Sistemas Operativos
2º Curso – Grado en Ingeniería Informática

Tema 3:

Gestión de memoria

José Antonio Gómez Hernández, 2016.

Gestión de memoria: índice

- Repaso (Tema 2 de FS):
  - Conceptos generales de gestión de memoria: hardware y software
  - Mecanismos de gestión: Paginación y segmentación
- Paginación multinivel
- Paginación bajo demanda
- Gestión de memoria en Linux:
  - Gestión de memoria del kernel
  - Gestión de memoria para procesos
  - Cachés del sistema

0.

Repaso

Conceptos básicos de gestión de memoria y paginación

Repaso: recursos

▷ Tema 2 de Fundamentos del Software.

▷ Libro de W. Stallings:

- Elementos hardware: Apartados 1.5 y 1.6.
- Gestión de memoria: Apartados 7.1 a 7.3
- Memoria virtual: Apartado 8.1, epigrafes de “Proximidad y memoria virtual” y “Paginación” (sin ver paginación a dos niveles, ni la tabla invertida de páginas, si el “Búfer de traducción anticipada”). El restp de epígrafes no es necesario.

Elementos hardware

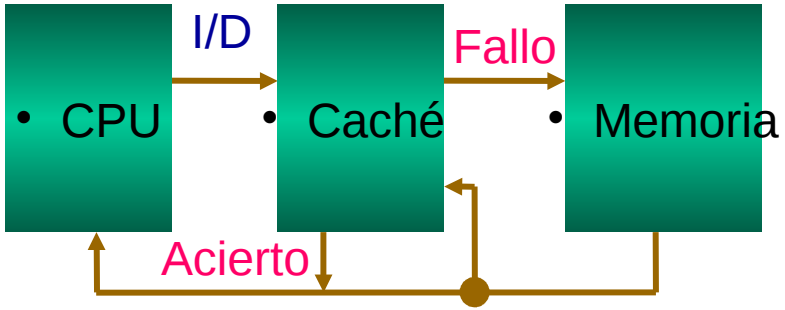
▷ Los computadores mantienen una jerarquía de memoria para mejorar el acceso de la CPU a memoria.

▷ Elementos importantes de la misma son las *cachés*: memorias intermedias que mantienen instrucciones/datos previamente accedidos y que son más rápidas que la RAM.

▷ Para cachés, definimos:

- *Acierto de cache*: la instrucción/dato buscado está en ella.
- *Fallo de caché*: la instrucción/dato no esta en la misma.

▷Esquema:



Cachés

▷ Las cachés “funcionan” porque explotan las **localidad** de las referencias del código, datos, y pila de los programas.

▷Tipos de localidad:

- *Espacial*: si un item es referenciado, las direcciones próximas a él tienden también a ser referenciadas.
- *Temporal*: si un item referenciado, tiende de nuevo a ser referenciado en breve.

▷Cuando se usan cachés, el coste de acceder a memoria viene dado por el Tiempo de Acceso Efectivo (TAE):

▷ `TAE = p*ta + (1-p)*tf`

▷donde: `p` =probabilidad de acierto; `ta` =tiempo de acceso si hay acierto; `1-p` = probabilidad de fallo, y `tf` =tiempo de acceso si hay fallo.

Requisitos

▷El SO asigna memoria a los procesos para su ejecución, garantizando:

- *Protección*: Un proceso no accede a memoria de otro. Diferentes módulos del programa deben tener diferentes permisos de acceso.
- *Compartición*: De datos/código entre procesos. Permite el ahorro de memoria.
- *Reubicación*: En sistemas multiprogramados, un programa debe poder cargarse en diferentes zonas de memoria.

Diseño

- El SO debe esconder la **Organización física** (jerarquía de niveles, estructura no lineal) de la memoria física.
- para que el usuario tenga una **visión lógica** simple de la memoria como una matriz lineal. Además permitirá la estructuración de un programa en módulos.

Espacios lógicos y físicos

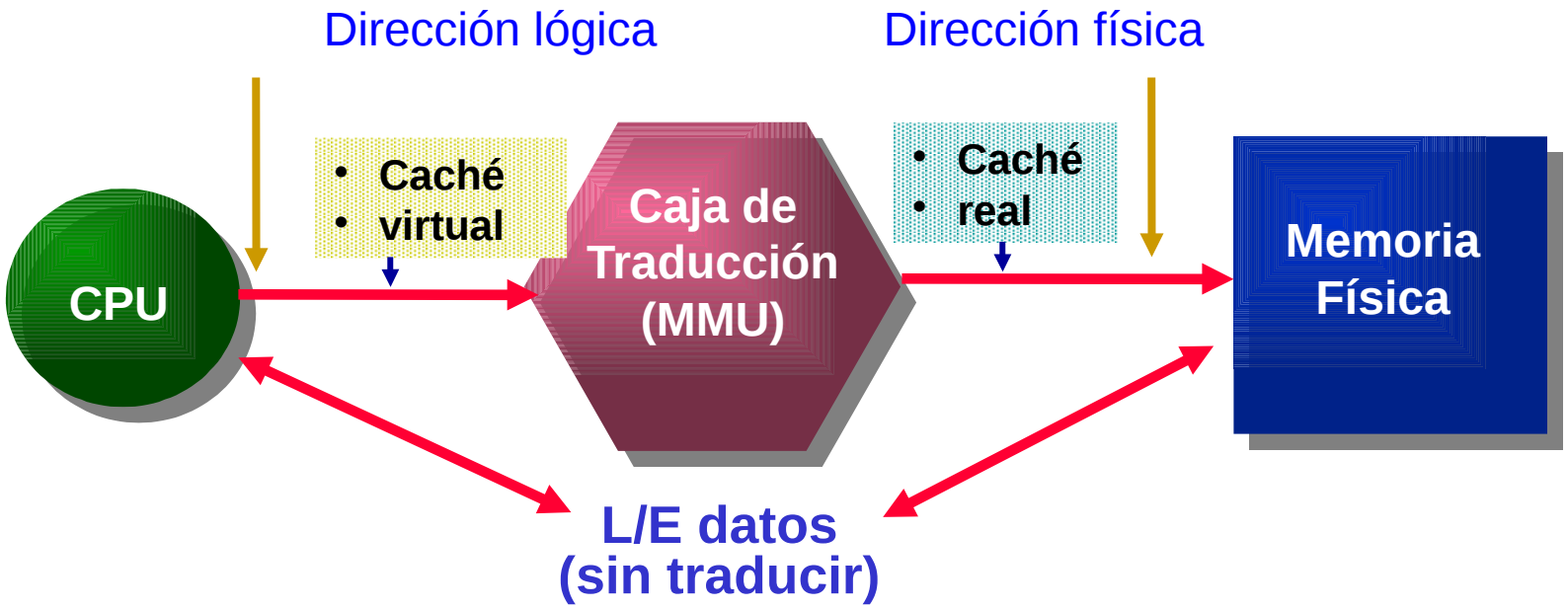
▷ La necesidad de poder reubicar un programa en memoria hace necesario separar el espacio de direcciones generadas por el compilador, **espacio lógico o virtual**, del **espacio físico** en el que se carga, el espacio de direcciones físicas (en RAM).

▷ Denominamos:

- **Dirección lógica** - la generada por la CPU; también conocida como virtual.
- **Dirección física** - dirección que se pasa al controlador de memoria.

Traducción de direcciones

▷ La separación de espacios me obliga a realizar una traducción de direcciones:



Unidad de Gestión de Memoria

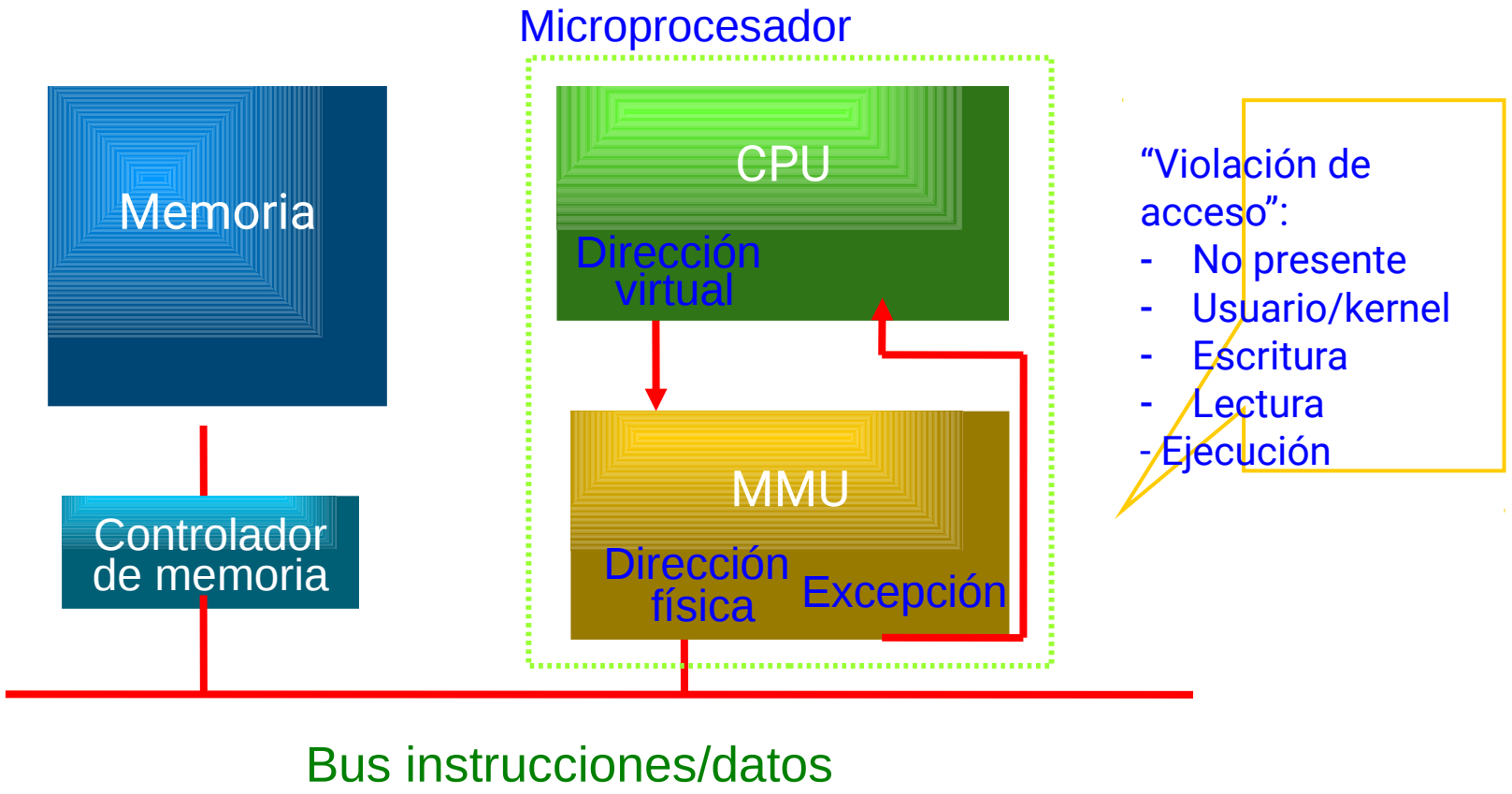
▷ La MMU (Memory Management Unit) es el dispositivo hardware que:

- Traduce direcciones virtuales en direcciones físicas.
- Implementa la protección.

▷ El hardware determina la forma en la que el SO gestiona la MMU.

▷ La forma de la MMU dependerá del esquema de gestión de memoria implementado en hardware. En el esquema más simple, contendrá un registro de reubicación que almacena el valor a sumar a cada dirección generada por el proceso de usuario al mismo tiempo que es enviado a memoria.

MMU: funcionamiento



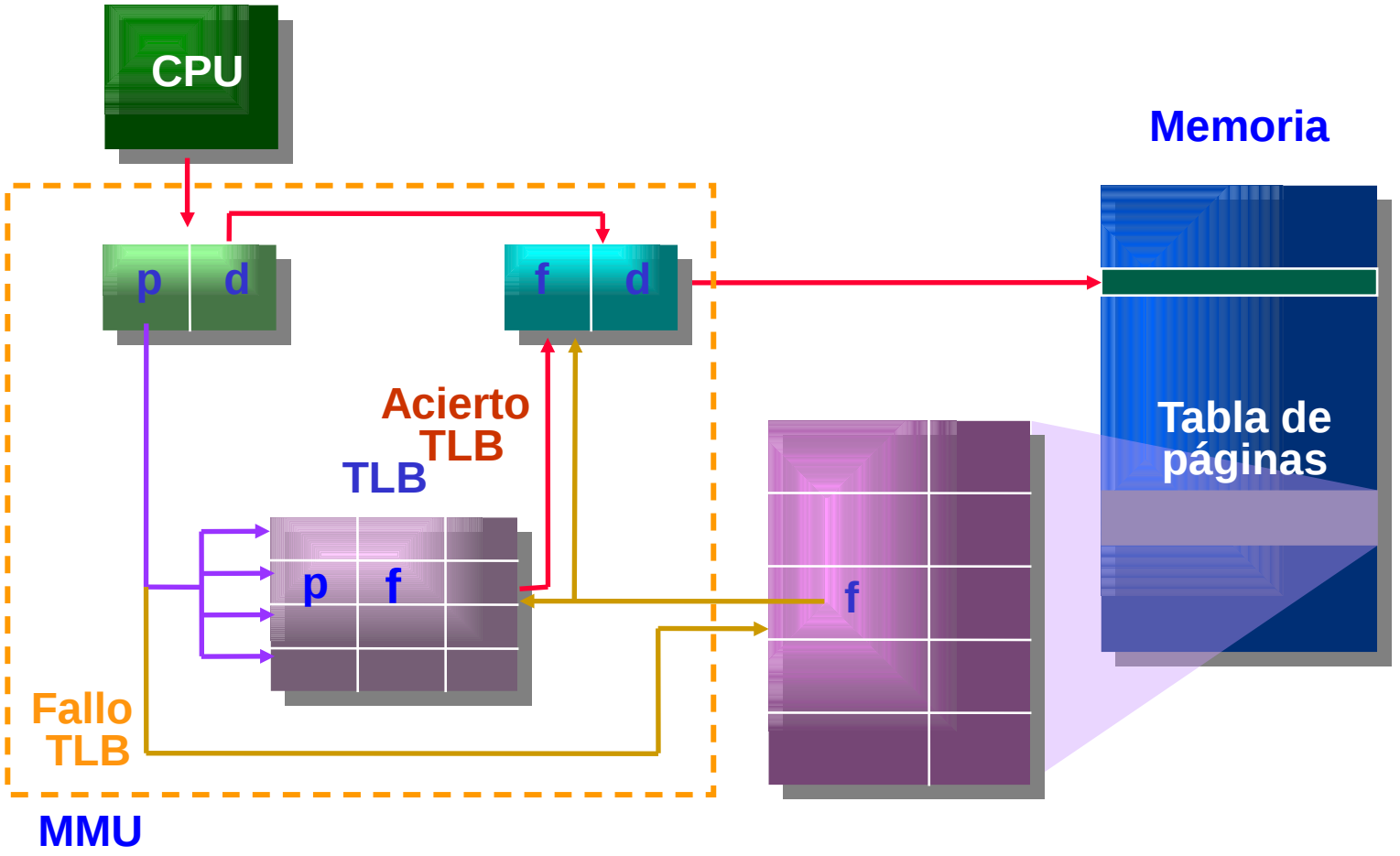
Fragmentación

- Denominamos **fragmentación de memoria** ha aquella fracción de la misma que no es asignable debido al propio mecanismo de gestión de memoria.
- Los sistemas de gestión de memoria han evolucionado con el objetivo principal de reducir la fragmentación de memoria.
- Al desacoplar los espacios lógicos de los físicos, podemos hacer que el espacio de direcciones de un proceso no sea continuo, podemos trocearlo, reduciendo así la demanda de memoria contigua.
- Los SOs actuales suelen utilizar paginación como esquema básico de gestión de memoria, si bien, dependiendo del procesador, deben también utilizar segmentación. Por ejemplo, los procesadores Intel implementan segmentación como esquema básico de gestión de memoria (protección:modos de funcionamiento del procesador) y opcionalmente se puede activar o no la paginación.

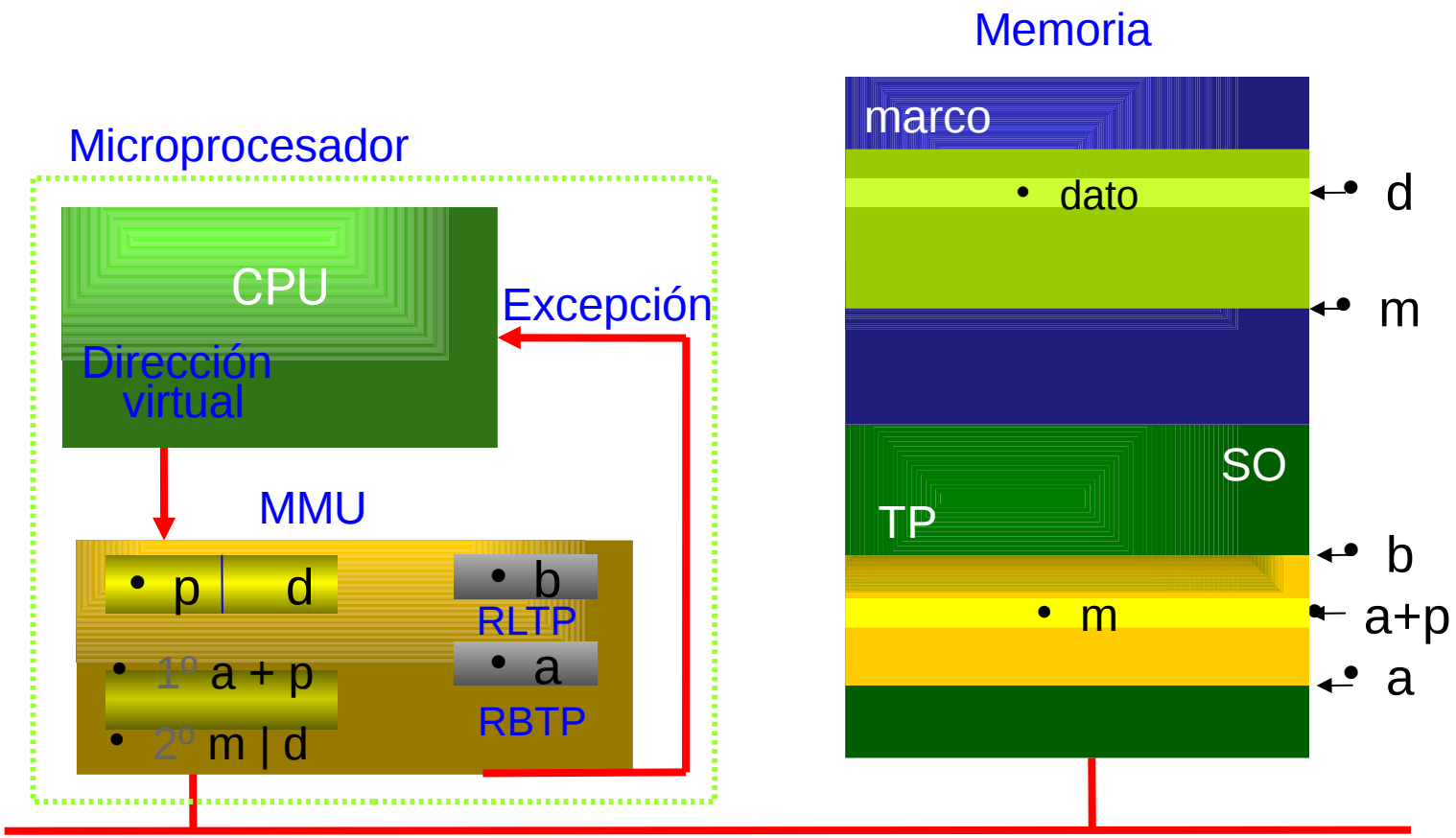
Paginación

- La MMU “divide” el programa en bloques del mismo tamaño, denominados **páginas**, para cargarlos en bloques de memoria principal del mismo tamaño, denominados **marcos**.
- Esto permite evitar que la búsqueda de un hueco de RAM para cargar una página sea una asignación dinámica.
- El SO mantiene la pista de cuales son los marcos que contienen las páginas de un programa mediante una estructura de datos por proceso denominada **tabla de páginas** (TP). Esta estructura tiene una **entrada de TP** (PTE) por cada página del proceso, donde cada entrada indica cual es la dirección base de memoria principal del marco que la contiene. También contiene información de protección de la página. Si una entrada no es válida en el espacio de direcciones se desactiva el *bit de validez* de la PTE correspondiente.

Paginación: traducción



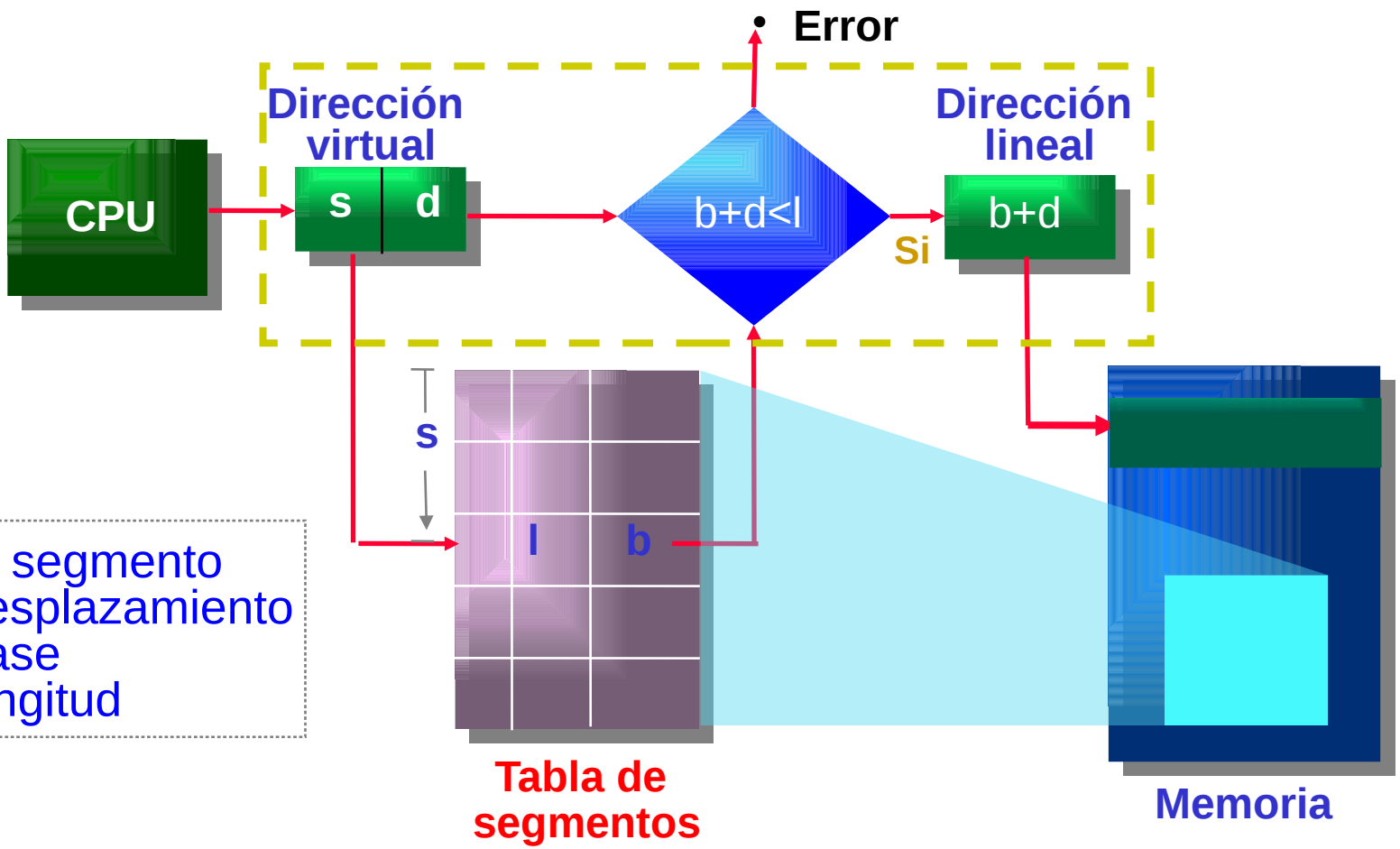
Paginación: ejemplo de traducción



Segmentación

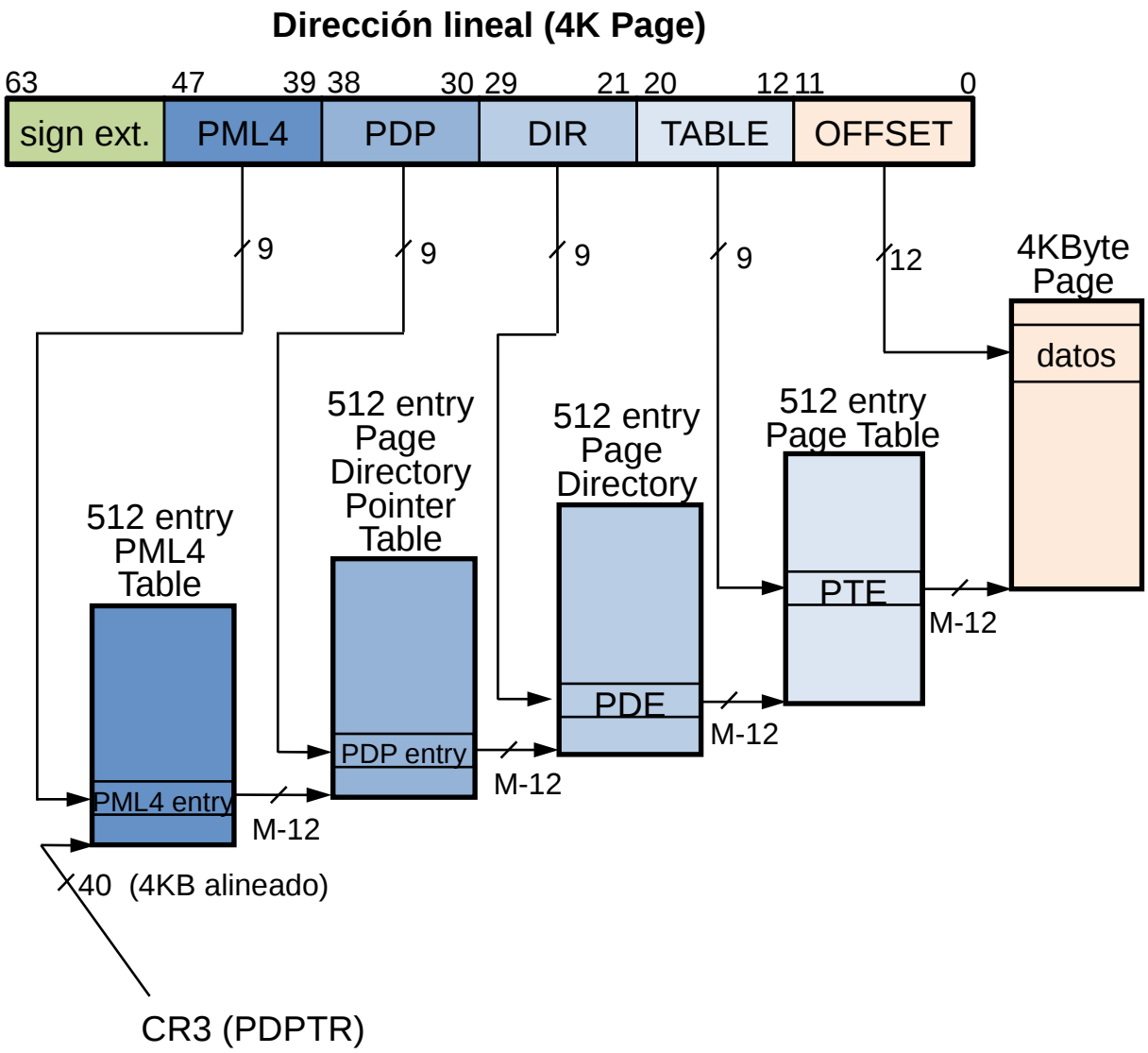
- Troceamos el programa en unidades lógicas de programación (procedimientos, pilas, código, datos, tabla de símbolos, etc.) denominadas segmentos.
- Cada segmento suele tener una tamaño diferente del resto.
- Ahora, una dirección lógica es una tupla:
- <número_de_segmento, desplazamiento>
- La **Tabla de Segmentos** aplica direcciones bidimensionales definidas por el usuario en direcciones físicas de una dimensión (lineales). Cada entrada de la tabla de segmentos tiene los siguientes elementos:
  - *base* - dirección física donde reside el inicio del segmento en memoria.
  - *límite* - longitud del segmento.

Segmentación: esquema

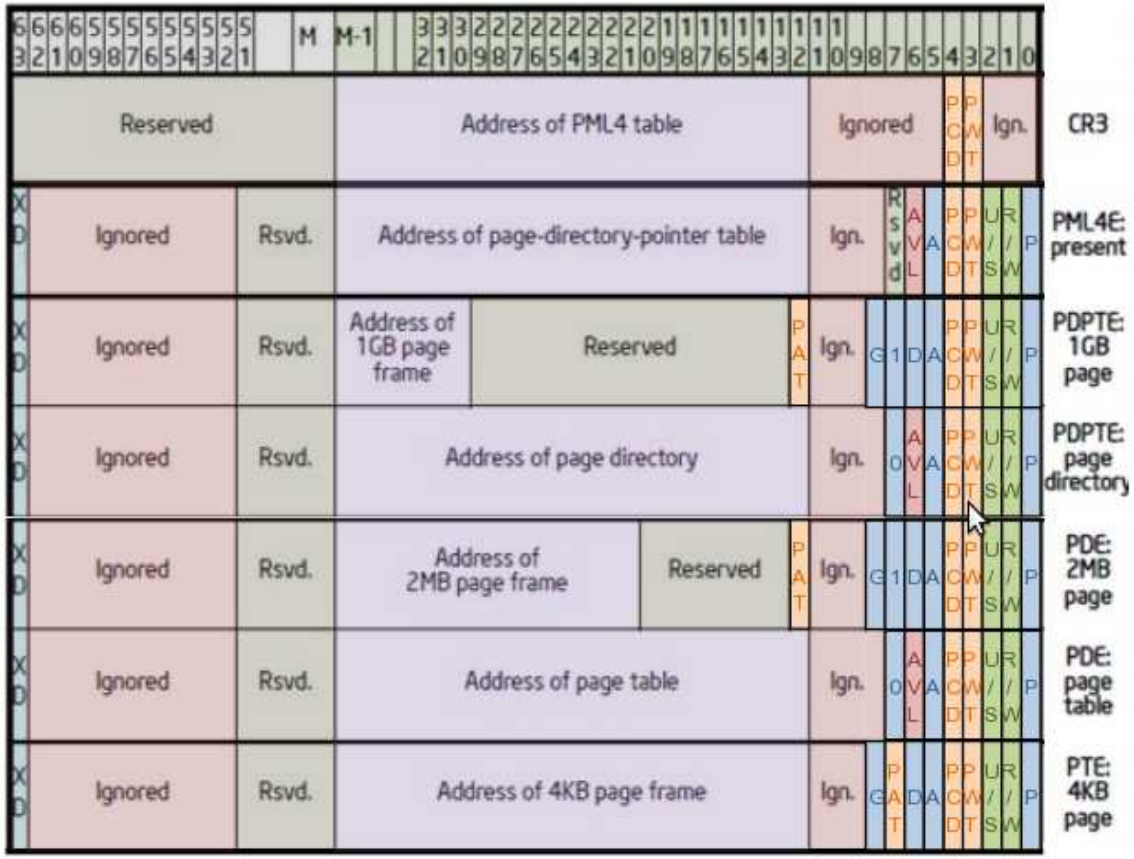


- s = nº segmento
- d = desplazamiento
- b = base
- l = longitud

Paginación Intel x64: páginas 4KB



Entradas de Tablas de Pg en x64



- Memoria Virtual:
  - Presente (P)
  - Accedida (A)
  - Sucia (D)
  - Gobal (G)
  - Tamaño pg. (bit 7–0:4KB;1:4MB)
- Protección:
  - Escritura (RW)
  - Usuario/supervisor (U/S)
  - Ejecución (XD)
- Caché:
  - Página Write-Though (PWT)
  - Cache pg. Deshabilitada (PCD)
  - PAT (atributo indice PT)(PAT)