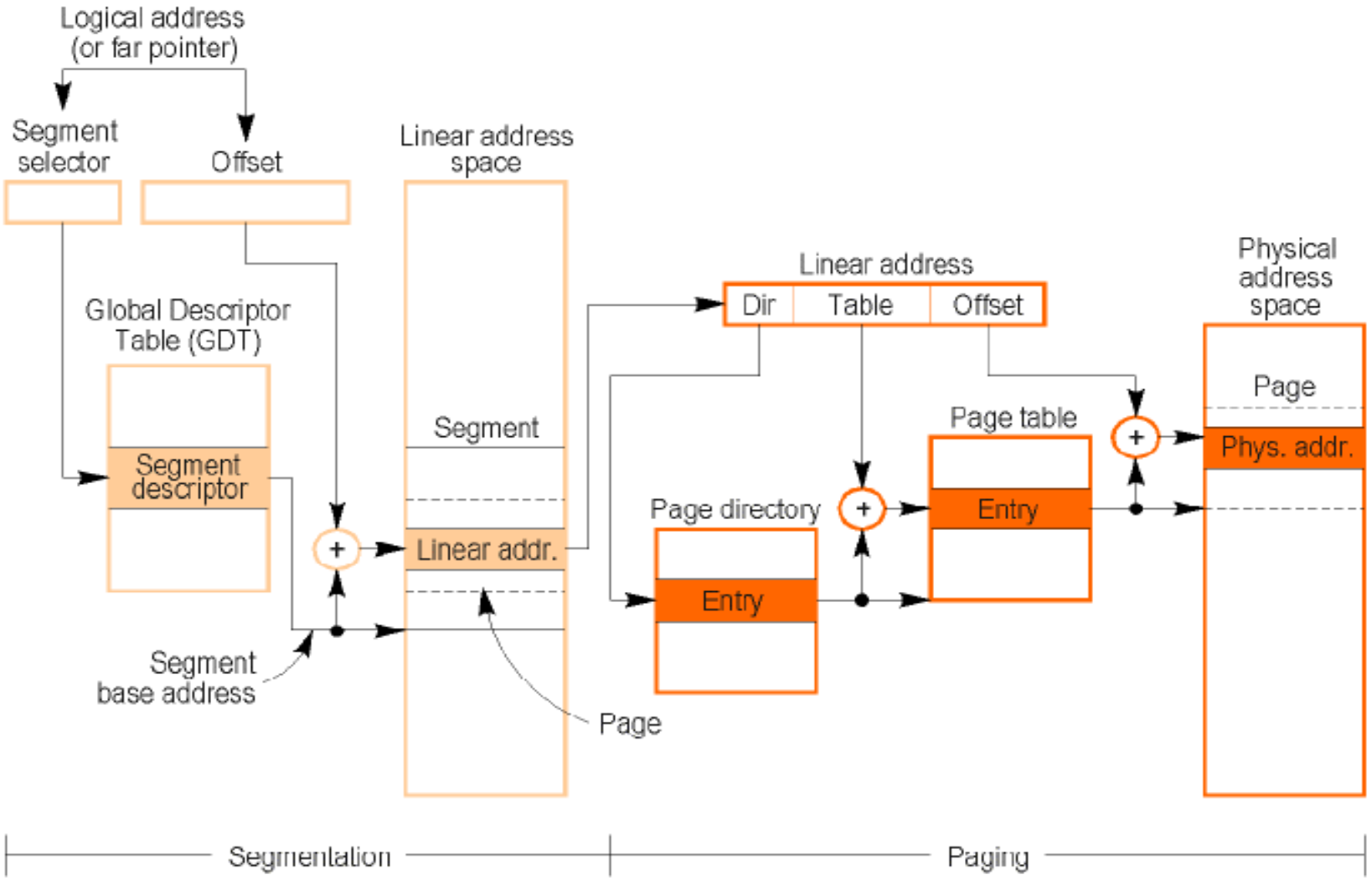
Entradas de tablas

Las tablas contienen los campos:

- *Present*: determina si está cargada en un *frame* en memoria.
- Campo de 20bits que contiene la dirección del comienzo del *frame* en memoria física.
- *Accessed*: se enciende cada vez que la unidad de página direcciona sobre ella.
- *Dirty*: solo para entradas de tabla de página, es encendida cada vez que se realiza una escritura sobre la página.
- *Read/Write*: contiene los permisos de acceso.
- *User/Supervisor*: nivel de privilegio de acceso.

- Si el *flag* de presencia en memoria no está encendido, la unidad de paginación carga la dirección lineal en el registro `cr2` y genera la excepción número 14 .

Resumen direccionamiento

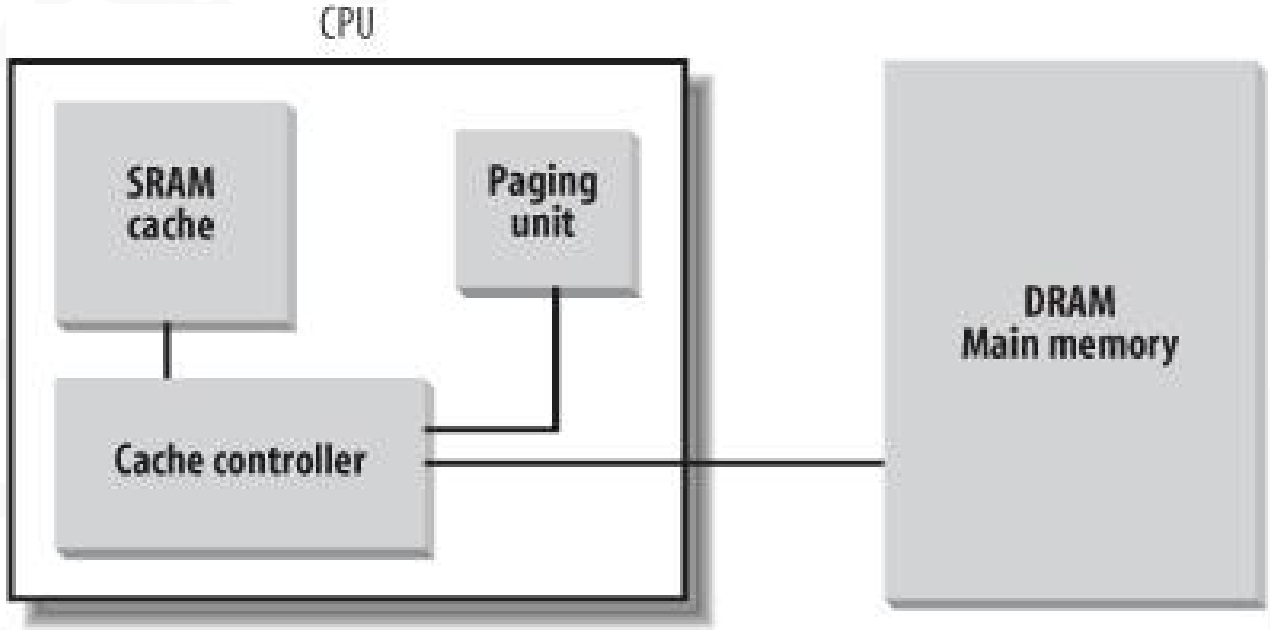


Memoria Cache

- Son introducidas para mejorar la diferencia de velocidad entre el procesador y la memoria.
- Utilizan el principio de localidad.
- Van de mapeo directo (*direct-mapped*) a totalmente asociativas (*fully associative*).
- Cada acceso a memoria es realizado a través de la controladora de memoria *cache*.
- Se verifica si la dirección física generada con los tags del conjunto asociativo correspondiente.
- Si coincide (*cache hit*), se distingue entre lectura y escritura.
  - Si es lectura se transfiere la línea de la memoria *cache* al procesador.

Memoria Cache

- Si no existe coincidencia (cache miss) se debe cargar el valor desde memoria RAM a la memoria cache.
- En sistemas multiprocesadores se utilizan estrategias de *snooping* para la consistencia entre las caches.



TLB – Translation Lookaside Buffer

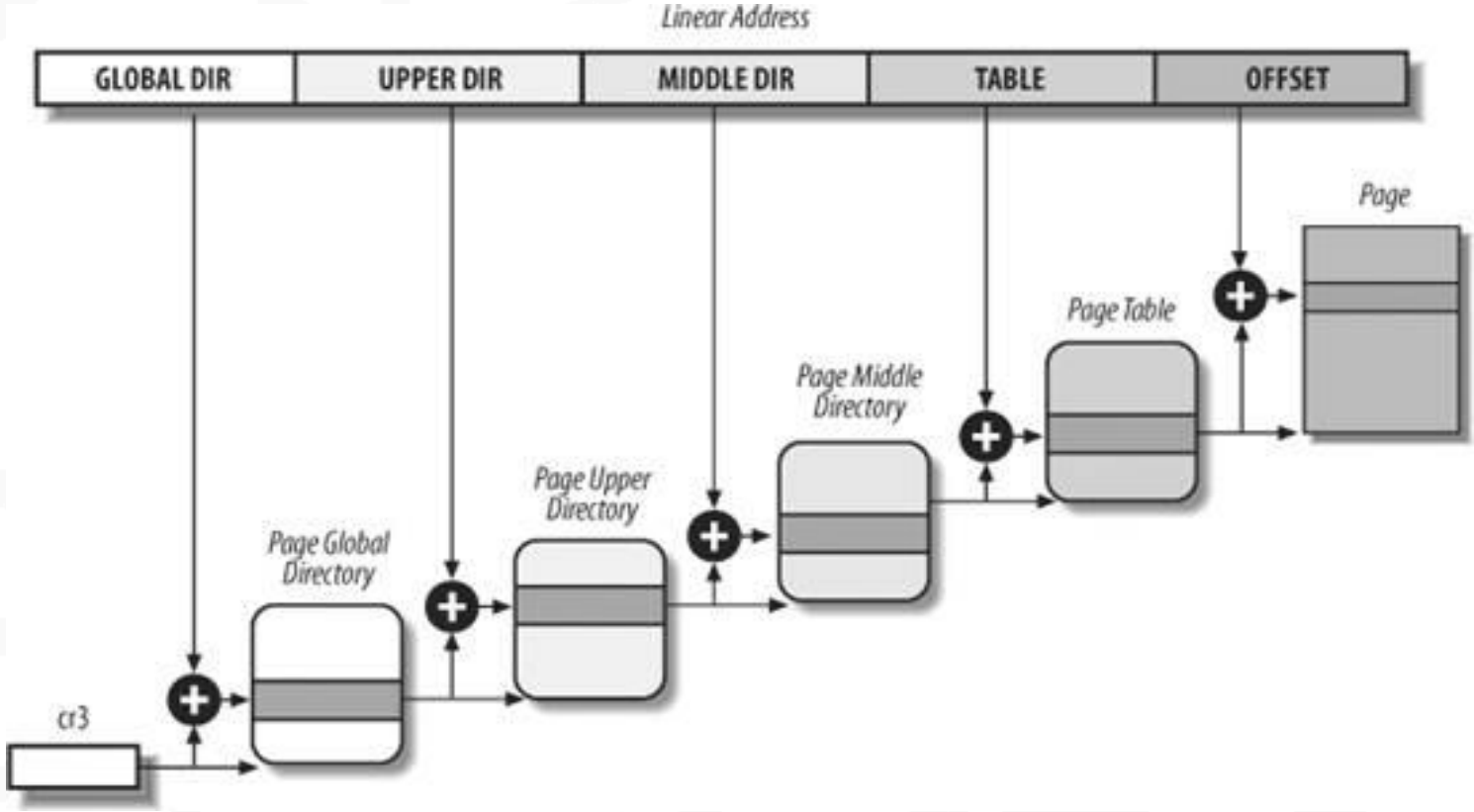
- Utilizadas para mejorar la eficiencia en la traducción de direcciones lineales.
- Cuando se realiza un acceso a memoria RAM a una tabla (directorio o de página), se guarda el valor en la TLB.
- Posteriores acceso, con el mismo `cr3`, permitirán mayor tiempo de respuesta en la traducción.
- El cambio del registro `cr3` inválida todas las entradas de la TLB.

Direccionamiento en Linux

- Linux utiliza la segmentación de forma limitada.
- Los procesos a nivel de modo usuario utilizan los mismos segmentos de código y de datos (*user code segment*, *user data segment*).
- Los procesos a nivel de modo protegido (núcleo) utilizan los mismos segmentos de código y datos (*kernel code segment*, *kernel data segment*).
- Se definen macros para los selectores de los segmentos (`__USER_CS`, `__USER_DS`, `__KERNEL_CS` y `__KERNEL_DS`).
- Los valores base y límite son los mismos para los cuatro segmentos:
  - Base: `0x00000000`.
  - Límite: `0xFFFFFFFF`.

Paginación en Linux

- A partir del núcleo 2.6.11 Linux adopto el mismo modelo de paginación para 32 y 64 bits en IA.
- El modelo dispone de 4 niveles de direccionamiento, en IA32 dos no son utilizados (*upper* y *middle*).



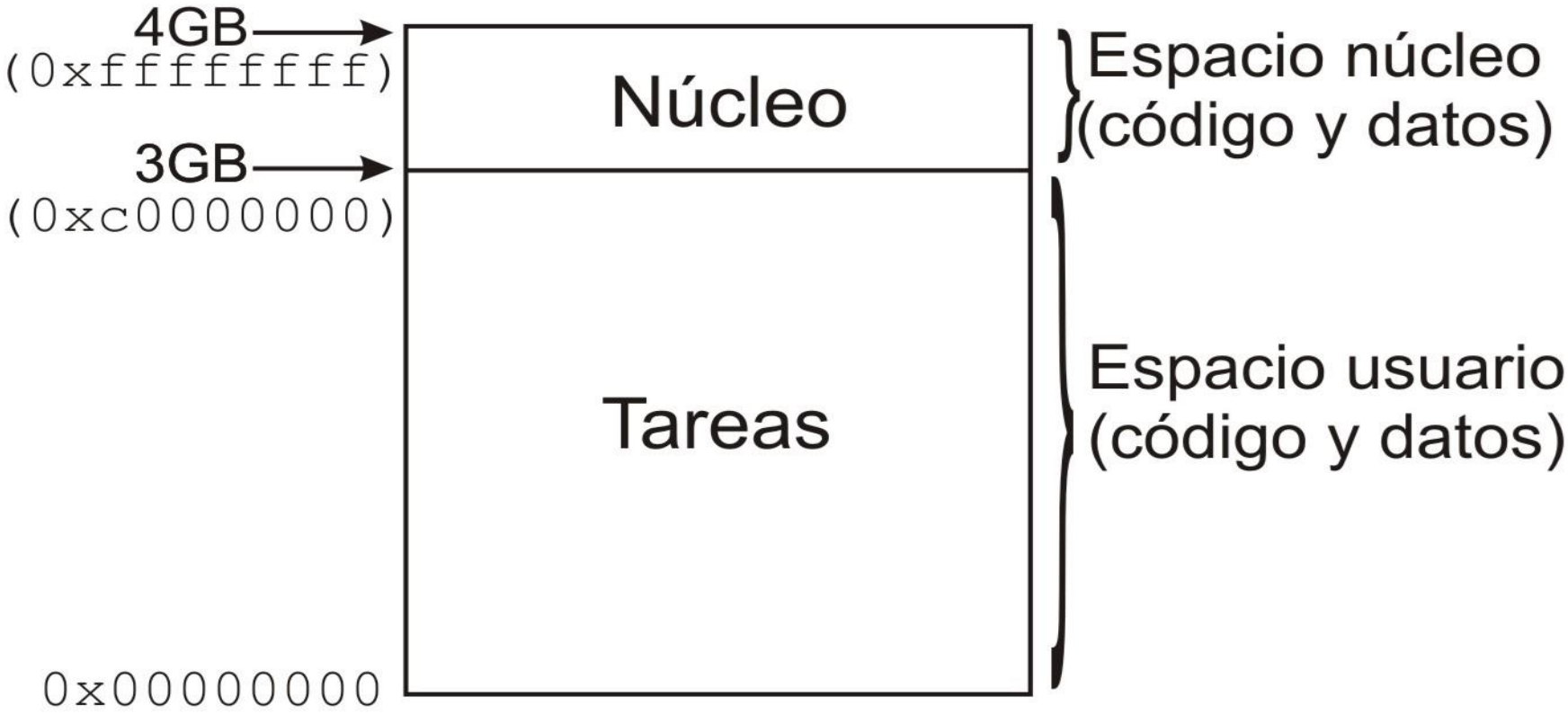
Paginación en Linux

- El núcleo activa la paginación en el proceso de inicialización del sistema.
- Linux reserva *frames* exclusivamente para el código del núcleo y sus estructuras de datos, que nunca son "*swapeadas"* a disco.
- Cada proceso tiene su conjunto de tablas.
- El registro base (`cr3 – PTBR`) es cargado al realizar el cambio de contexto.
- Al cargar el registro se invalidan todas las entradas de la TLB.

Tabla de páginas de los procesos

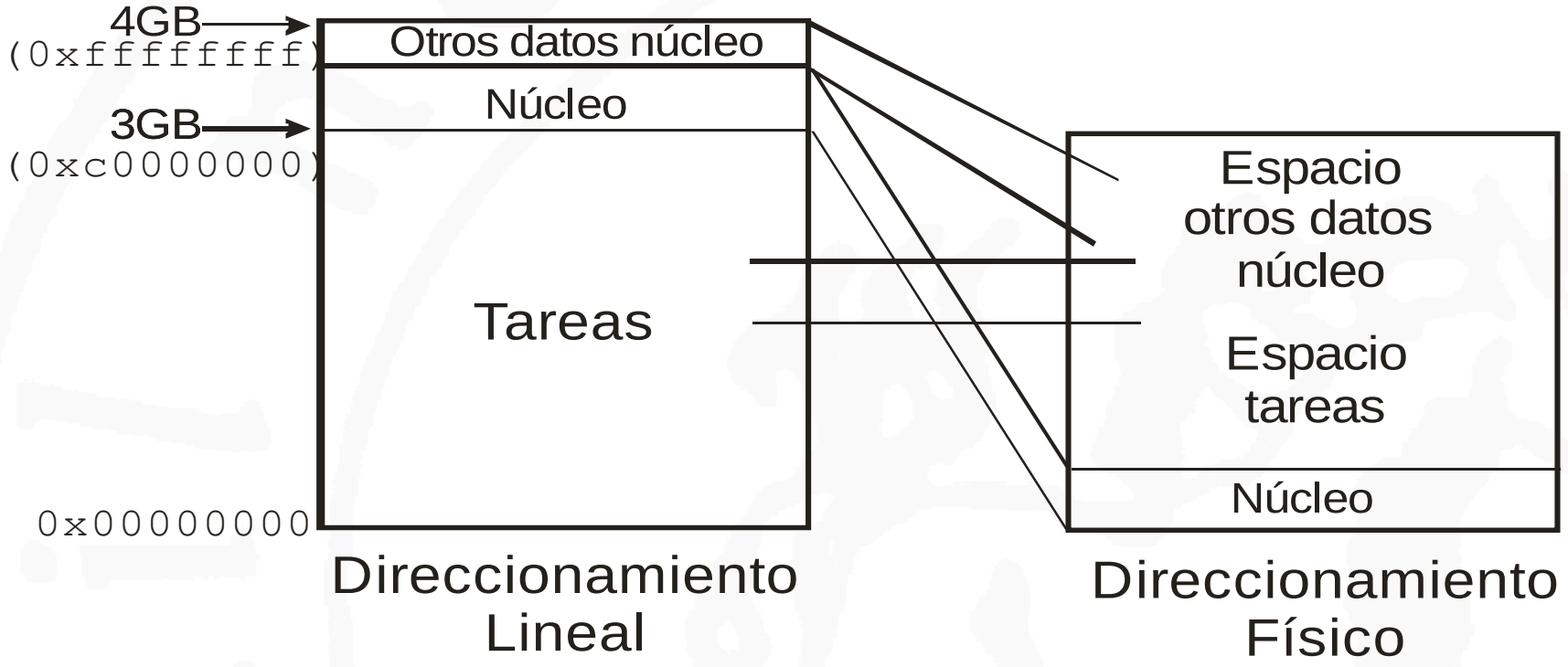
- El espacio de direcciones lineales de los procesos es dividido en dos partes:
  - Las direcciones lineales desde `0x00000000` a `0xBFFFFFFF` son accedidas tanto por el núcleo y los usuarios (primeras 768 entradas de la *global directory*).
  - Las direcciones lineales desde `0xC0000000` a `0xFFFFFFFF` son accedidas solamente por el núcleo (últimas 256 entradas de la *global directory*).
- Esta forma de direccionamiento permite que los procesos tengan un máximo de 3GB de direccionamiento lineal.
- Las primeras 768 entradas son propias del proceso.
- Las 256 restantes son las mismas para todos los procesos del sistema.

Tabla de páginas de los procesos



Mapeo lineal del núcleo

- El núcleo mapea en forma lineal la memoria física en el espacio lineal a partir de la dirección `0xC0000000`.
- La dirección `X` mapea a la dirección física `X - 0xC0000000`.



La memoria cache en Linux

- Linux favorece la obtención de *cache hits* a través de dos formas:
  - Los campos más usados en las estructuras de datos que define son agrupados para que entren en la misma línea de la memoria *cache.*
  - Cuando es necesario asignar en memoria una gran cantidad de datos, el núcleo intenta distribuirlo de forma que todas las líneas del *cache* sean utilizadas.

La TLB en Linux

- Cuando se escribe el registro `cr3` el hardware. automáticamente inválida todas las entradas de la TLB.
- Por lo tanto, Linux trata de no actualizar el registro `cr3` en los siguientes casos:
  - Cuando hay un cambio de contexto entre dos procesos que utilizan el mismo conjunto de tablas.
  - Cuando se realiza un cambio de contexto entre un proceso regular y un *thread* del núcleo.
- Al realizar un llamado a sistema y pasar a ejecutar en modo protegido, también no es necesario actualizar el registro.