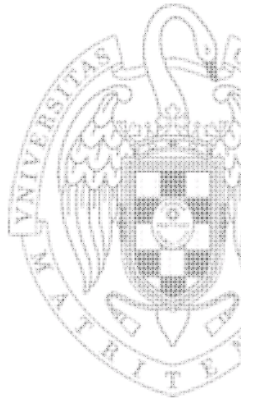


Módulo 4. Jerarquía de memoria:

Memoria Virtual

ec

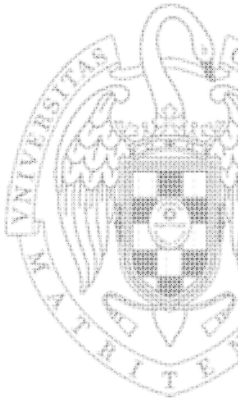
Bibliografía

- Capítulo 2 de Hennessy & Patterson (5th ed.)
- Apéndice B de Hennessy & Patterson (5th ed.)
- Appendix L: Historical Perspectives with References de Hennessy & Patterson
  - http://booksite.mkp.com/9780123838728/historical.php

Por qué se necesita la Memoria Virtual

- Espacio de direcciones del procesador superior al de la Memoria Principal
- Espacio real de almacenamiento superior al de la Memoria Principal
- Espacio utilizado por un *programa* (proceso) muchas veces superior al de la Memoria Principal
- Más de un *programa* (proceso) diferente puede estar utilizando la Memoria Principal a la vez
- No es necesario que un mismo *programa* (proceso) se almacene siempre en la misma posición de Memoria Principal

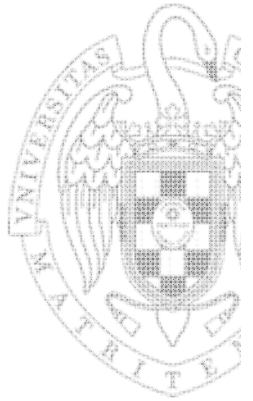
Motivaciones históricas (i)



- Motivaciones iniciales:
  - Eliminar las cargas de programación
  - Compartir memoria de manera eficiente y segura
    - Hoy en día este segundo aspecto es el más importante
- Eliminar cargas de programación:
  - Antiguamente si un programa era más grande que la MP el programador era el encargado prepararlo para su ejecución
    - Dividir el programa en partes
    - Identificar partes excluyentes
    - Estas partes se cargan o no bajo control del programador
      - Asegurando que nunca se intenta acceder a una parte no cargada
      - Las partes cargadas nunca superan el tamaño de la memoria
  - La memoria virtual se inventa para reducir el trabajo de los programadores
    - Se encarga de gestionar automáticamente el intercambio de información entre los dos niveles

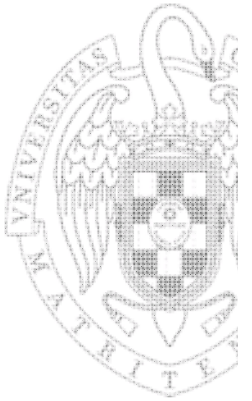
Kilburn et al. [1962] propuso la gestión del intercambio de información entre dos niveles de memoria: Atlas de la Universidad de Manchester
El IBM 370 la introdujo en 1972, Translation Lookaside Buffer

Motivaciones históricas (ii)

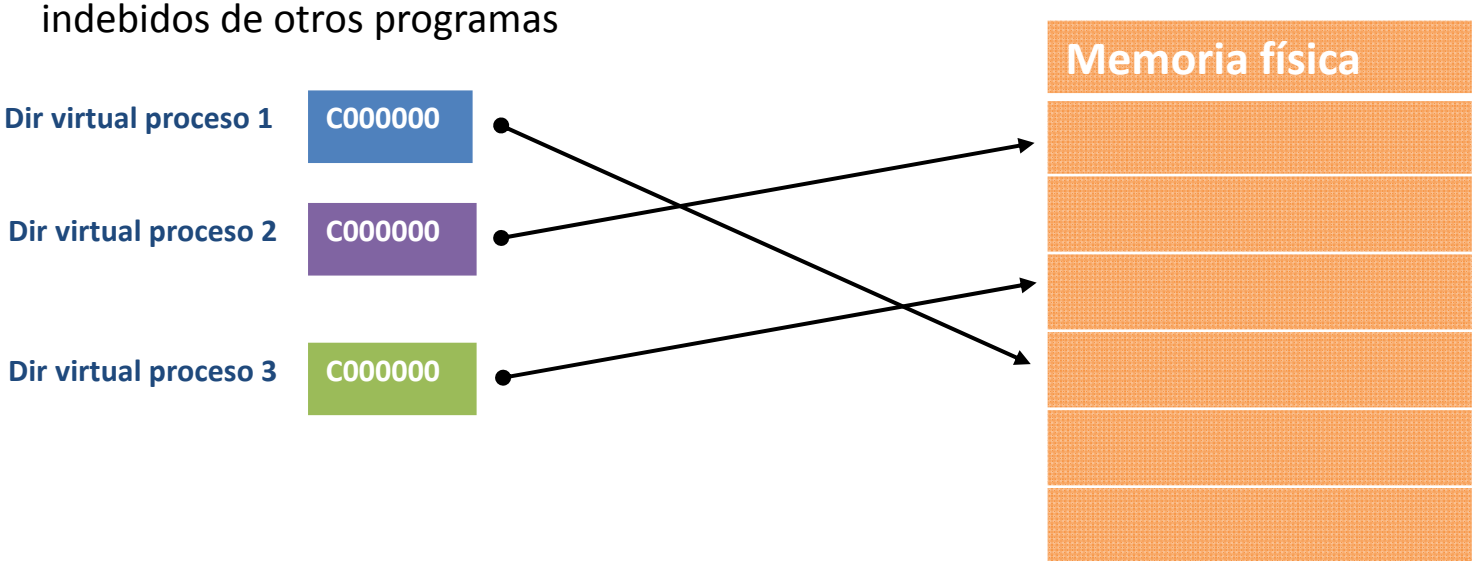


- Compartir memoria de manera eficiente y segura
  - En un computador pueden existir varios procesos ejecutándose simultáneamente
  - La cantidad de memoria total necesaria para todos los procesos es mucho mayor de la que se dispone
  - Se observa:
    - Sólo una pequeña fracción de la memoria se usa activamente en cualquier instante de tiempo
      - La MP sólo necesita contener las porciones activas de cada proceso
    - El principio de localidad permite llevar a MP sólo los bloques que se van a utilizar
  - Como resultado de estas observaciones se implementa la memoria virtual

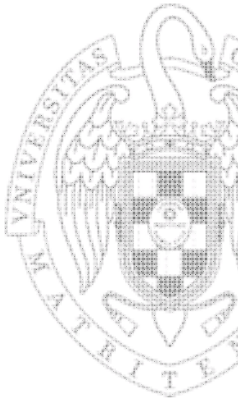
Motivaciones históricas (iii)



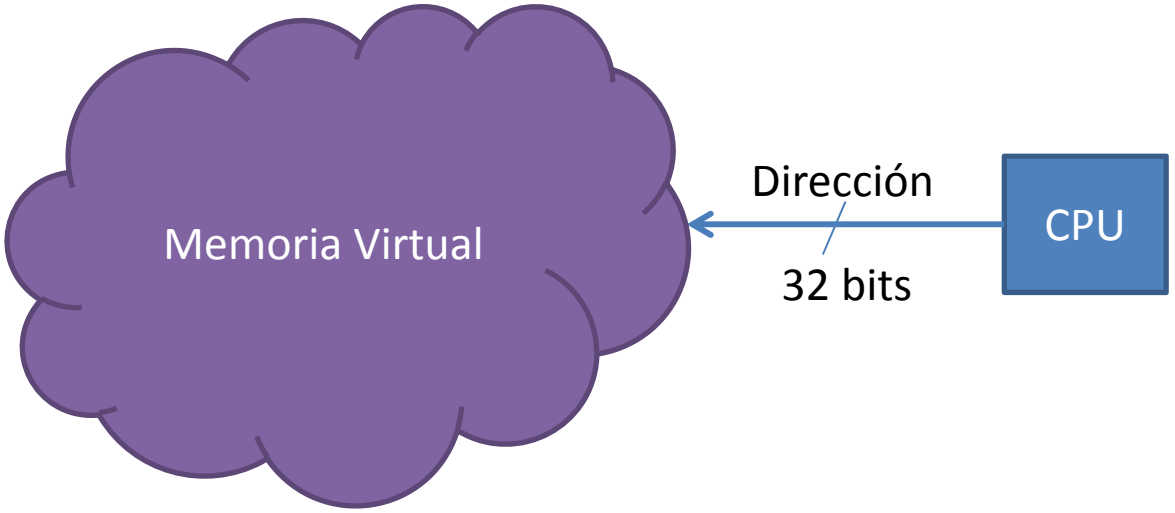
- Solución
  - Un programa sólo puede leer y escribir en la zona de MP que se le asigna
  - Compilar cada programa con su propio espacio de direcciones lo que implica que la misma dirección virtual (la proporcionada por el procesador) de dos procesos diferentes se cargue en direcciones físicas diferentes
- Implementación de la solución:
  - La memoria virtual traduce el espacio de direcciones virtual de un programa al espacio de direcciones físicas
  - Esta traducción protege el espacio de direcciones de un programa de los accesos indebidos de otros programas



¿Qué es la memoria virtual?

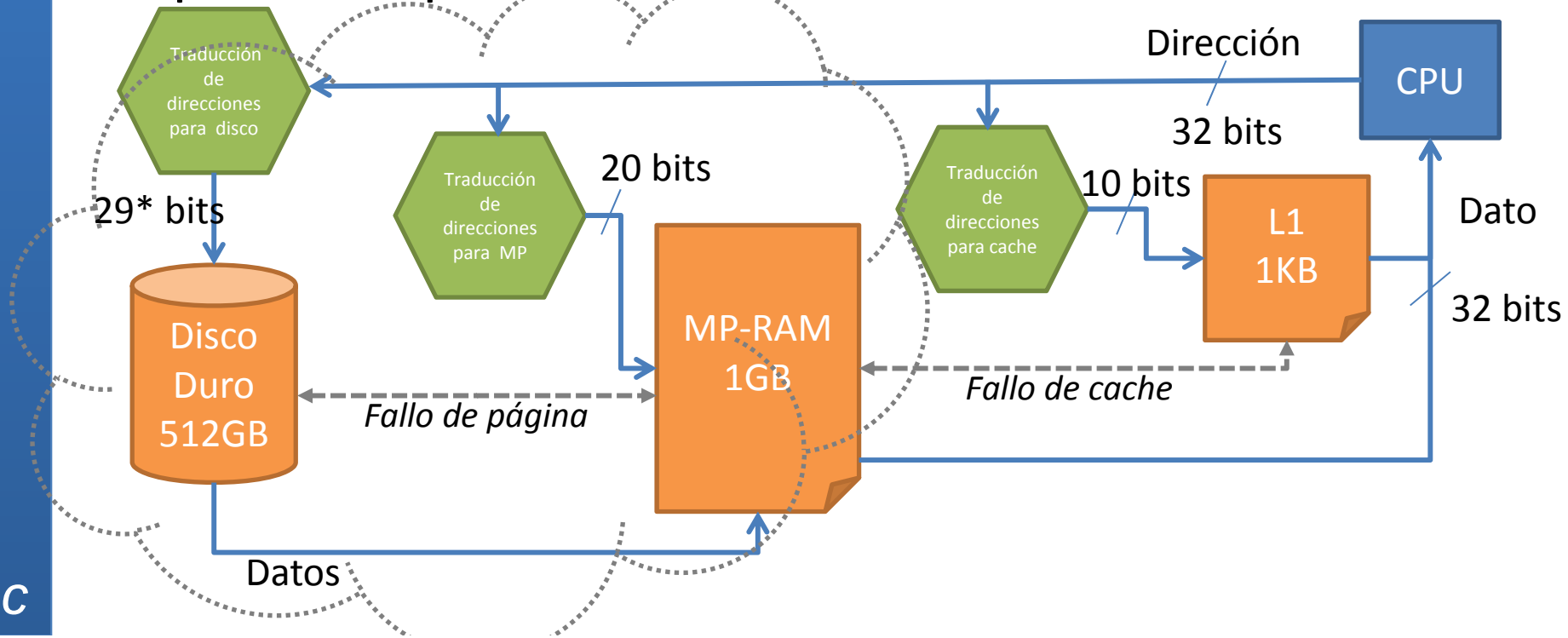


- Sistema de almacenamiento administrado por el sistema operativo (SO) que da al procesador la sensación de tener un espacio de direcciones superior al que realmente tiene

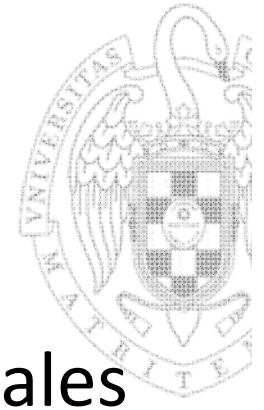


¿Qué es la Memoria Virtual?

- Sistema de almacenamiento administrado por el sistema operativo (SO) que da al procesador la sensación de tener un espacio de direcciones superior al que realmente tiene



¿Qué es la Memoria Virtual?



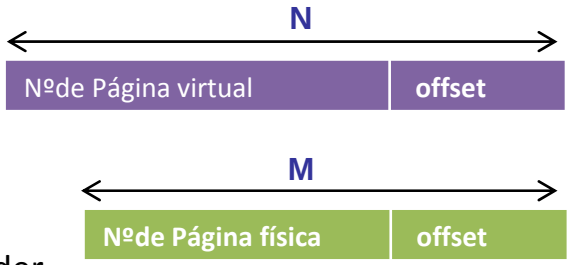
- La memoria virtual utiliza las direcciones virtuales generadas por el procesador que son de mayor tamaño que las direcciones físicas
  - Las direcciones virtuales se traducen a direcciones físicas (Memoria Principal)
  - Se necesita un mecanismo de traducción de direcciones
  - Si el dato buscado no está en MP se buscará en Memoria Secundaria
- El tamaño de la dirección generada por el procesador determina el tamaño de la memoria virtual

Definiciones

- Como funciona
  - Se divide la memoria principal en bloques del mismo tamaño llamados **páginas** físicas
  - Se divide un proceso en **páginas** virtuales del mismo tamaño que las fiscas (el proceso está almacenado en MS)
  - Se carga en la MP un pequeño conjunto de páginas del proceso
  - El resto de páginas se quedan en MS
  - Se van trayendo nuevas páginas a la MP según se van necesitando
- Necesidades
  - Mecanismo de traducción de direcciones
  - Estrategias de administración → Sistema operativo
- Fallo de página.-
  - Cuando se intenta acceder a una página que no está en memoria principal
  - Los gestiona el SO

Dirección virtual y dirección física

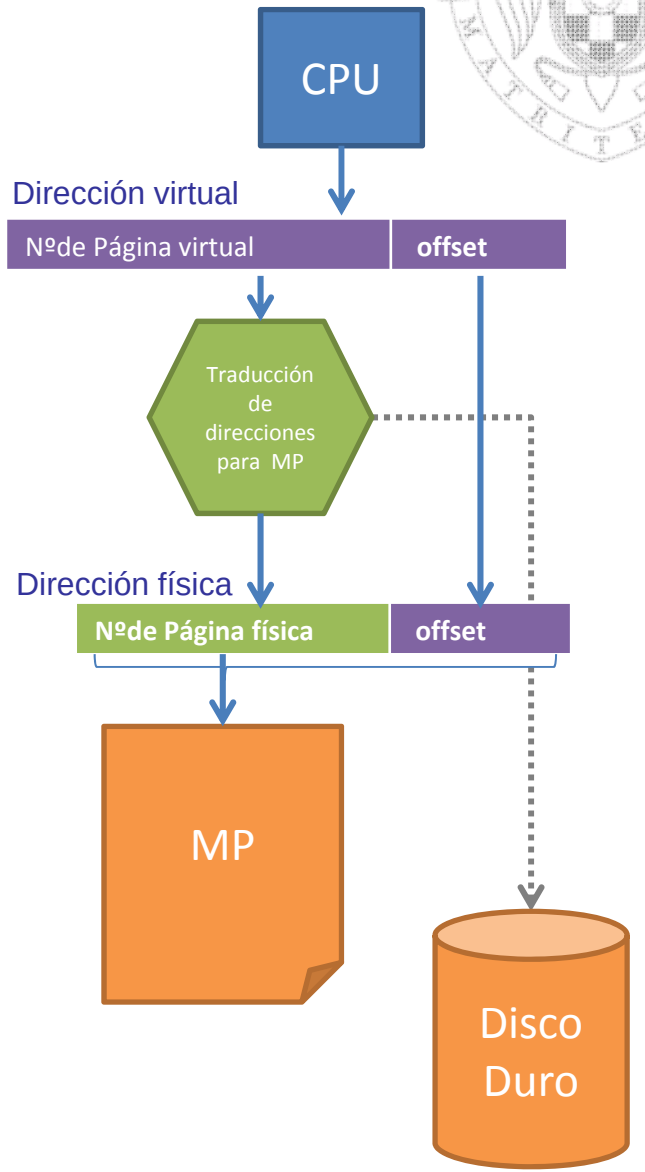
- La dirección virtual
  - Es la proporcionada por el procesador
  - Se divide en dos campos:
    - Número de página virtual
    - Desplazamiento (offset)
      - Indica que palabra dentro de la página se quiere acceder
      - El número de bits del offset determina el tamaño de página



- La dirección física
  - La dirección física es la utilizada por la Memoria Principal
  - Para su tratamiento en la gestión de la memoria virtual se divide en dos campos
    - Nº de pagina física
    - Offset (del mismo tamaño que en la dirección virtual)
- El tamaño de la DV >> tamaño de la DF (N>M)
  - El número de páginas virtuales no tiene que ser igual al número de paginas físicas

Mecanismo de traducción

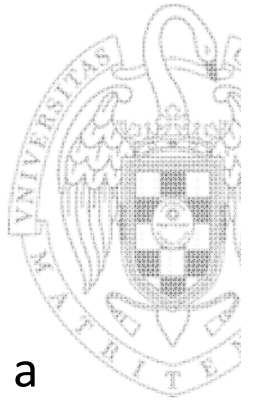
- Traduce la dirección virtual a la dirección física
  - se traduce en número de página virtual al número de pagina física
  - el desplazamiento (offset) es el mismo para la dirección virtual y la física
- La Traducción es dinámica lo que significa que se lleva a cabo en tiempo de ejecución
- Tiene que implementar un mecanismo para direccionar las páginas virtuales que se encuentra en MS



Mecanismo de traducción y emplazamiento

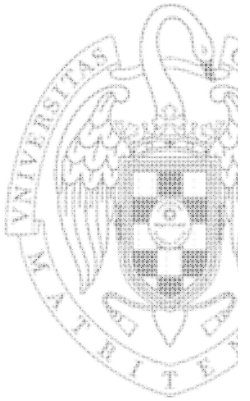
- El emplazamiento determina en que página física se puede ubicar una página virtual
  - Determina el mecanismo de traducción
- Debido a las elevada penalización de los fallos de página es conveniente reducir la tasa de fallos
  - Se consigue optimizando el emplazamiento
- Se usa el emplazamiento totalmente asociativo:
  - Si una pagina se puede emplazar en cualquier posición de memoria física el SO puede escoger cualquier página para reemplazar
  - SO utiliza algoritmos para escoger una página que no vaya a ser utilizada en mucho tiempo
- La dificultad del emplazamiento totalmente asociativo está en localizar la entrada puesto que ésta puede estar en cualquier posición
  - Tablas de páginas
    - Estructura de datos que asigna a cada página virtual una página física
    - Asegura que una página virtual se puede ubicar en cualquier página física

Tabla de páginas



- Es una estructura de datos que permite traducir direcciones virtuales a direcciones físicas
- La tabla tendrá tantas entradas como posibles páginas virtuales tenga nuestra arquitectura
  - Es decir, se indexa con el número de página virtual y se obtiene el número de página física
  - Va a contener entradas de páginas que no se encuentran en MP
- Reside en memoria principal
- La crea el SO cuando se crea un proceso por primera vez
- Otra información que puede contener una entrada
  - Bit de validez
  - Bit de referencia
  - Bit dirty
  - Bits de seguridad
- Debido a que la tabla de páginas contiene una entrada para cada página virtual no hacen falta tags

Tabla de páginas



- Para su implementación necesita un Registro de tabla de páginas
  - Indica la dirección de MP en la que comienza de la tabla
  - El número de página virtual actúa como un desplazamiento que se suma al contenido del registro de tabla para calcular la posición de la entrada
- Incluye un **bit de validez** por cada entrada para indicar si la página virtual está en MP
  - Bit a 1 → página en MP → la entrada contiene un número de página física
  - Bit a 0 → no está en MP → fallo de página

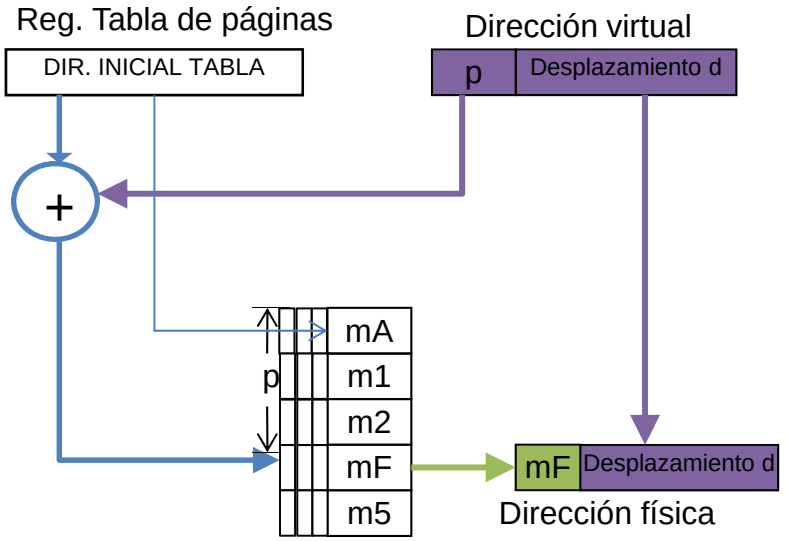
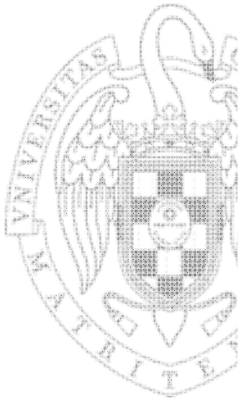


Tabla de página directa

Fallo de página



- ¿Qué es?
  - La página a la que se intenta acceder no está en Memoria Principal
- ¿Cómo se sabe que ha ocurrido?
  - bit de validez = 0
- ¿Qué ocurre?
  - Se produce una excepción y el SO toma el control
  - Acciones:
    - Buscar la página en disco duro
    - Decidir donde localizar la nueva página en Memoria Principal → reemplazamiento
- ¿ Como se busca la página en el Disco Duro (DD)?
  - La dirección virtual no nos indica donde esta la página en DD
  - El SO conoce la localización en DD de cada página virtual
  - Como no se conoce de antemano cuando una página en MP será elegida para ser reemplaza el SO reserva un espacio en disco para todas las paginas del proceso cuando lo crea. Espacio de swap.
  - Al mismo tiempo crea una estructura de datos para guardar en que posición de disco está almacenada cada página virtual
    - Esta estructura puede ser parte de la tabla de páginas o una estructura auxiliar que se indexa del mismo modo

Fallo de página-reemplazamiento LRU

- Remplaza el menos recientemente usado (LRU)
  - El SO crea una estructura de datos que guarda información sobre el uso de las páginas
- Las páginas reemplazadas se guardan en el espacio de swap
  - Ejemplo:
  - Primero se referencian las páginas 10,12,9,7,11,10,
  - Si después se referencia la 8, que no está en MP, se remplaza la 12
- Implementar un sistema exacto de LRU es muy caro puesto que hay que actualizar la estructura de datos cada referencia a memoria
- LRU aproximado:
  - Se incluye un Bit de uso o referencia en cada entrada de página
  - Cada vez que se accede a una página se activa
  - El SO lo limpia cada cierto tiempo:
    - Nos indica que paginas fueron accedidas en un periodo de tiempo
    - SO puede seleccionar una de las páginas menos recientemente usadas (bit uso =1)

Problemas de las tabla de páginas

- Sea sistema de memoria virtual con las siguientes características
  - Una dirección virtual de 32 bits
  - Un tamaño de página de 4 kb,
  - 4 bytes por entrada de página,
- ¿Cuál es el tamaño de la tabla de páginas?
  - Número de páginas= 2^{32} / 2^{12} =2^{20}
  - Tamaño de la tabla de páginas= Nº de entradasxbytes por entrada= 2^{20} x 2^{2} = 4MB
- Problema:
  - Sabiendo que un computador puede tener entre decenas y centenas de procesos activos y considerando el tamaño de página fijo casi toda la memoria se usaría para almacenar tablas de páginas
- Solución:
  - Tabla de páginas de dos niveles
  - Tabla de páginas inversa

Tabla de páginas de dos niveles

- La tabla de páginas se divide en subtablas que pueden estar en Memoria Principal o secundaria -> tabla virtual.
- No es necesario que las subtablas estén en posiciones consecutivas en memoria
- Es necesaria una dirección base para cada subtabla
  - Tabla de direcciones de subtabla
- La dirección de página virtual se divide en dos
  - Dvs dirección virtual de subtabla
  - P dirección de la página en la subtabla
- Pentium

Tabla de páginas de dos niveles

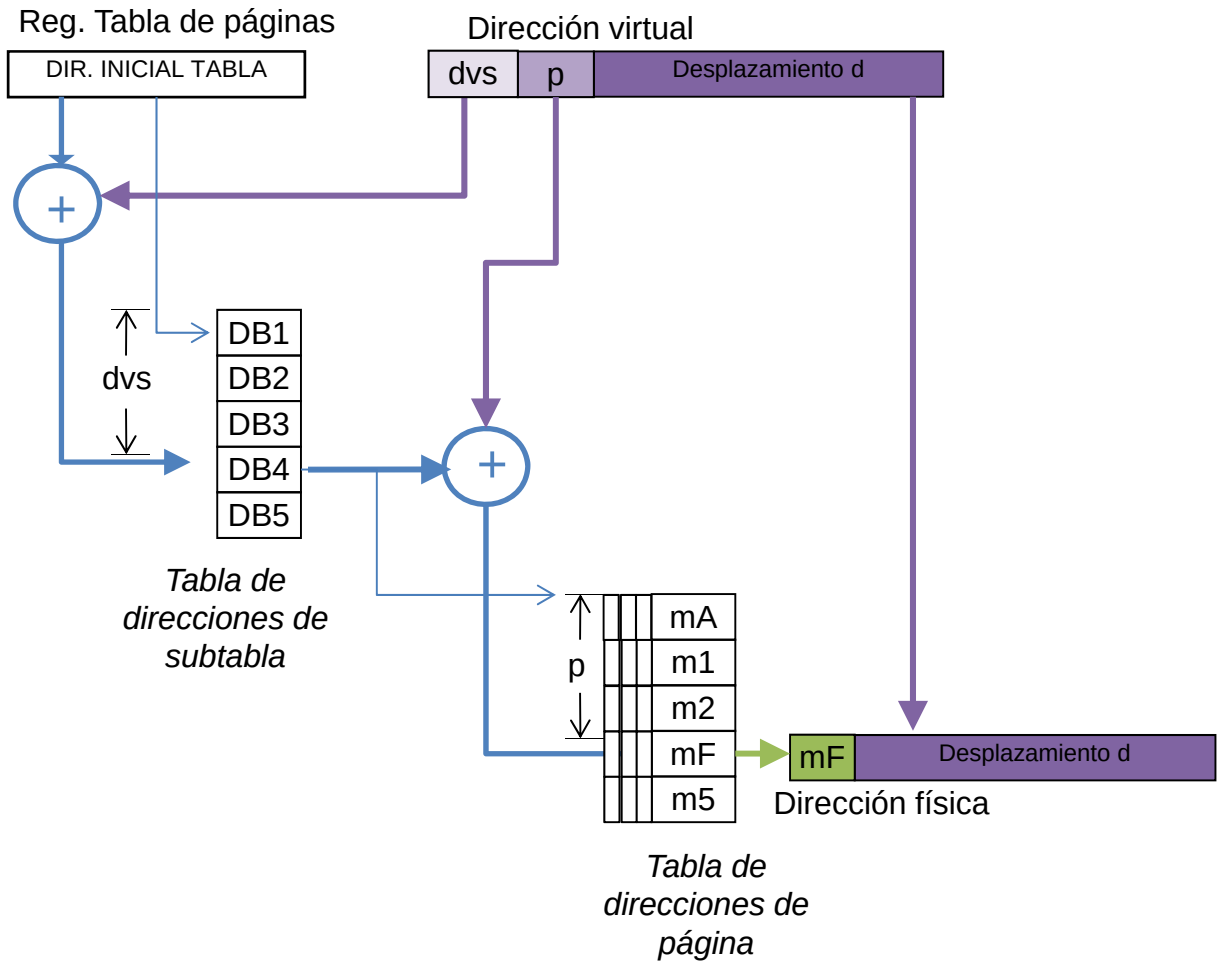
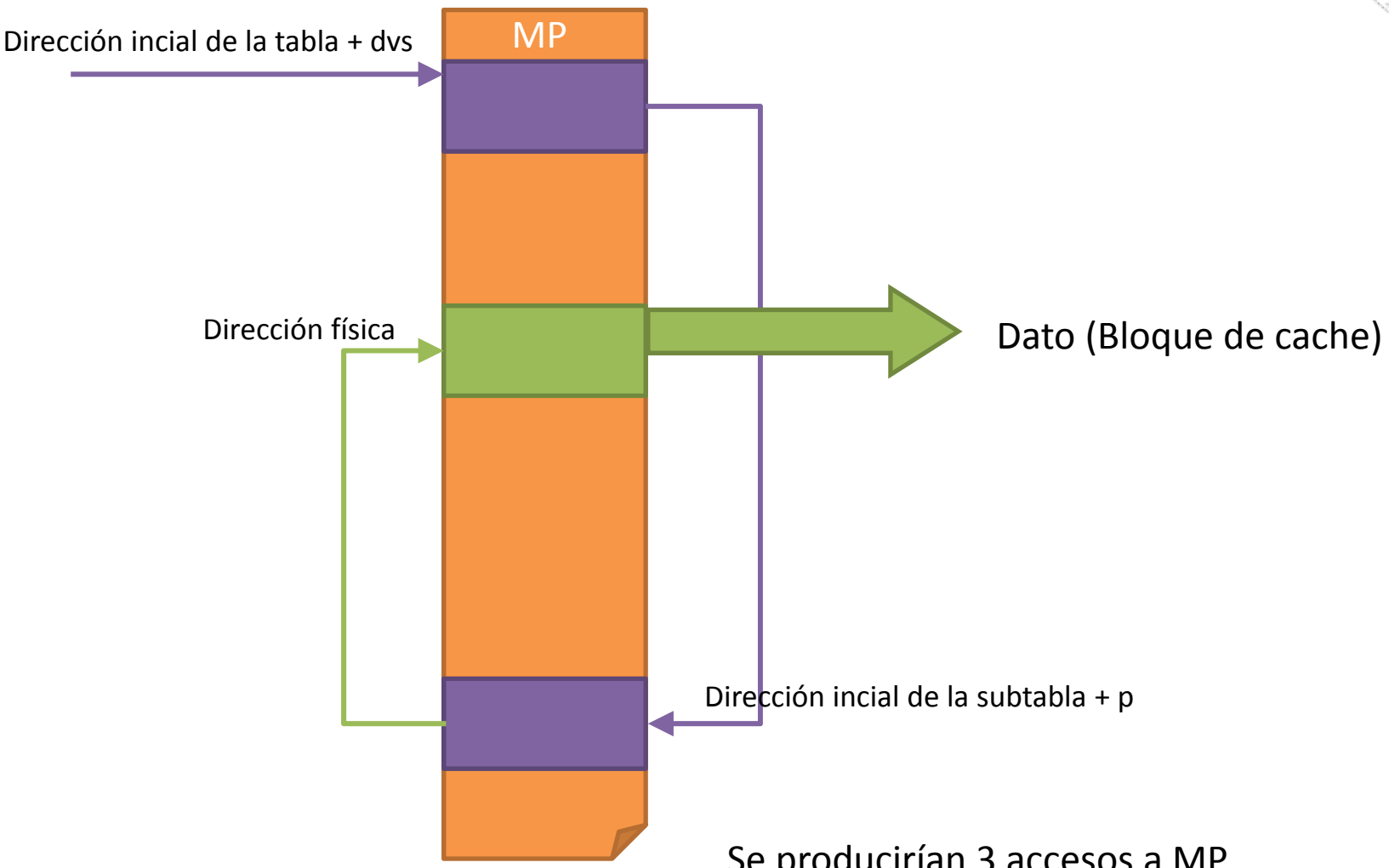


Tabla de páginas dos niveles



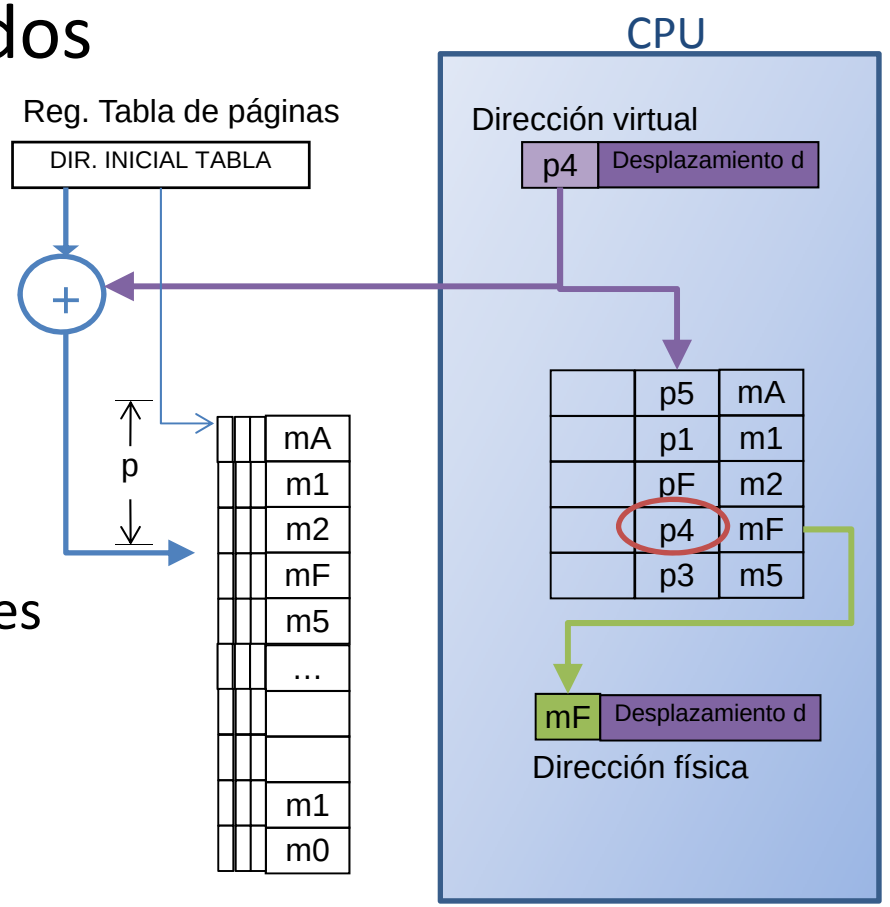
Se producirían 3 accesos a MP

Translation Lookaside Buffer (TLB)

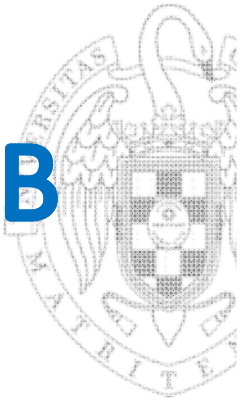
- Problema de las tablas de páginas: Tiempo de acceso a un dato
  - Puesto que la tablas están almacenadas en MP para obtener un dato se necesitan dos accesos a memoria:
    - un acceso para obtener la dirección física
    - un acceso para obtener el dato
    - En tablas multinivel puede ser un tiempo mucho más largo
- Solución :Translation Lookaside Buffer (TLB)
  - Cache de traducciones que guarda información sobre las traducciones recientes
  - Muy pequeña y muy rápida
  - Se basa en la localidad de referencias
    - Si una dirección se ha traducido, la probabilidad de que se vuelva a traducir en un futuro cercano es muy elevada
  - Está situada en el chip procesador
  - La etiqueta almacena una porción del número de página virtual
  - El dato almacena el número de página física
- Ya no se accede a la tabla de página en cada referencia sino a la TLB → necesita incluir otro bits de información necesarios para la gestión
  - Bit dirty- ¿se ha escrito?
  - Bit de referencia - ¿se ha accedido?
  - Bit de validez

Translation Lookaside Buffer TLB

- El sistema de memoria virtual está formado por dos tablas :
  - TLB
    - Contiene las últimas traducciones
    - En el chip procesador
  - Tabla de páginas
    - Contiene todas las direcciones de páginas
    - En Memoria Principal



Translation Lookaside Buffer TLB

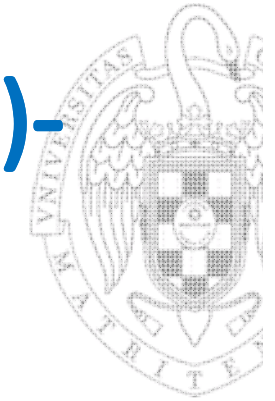


- La TLB posee los siguientes campos
  - Dependiendo de su valor se producirá acierto (hit) o fallo
  - Acierto: Se sabe dónde está la página buscada en MP sin necesidad de pasar por la tabla almacenada en MP
  - Fallo: Puede que la página no esté en MP, hay que comprobar la tabla almacenada en MP

V	R	W	D	Página Virtual	Página Física

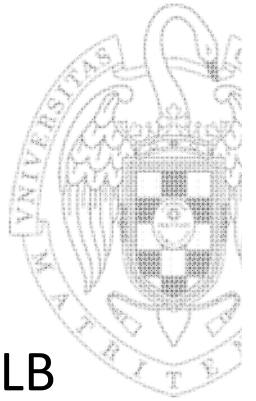
Si la página Virtual se encuentra aquí
Se conoce directamente en qué posición de MP está

Translation Lookaside Buffer (TLB)- alternativas de diseño



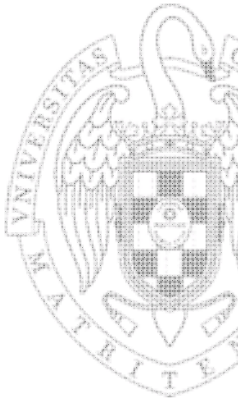
- Existen dos alternativas en el diseño de una TLB
  - Con identificadores de proceso
  - Sin identificadores de proceso
- TLB sin identificador de proceso
  - Se accede al TLB sólo con el número de página virtual
  - Cada vez que hay un cambio de proceso el sistema operativo debe invalidar la TLB ya que cada proceso tiene su propio mapa
- TLB con identificadores de proceso.
  - Se accede al TLB con el número de página y un identificador de proceso.
  - En cada entrada del TLB se almacena también este identificador
  - SO debe se encarga de asignar un identificador a cada proceso
  - En los cambios de proceso no es necesario que el sistema operativo invalide todo el TLB en cada cambio de proceso
- Bit de validez del TLB se utiliza para invalidar la información

Gestión de la TLB



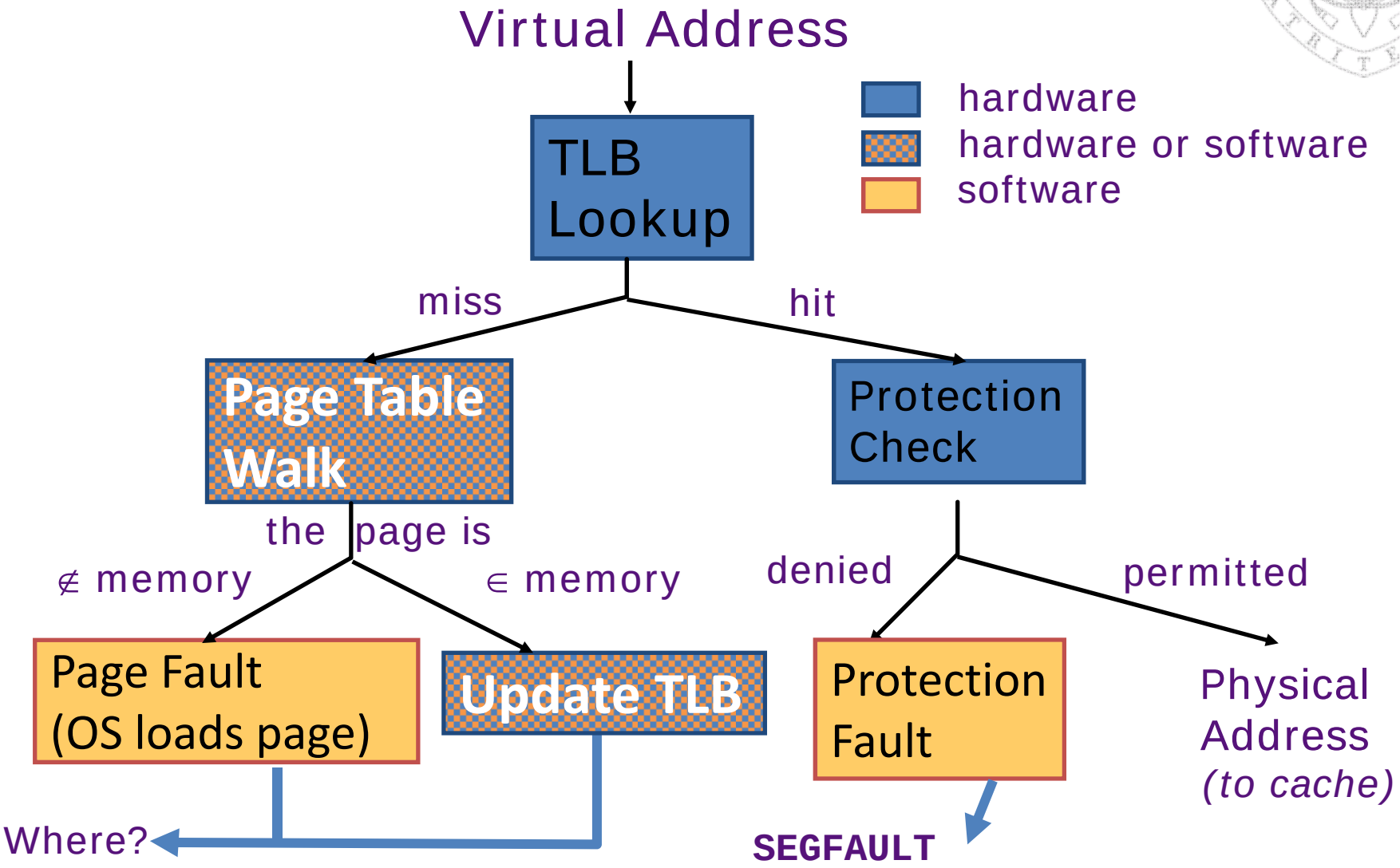
- En cada referencia se busca el número de página virtual en el TLB
  - Acierto → la entrada que se busca está en el TLB y el bit de validez (V) activo
    - Se usa el número de página física para calcular la dirección física
    - El bit de referencia se activa para cálculos LRU (R)
    - Si el proceso es de escritura el bit dirty se activa (D)
  - Fallo en TLB→la entrada que se busca no está en el TLB o el bit de validez inactivo
    - Se debe determinar si es un fallo de TLB o es un fallo de página
    - Si la pagina está en memoria MP es un fallo de TLB
      - Acción: se copia la información de la entrada en el TLB y se intenta la referencia de nuevo
    - Si la página no está en MP → es un fallo de página verdadero
      - Acción: Se invoca al SO mediante una excepción
    - Como el TLB tiene muchas menos entradas que el número de paginas en MP, un fallo de TLB será mucho mas frecuente que un fallo de página

Gestión de la TLB



- Los fallos de TLB pueden tratarse
  - SW
  - HW
  - Ambos tienen un rendimiento parecido
- Cuando se produce un fallo de TLB se debe seleccionar una entrada de TLB para reemplazar
  - Puesto que los bits de dirty y de referencia están grabados en el TLB, se deben salvar en la entrada de la tabla de páginas antes de reemplazarlo
- Valores típicos
  - Tamaño de TLB 16-512 entradas
  - Tamaño de bloque 1-2 entradas de tabla de página (4-8 bytes cada una)
  - Tiempo de acierto 0,5 -1 ciclo de reloj
  - Penalización de fallo 10-100 ciclos de reloj
  - Tasa de fallos 0.01-1%

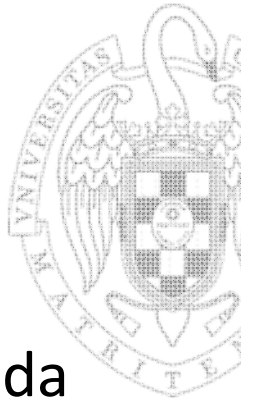
Gestión de la TLB



Tamaño de página

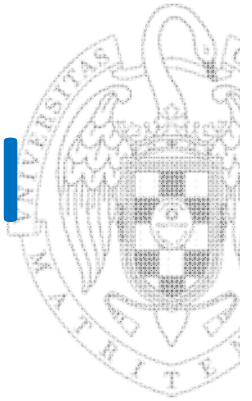
- El tamaño de las tablas de páginas es inversamente proporcional al tamaño de página
- Tamaños de página grandes:
  - Tamaños de cache mayores con tiempos de acierto rápidos
  - Transferir páginas desde almacenamiento secundario es más eficiente
    - La mayor parte del tiempo se consume en obtener la primera palabra
  - El tamaño de la TLB es muy pequeño,
    - A páginas más grandes se puede mapear más memoria de manera más eficiente y por lo tanto reducir los fallos de TLB
- Tamaños de página pequeños
  - Optimizan el uso de la memoria puesto que el tamaño de un proceso no siempre es igual a un múltiplo del tamaño de páginas (fragmentación interna)
  - La memoria malgastada es 1.5 veces el tamaño de página.
    - Cantidad despreciable para computadores con cientos de megabytes de memoria y tamaños de página de 4kb a 8kb,
    - Significativa para tamaños de página de más 32 kb,
      - Pueden aparecer problemas de almacenamiento principal y secundario además de problemas de ancho de banda de e/s
  - Para procesos pequeños un tamaño grande de página puede tener como resultado un tiempo de invocación del proceso demasiado largo
- Tamaño de páginas entre 4096 y 8192 bytes

Escritura



- Los ordenes de magnitud de los tiempos de acceso a cada nivel son:
  - Cache decenas de ciclos
  - Memoria principal cientos de ciclos de reloj
  - Memoria secundaria millones de ciclos de reloj
- En la memoria virtual la escritura directa es impracticable
- Se usa post escritura
  - Las escrituras individuales se realizan en la página de MP
  - Cuando es reemplazada la página se copia en disco duro
  - Se puede mejorar indicando si las páginas necesitan copiarse en disco duro
    - Bit dirty indica si la página se ha escrito
      - Bit dirty=0 no hace falta copiarla en disco duro
      - Bit dirty=1 hay que copiarla en disco duro

Protección vía memoria virtual

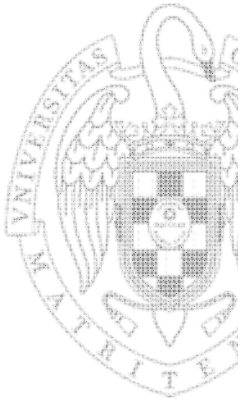


- Multiprogramación: varios programas se ejecutan concurrentemente compartiendo el procesador
  - Necesita mecanismos de protección y compartición
  - Concepto de proceso: un programa en ejecución más el estado que necesita para seguir ejecutándose
- Memoria virtual es el principal mecanismo que protege los procesos unos de otros
  - Existen diferentes niveles de protección
  - Las restricciones de protección se incluyen en cada entrada de tabla de páginas
    - Permiso de lectura de la página
    - Permiso de escritura de la página
    - Si un código se puede ejecutar desde esa página
  - Dado que el SO es el que genera la tabla es el SO el que fija los niveles de seguridad
- ¿Por qué existen los fallos de seguridad?
  - Problemas con la exactitud tanto del SO como del hw
  - SO tienen decenas de millones de líneas de código
    - Tienen miles de errores
    - Estos errores dan lugar a la vulnerabilidad del sistema

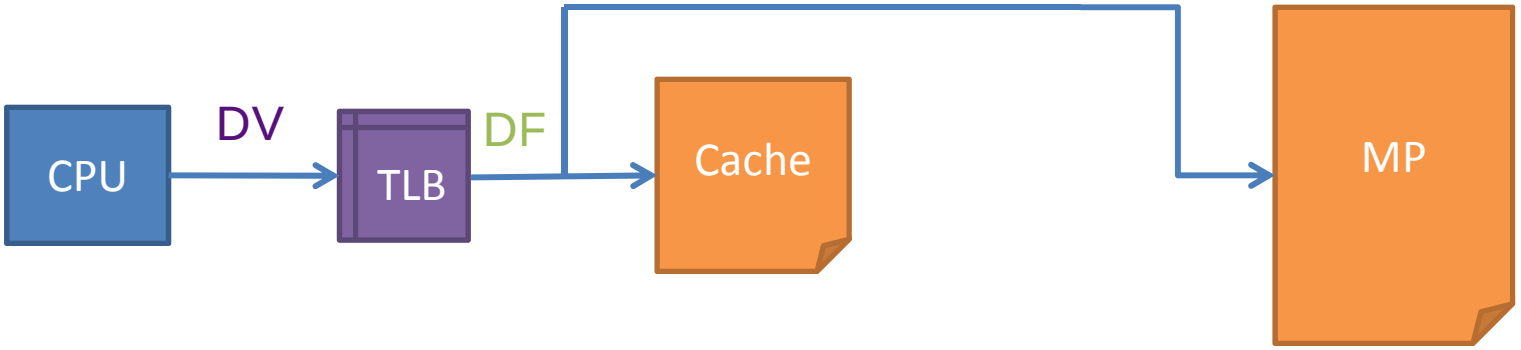
Integración de la Memoria virtual y cache

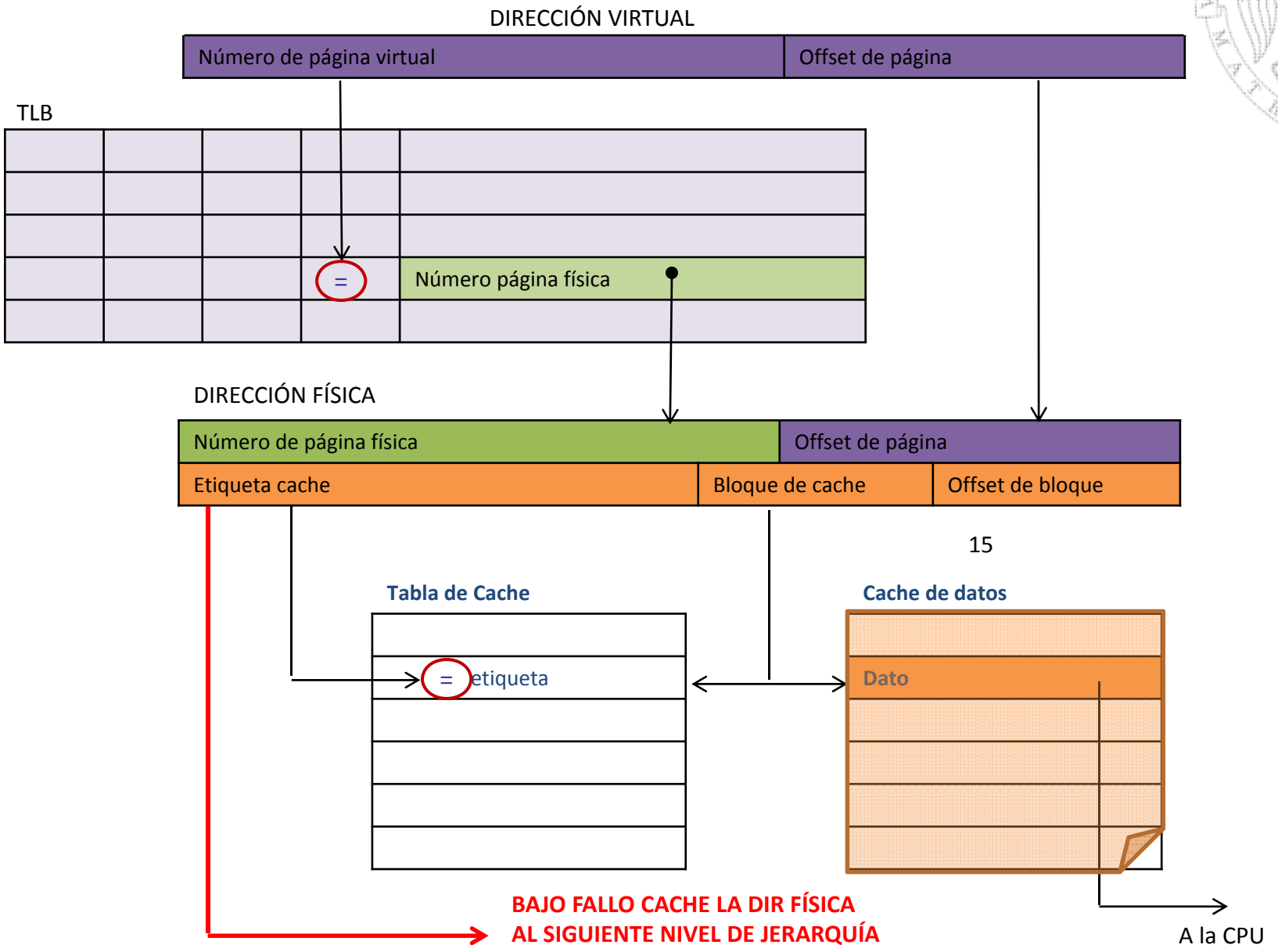
- La memoria virtual y la memoria cache trabajan juntas en la jerarquía
  - Un dato no puede estar en cache a menos que esté en MP
- El SO juega un importante papel al eliminar páginas de MP
  - Limpiando la MC
  - Modificando la tabla de páginas y TLB de manera que al intentar acceder a la página se produzca un fallo de página
- ¿Cómo se relaciona la dirección virtual con la memoria cache?
  - ¿Se utiliza una dirección física o una dirección virtual para comparar las etiquetas?
  - ¿Se utiliza una dirección física o una dirección virtual para indexar la cache?
  - Existen tres aproximaciones
    - Caches físicas
    - Caches virtuales
    - Caches virtualmente accedida, físicamente marcada

Cache de direcciones físicas

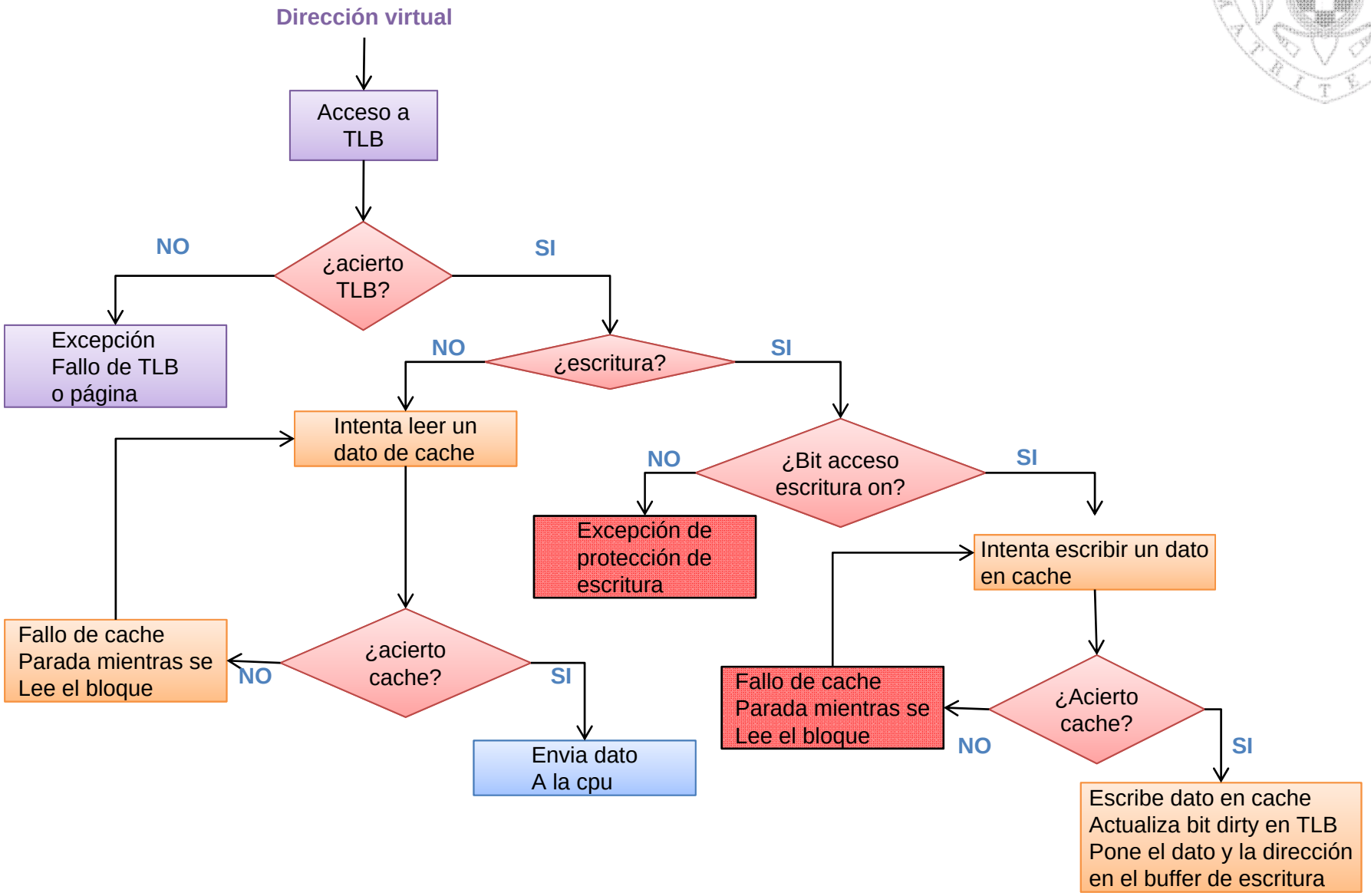


- La cache utiliza direcciones físicas
  - Utiliza la dirección física tanto para indexar como para comparar etiquetas
- La dirección virtual generada por el procesador se traduce siempre a través del TLB
- Permite que en la memoria cache convivan varios contextos es decir varios procesos activos.
  - La misma dirección virtual de diferentes procesos direccionada a diferentes direcciones físicas
- Retardos en los accesos a cache puesto que hay que traducir la dirección



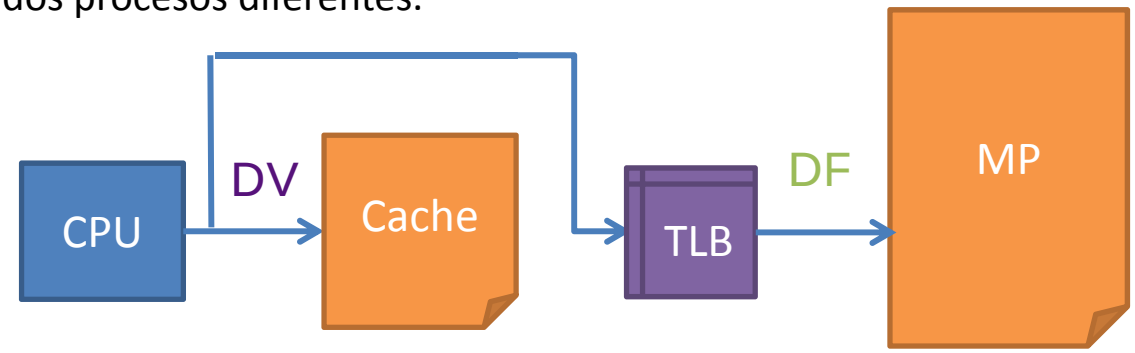

DIRECCIÓN VIRTUAL
Número de página virtual
Offset de página
TLB
=
Número página física
DIRECCIÓN FÍSICA
Número de página física
Offset de página
Etiqueta cache
Bloque de cache
Offset de bloque
15
Tabla de Cache
Cache de datos
= etiqueta
Dato
BAJO FALLO CACHE LA DIR FÍSICA
AL SIGUIENTE NIVEL DE JERARQUÍA
A la CPU

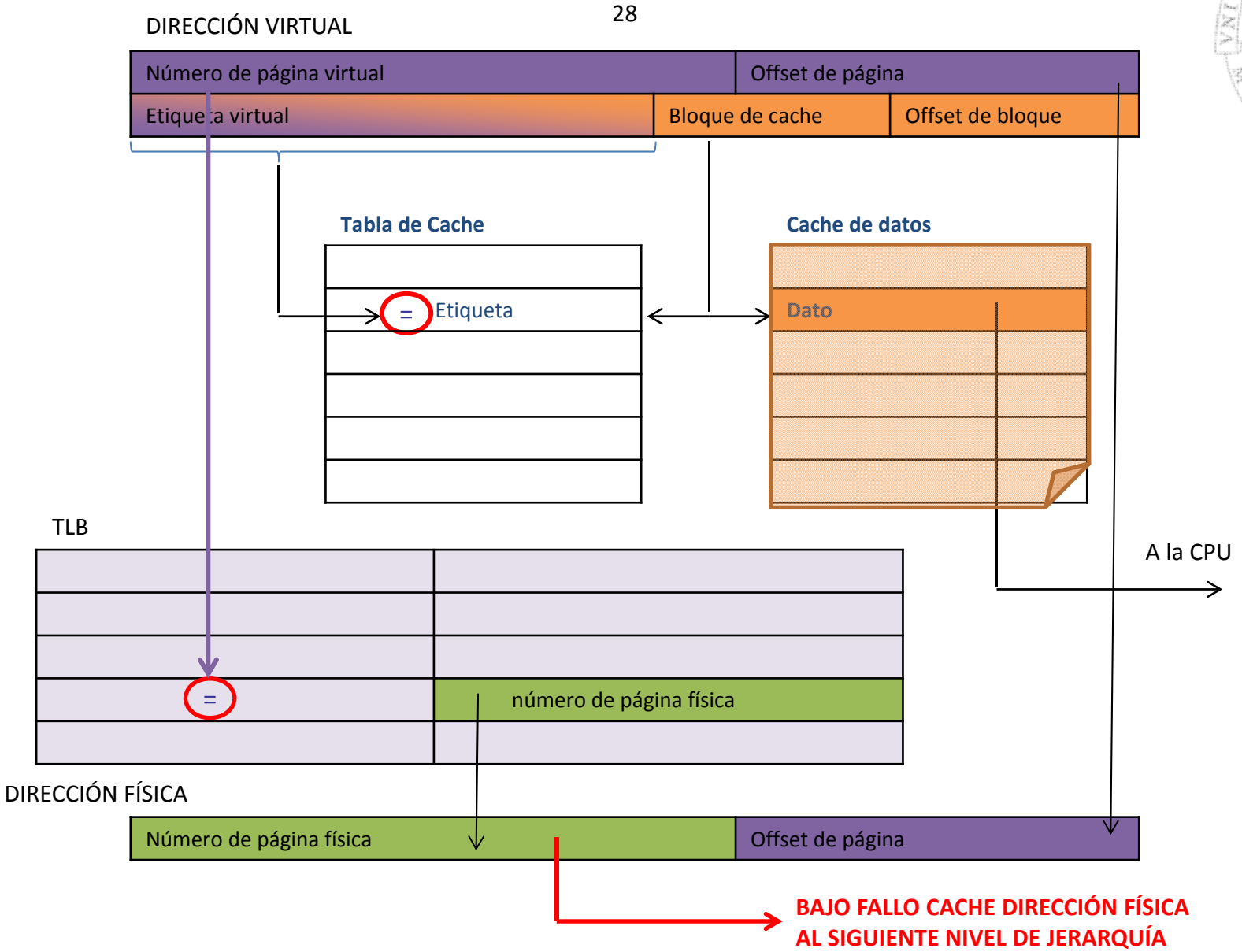
Cache de direcciones físicas



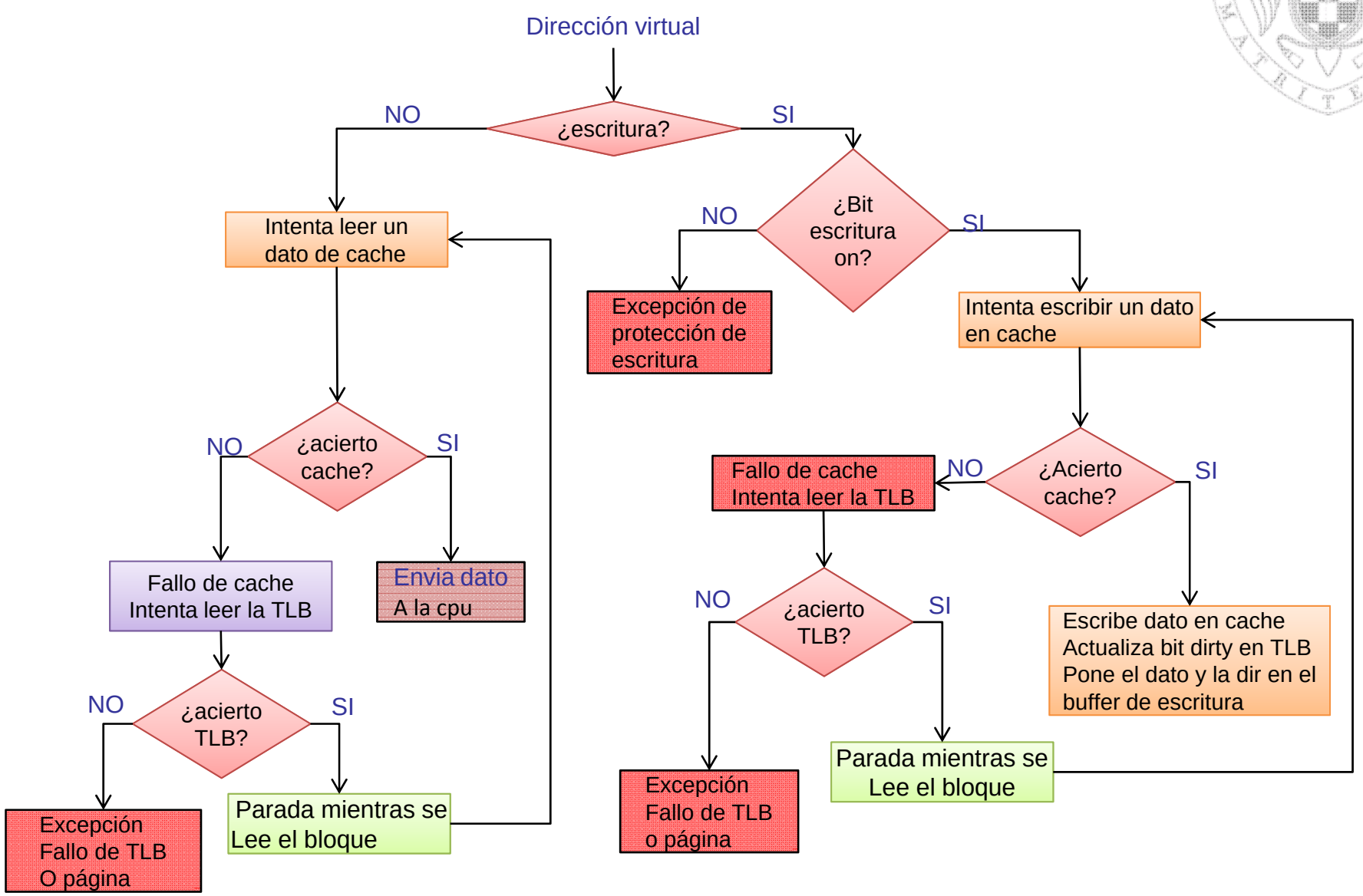
Cache de direcciones virtuales

- Se accede a la cache con la dirección virtual
  - Utiliza la dirección virtual tanto para indexar como para comparar etiquetas
  - La dirección virtual se reinterpreta NO se traduce
- Ventaja: acceso a la cache más rápido puesto que no se pasa por el TLB
  - Sólo se pasa por el TLB cuando hay un fallo de cache y se tiene que generar la dirección física para buscar el bloque en MP
  - Más tiempo en la gestión de un fallo de cache
- Problema del cambio de contexto:
  - Al cambio de contexto hay que borrar la cache.
    - De lo contrario: falsos aciertos
  - Tiempo de cambio contexto= tiempo de borrado (flush) + fallos iniciales
- Solución:
  - Añadir un identificador de proceso (PID) a la DV.
  - Permite distinguir DVs de dos procesos diferentes.




DIRECCIÓN VIRTUAL
Número de página virtual
Offset de página
Etiqueta virtual
Bloque de cache
Offset de bloque
Tabla de Cache
Cache de datos
=
Etiqueta
Dato
TLB
A la CPU
=
número de página física
DIRECCIÓN FÍSICA
Número de página física
Offset de página
BAJO FALLO CACHE DIRECCIÓN FÍSICA
AL SIGUIENTE NIVEL DE JERARQUÍA

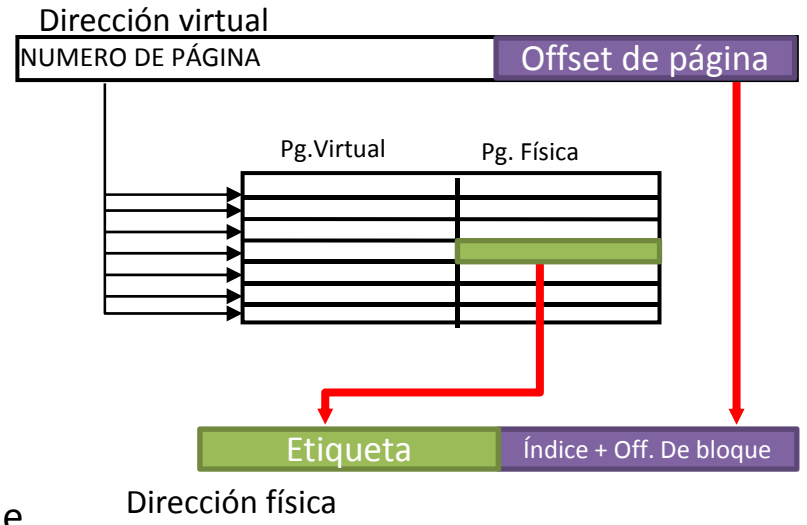
Cache de direcciones virtuales

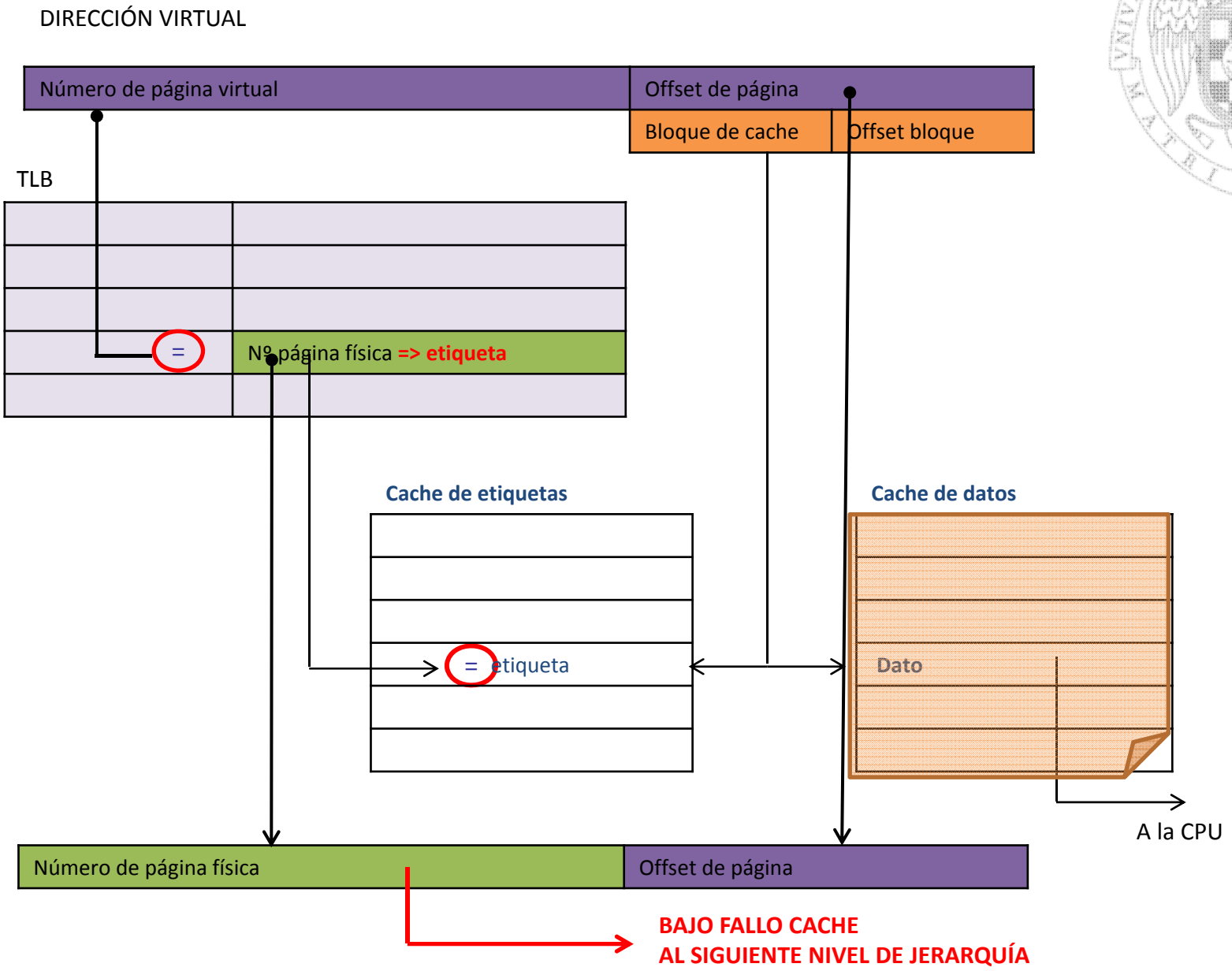


ec

Cache físicamente marcada virtualmente accedida

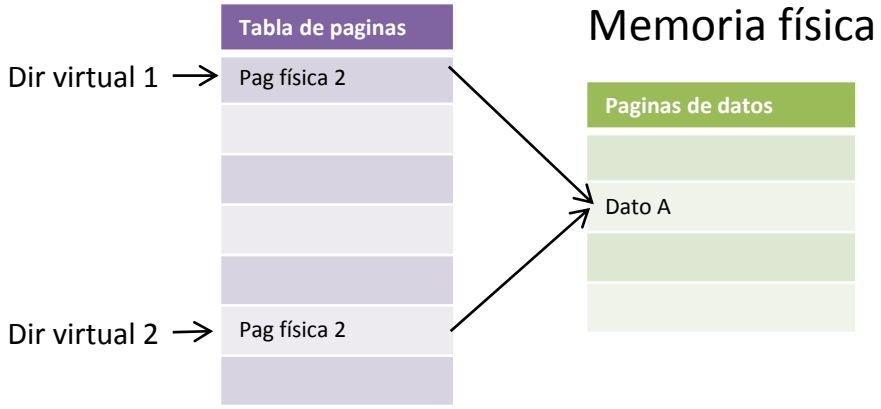
- Utiliza el offset de la dirección virtual que es igual para la dirección física para indexar la cache
- Se realiza en paralelo las siguientes acciones:
  - Lectura del dato usando el offset de la dir virtual
  - Se comparan las etiquetas utilizando la página física obtenida mediante el TLB
- Ventaja:
  - Ocultar la latencia de traducción DV => DF
- Desventaja
  - Con cache directa, limita el tamaño de la cache al tamaño de página
- Un modo de aumentar el tamaño máximo de la cache es aumentar la asociatividad




DIRECCIÓN VIRTUAL
Número de página virtual
Offset de página
Bloque de cache
Offset bloque
TLB
=
Nº página física => etiqueta
Cache de etiquetas
= etiqueta
Cache de datos
Dato
A la CPU
Número de página física
Offset de página
BAJO FALLO CACHE
AL SIGUIENTE NIVEL DE JERARQUÍA

Problema de los sinónimos

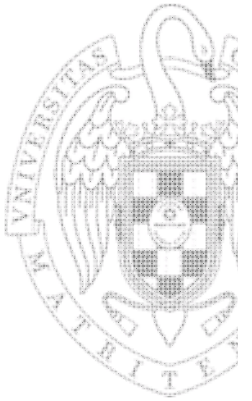
- Se puede dar en
  - caches virtuales
  - virtualmente accedidas físicamente marcadas
    - Tamaño de cache>tamaño de página
  - Dos direcciones virtuales apuntan la misma dirección física.
    - Misma dirección física diferentes posiciones en la cache virtual
  - Implica que pueden residir en la cache dos copias de la misma DF
    - Se modifica una de las copias pero no la otra
    - Esto no ocurre con la cache física puesto que las traducción se produce antes de acceso a cache
  - Solución: antialiasing
    - mecanismos hw para garantizar que cada bloque de la cache se corresponde con un bloque de MP diferente
    - Actúa bajo fallo, cuando se trae el nuevo bloque se comprueba que ninguna de las etiquetas físicas existentes coincide con la etiqueta del dato que se trae. En caso que así sea se elimina el dato que ya esta en cache



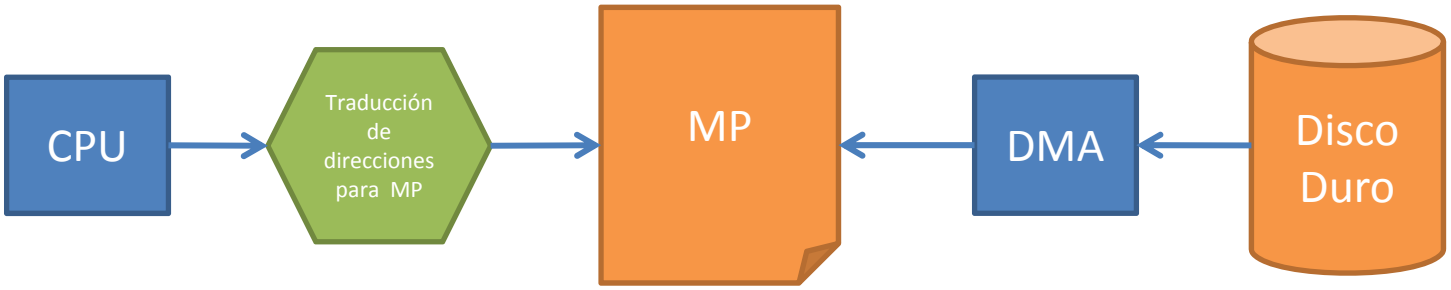
Cache virtual

tag	dato
Dir virtual 1	Primera copia del dato A
Dir virtual 1	Segunda copia del dato A

Memoria Virtual y DMA

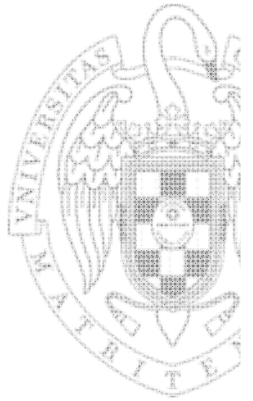


- Aparece una camino de acceso a MP alternativo
- Este camino no utiliza ni el mecanismo de traducción de direcciones ni la jerarquía de memoria



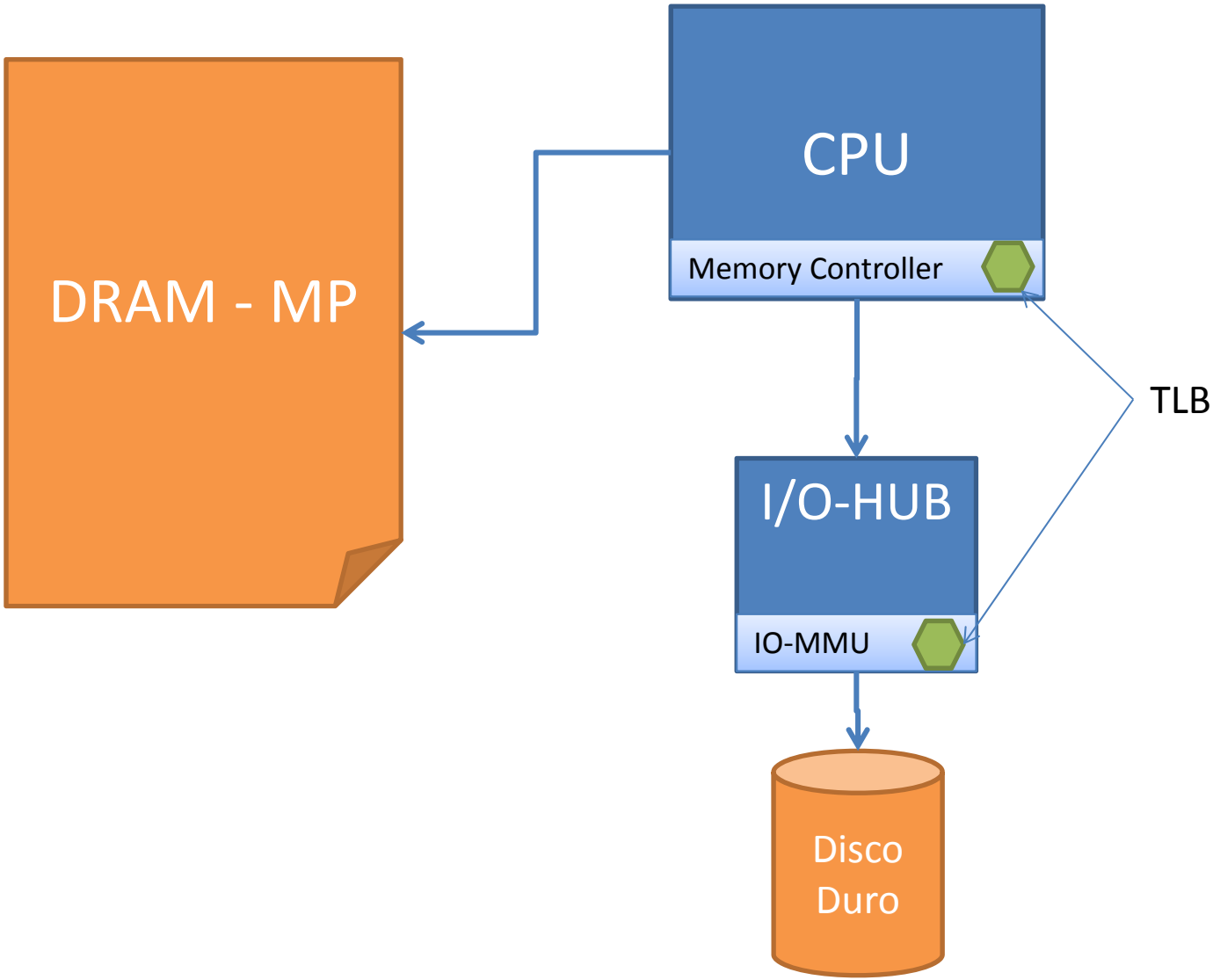
- ¿Cómo saber en qué posición de memoria se deben escribir los datos que proceden del disco duro?
  - La asignación de datos (páginas) a direcciones de MP es dinámica

Memoria Virtual y DMA

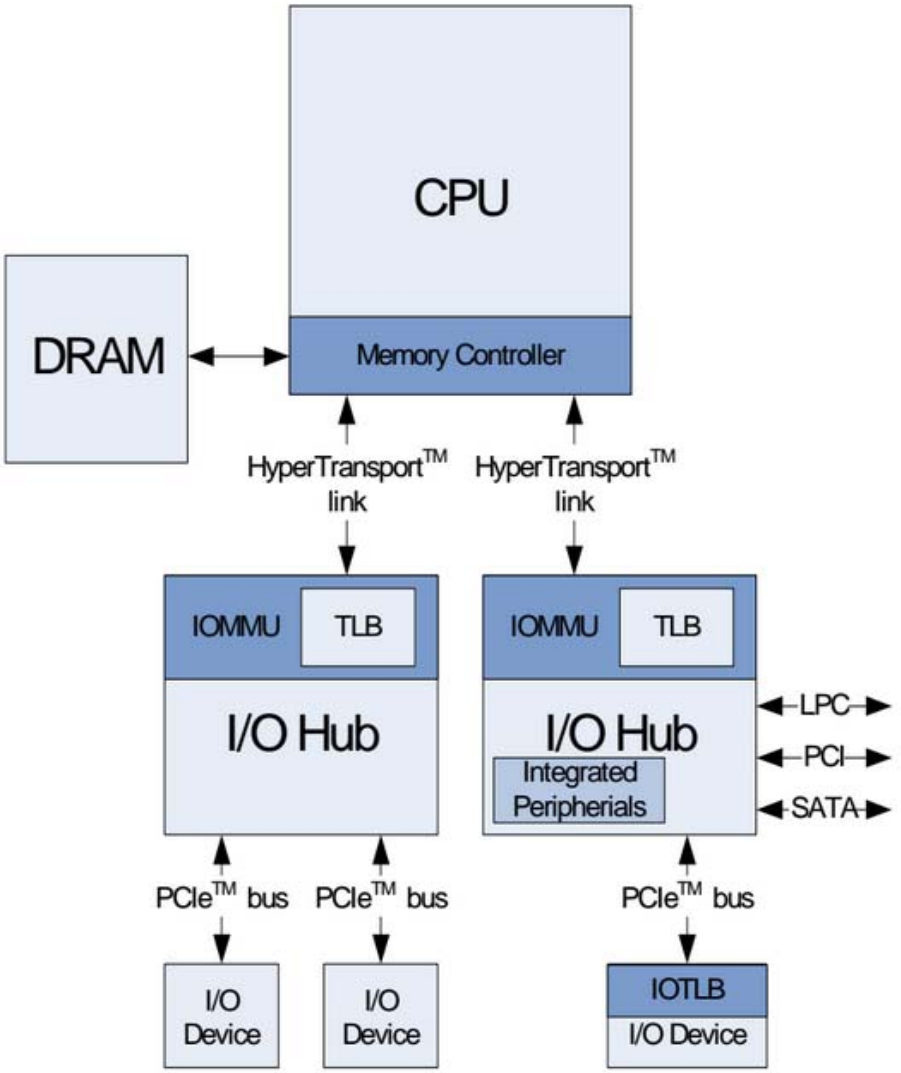


- DMA de direcciones virtuales
  - Utiliza las direcciones virtuales para acceder a la memoria
    - El DMA tiene un pequeño número de entradas de memoria que proporciona el mecanismo de traducción de direcciones para la transferencia
  - Durante el proceso la página(s) en MP donde se van a almacenar los datos no puede *desaparecer* de MP
    - Permiten la transferencia de tamaños superiores a una página
  - El sistema operativo es el encargado de suministrarlas cuando se inicia una operación de entrada/salida

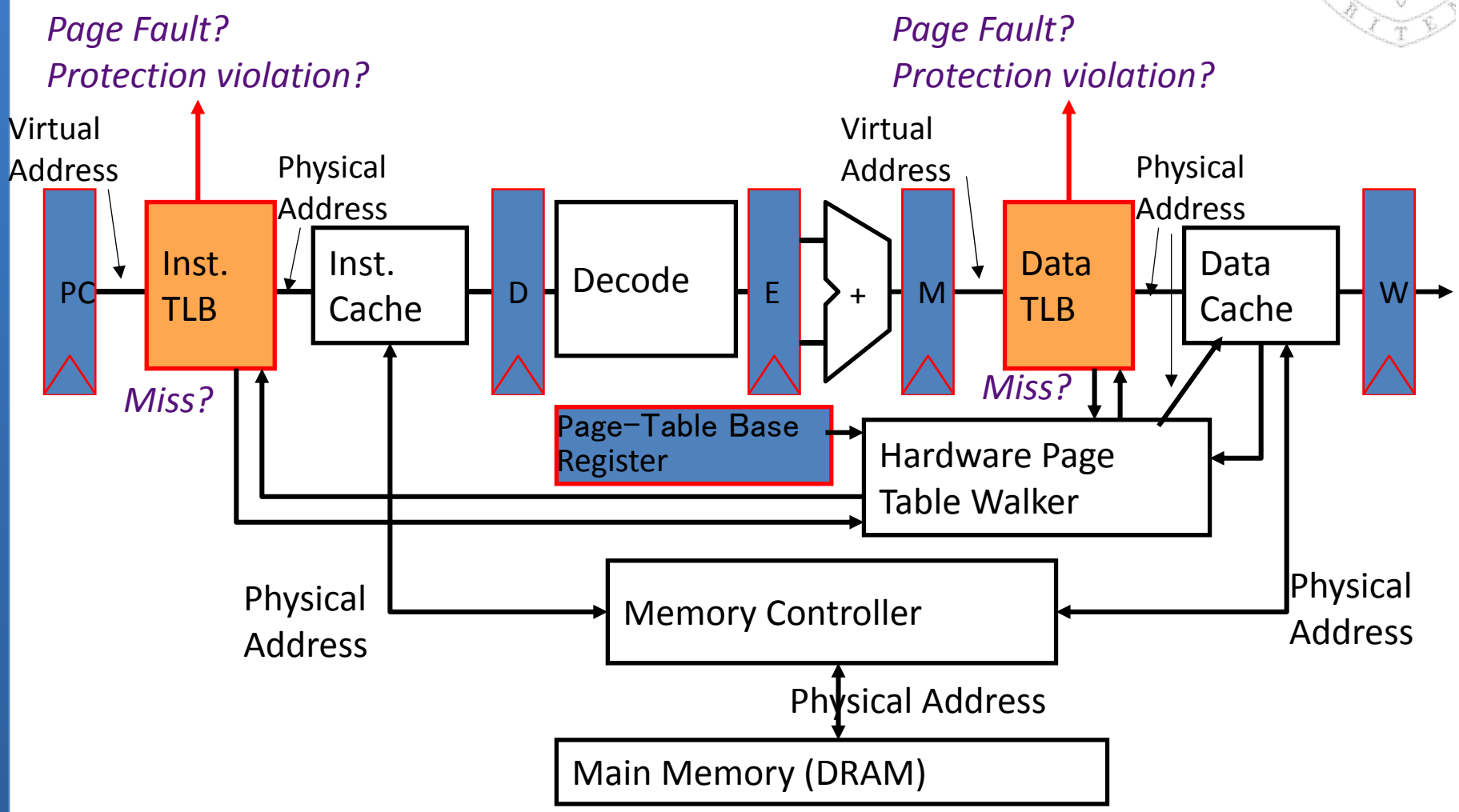
Memoria Virtual y DMA



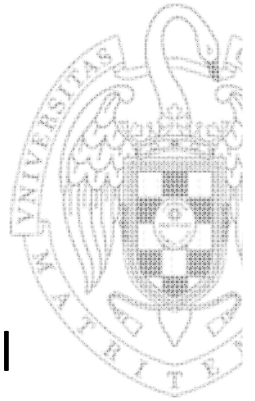
Memoria Virtual DMA (AMD)



Integración en el Procesador



Memoria Virtual en los sistemas actuales



- Servidores/PCs/portátiles/smartphones tienen memoria virtual
  - Portabilidad entre modelos con diferentes tamaños de memoria
  - Protección entre multiples usuarios o multiples tareas
  - Compartir memoria física (no muy grande) entre diferentes tareas (activas)
  - *Hacer la vida más fácil* al sistema operativo

- La mayoría de los procesadores empotrados y DSPs no tienen memoria virtual
  - No pueden permitirse el *lujo* de tener memoria virtual (area/rendimiento/potencia)
  - Muchas veces sólo disponen de una memoria (no tienen memoria secundaria)
  - Los programas estan *hechos a mano* para una configuración muy particular de memoria