ESTRUCTURA DE COMPUTADORES

Tema 3. Memoria Virtual

Dpto. Arquitectura de Computadores y Automática

Universidad Complutense de Madrid

ÍNDICE

- ¿Qué es la memoria virtual?
- Mecanismo de traducción
  - Tablas de páginas
  - Translation Lookaside Buffer (TLB)
- Escritura
- Protección vía memoria virtual
- Integración de la memoria virtual y la memoria cache
- Memoria virtual en sistemas reales

Bibliografía

- David A. Patterson & John L. Hennessy “Computer Organization and design. RISC-V Edition”, Morgan Kaufmann
- A. Bhattacharjee, D. Lustig “Architectural and Operating System Support for Virtual Memory”, Morgan & Claypool, 2018

¿ Qué es la Memoria Virtual?

- Sistema de almacenamiento de dos niveles administrado por el sistema operativo (SO) que da al usuario la sensación de tener una memoria principal (espacio de direcciones) superior al que realmente posee el sistema

- Se encarga de gestionar el intercambio de información entre la Memoria Principal (física) y la Memoria Secundaria (disco duro)
  - La Memoria Principal actúa como “cache” de Memoria Secundaria

OBJETIVOS DE LA MEMORIA VIRTUAL

- Objetivos:
  1. Permitir que los programas que se ejecutan tengan un tamaño mayor que el de la Memoria Principal
     - Casos:
       - Tamaño de un proceso mayor que el tamaño de memoria principal
       - Ejecución simultánea de varios procesos en los que la suma de sus tamaños es superior al tamaño de la memoria física

Objetivos de la Memoria Virtual

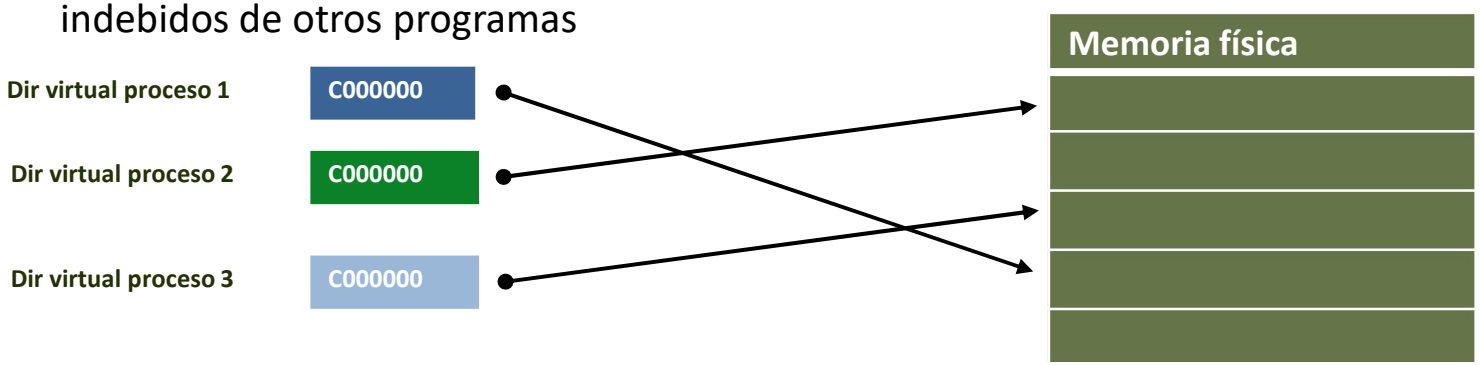
2. Compartir memoria de manera eficiente y segura
   - En un computador pueden existir varios procesos ejecutándose simultáneamente
   - La cantidad de memoria total necesaria para todos los procesos puede ser mucho mayor de la que se dispone
   - Se observa:
     - Sólo una pequeña fracción de la memoria 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
   - Problema:
     - La existencia de varios procesos en ejecución implica la necesidad de proteger los programas unos de otros
     - No se conoce en tiempo de compilación los programas que van a compartir memoria.
       - Los programas que comparten memoria varían dinámicamente durante la ejecución

Objetivos de la Memoria Virtual

3. Reubicación
   - Permite cargar un programa en cualquier posición de memoria principal
   - A la misma dirección virtual (generada por el procesador) se le pueden asignar diferentes direcciones físicas (memoria principal)
   - Hoy en día todos los sistemas de MV reubican
     - el programa se ve como un conjunto de páginas de tamaño fijo
     - elimina la necesidad de encontrar posiciones de memoria contiguas para cargarlo en memoria

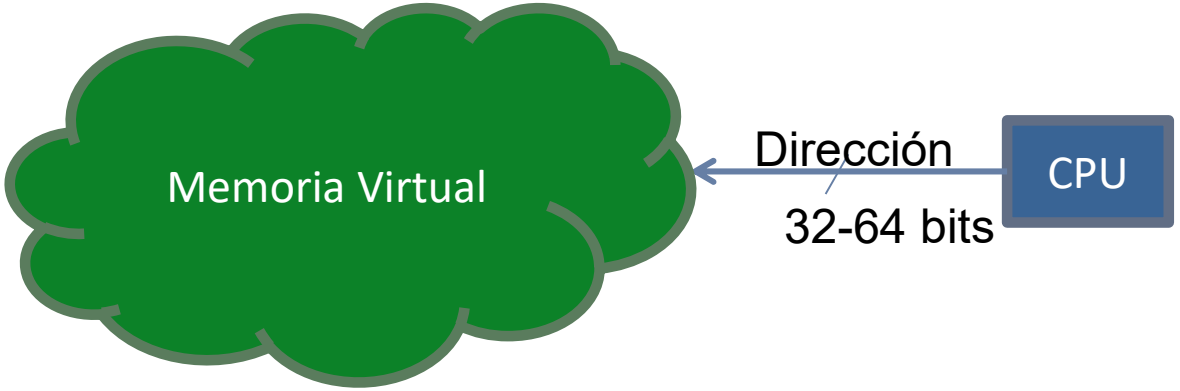
Memoria Virtual

- 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

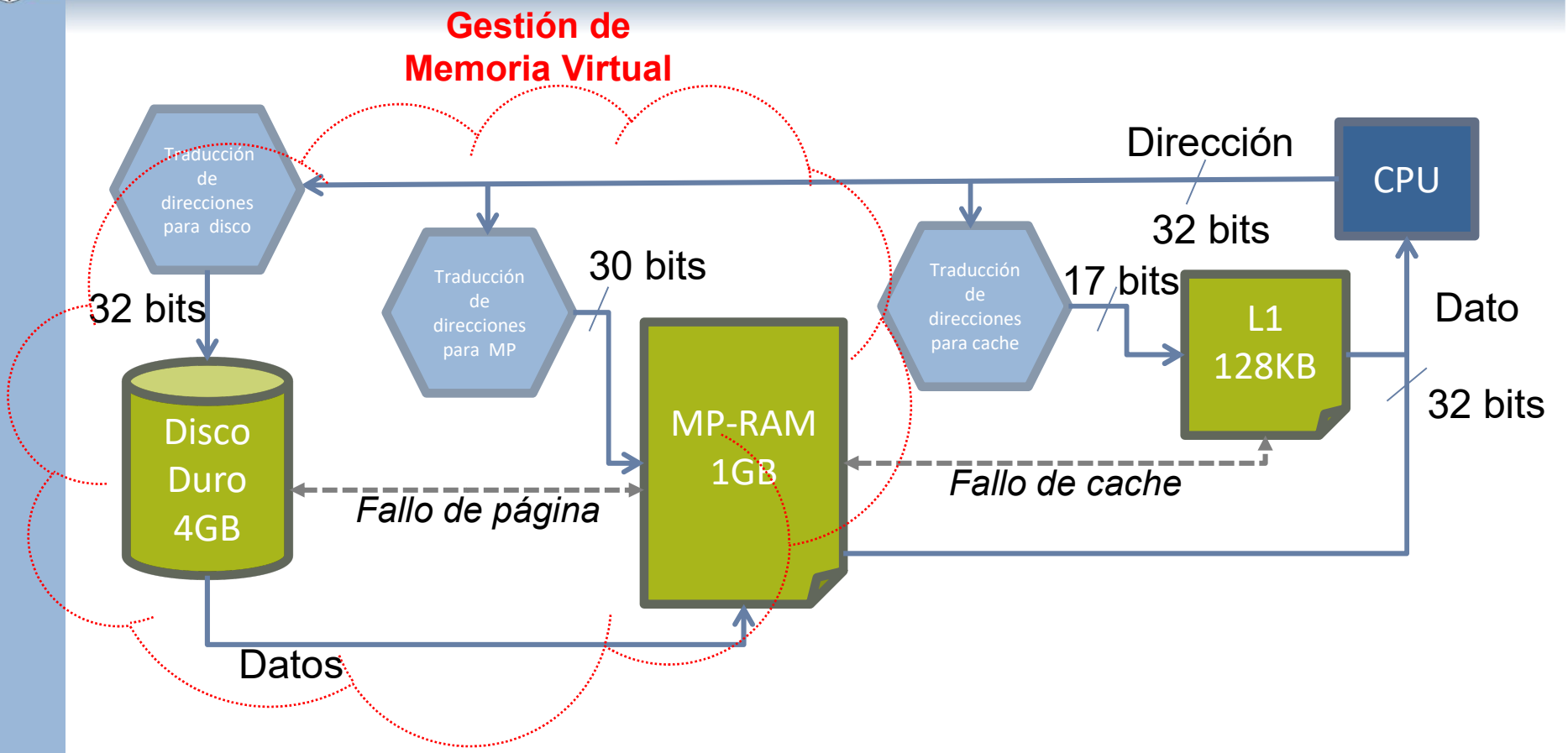


¿Qué es la memoria virtual?

- 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

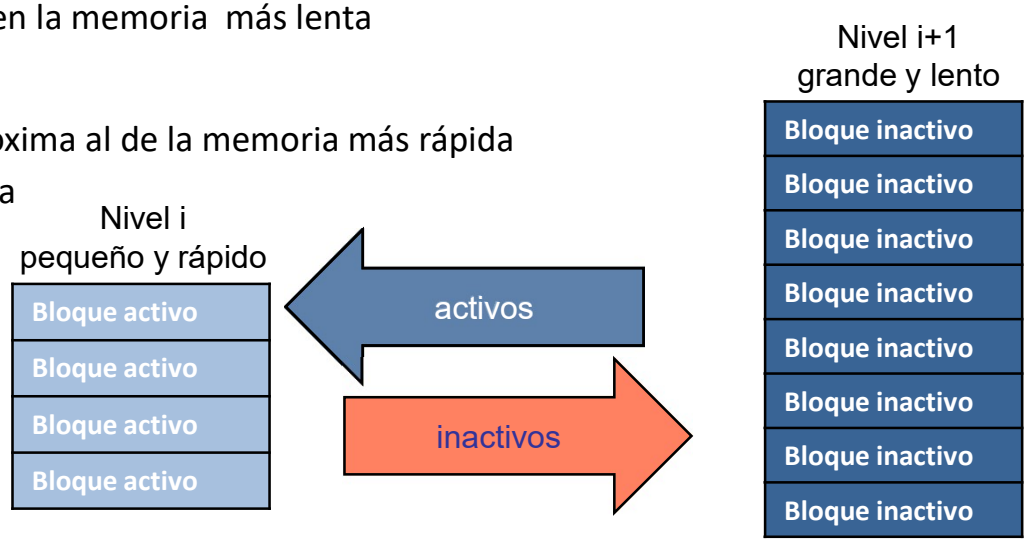


Gestión de la Memoria Virtual



MEMORIA CACHE VS MEMORIA VIRTUAL

- Ambas se basan en el principio de localidad
  - Guardar los bloques activos en la memoria de mayor velocidad
  - Guardar los inactivos en la memoria más lenta
- Si el algoritmo tiene éxito:
  - El rendimiento se aproxima al de la memoria más rápida
  - El coste a la más barata



- Sus diferentes raíces históricas dan lugar a terminologías diferentes:

	CACHE	VIRTUAL
BLOQUE	Línea	Página
FALLO	Miss	Fallo de página

MEMORIA VIRTUAL VS MEMORIA CACHE: DIFERENCIAS

- Gestionan diferentes niveles de la jerarquía de memoria
  - **Gestión de la memoria cache**
    - Controla la transferencia de información entre la memoria cache y la memoria principal
    - Se implementa mediante Hardware específico (MMU o “Memory Management Unit”)
  - **Gestión de la memoria virtual**
    - Controla la transferencia de información entre la memoria principal y la memoria secundaria
    - Parte de esta gestión se realiza mediante hardware específico (MMU) y otra parte la realiza el S.O
- Tratamiento de la dirección:
  - La cache utiliza direcciones físicas que son reinterpretadas
  - La memoria virtual utiliza las direcciones virtuales generadas por el procesador que suelen ser de mayor tamaño que las direcciones físicas
    - Las direcciones virtuales se traducen a direcciones físicas
      - Se necesita un mecanismo de traducción de direcciones

MEMORIA VIRTUAL VS MEMORIA CACHE: DIFERENCIAS

- Influencia del tamaño de la dirección generada por el procesador:
  - Determina el tamaño de la memoria virtual (espacio virtual de cada proceso)
  - El tamaño de la cache y de la memoria principal es independiente del tamaño de la dirección del procesador
- Existen importantes diferencias en la penalización de fallos (ver tabla)
  - Determinadas por los tiempos de acceso al disco duro
- Efecto de estas diferencias sobre las estrategias de gestión:
  - Los reemplazamientos por fallos de cache se gestionan mediante hardware
  - Los reemplazamientos en Memoria Virtual los gestiona el SO por software
  - Como la penalización es mucho mayor conviene que tenga una tasa de fallos menor.
  - El SO toma decisiones con algoritmos más lentos pero más efectivos

MEMORIA VIRTUAL VS MEMORIA CACHE: DIFERENCIAS

Parámetros	Cache L1	MV
Tamaño bloque (o página)	16-128 bytes	4096-65.536 bytes
Tiempo acierto	1-3 ciclos	100-500 ciclos
Penalización fallo:	8-200 ciclos	1.000.000-10.000.000 ciclos
(Tiempos acceso)	(6-160 ciclos)	800.000-8.000.000 ciclos
(Tiempo transferencia)	(2-40 ciclos)	200.000-2.000.000 ciclos
Tasa de fallos	0,1-10%	0,00001-0,001%
Dirección	25-45 dirección física 14-20 dirección cache	32-64 dirección virtual 25-45 dirección física

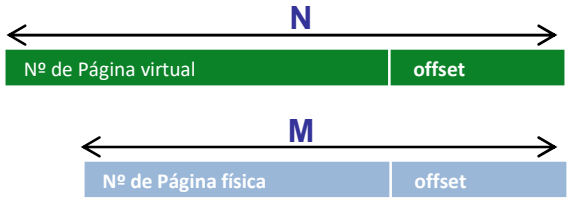
Ref. Hennessy & Patterson (5th ed.)

DEFINICIONES

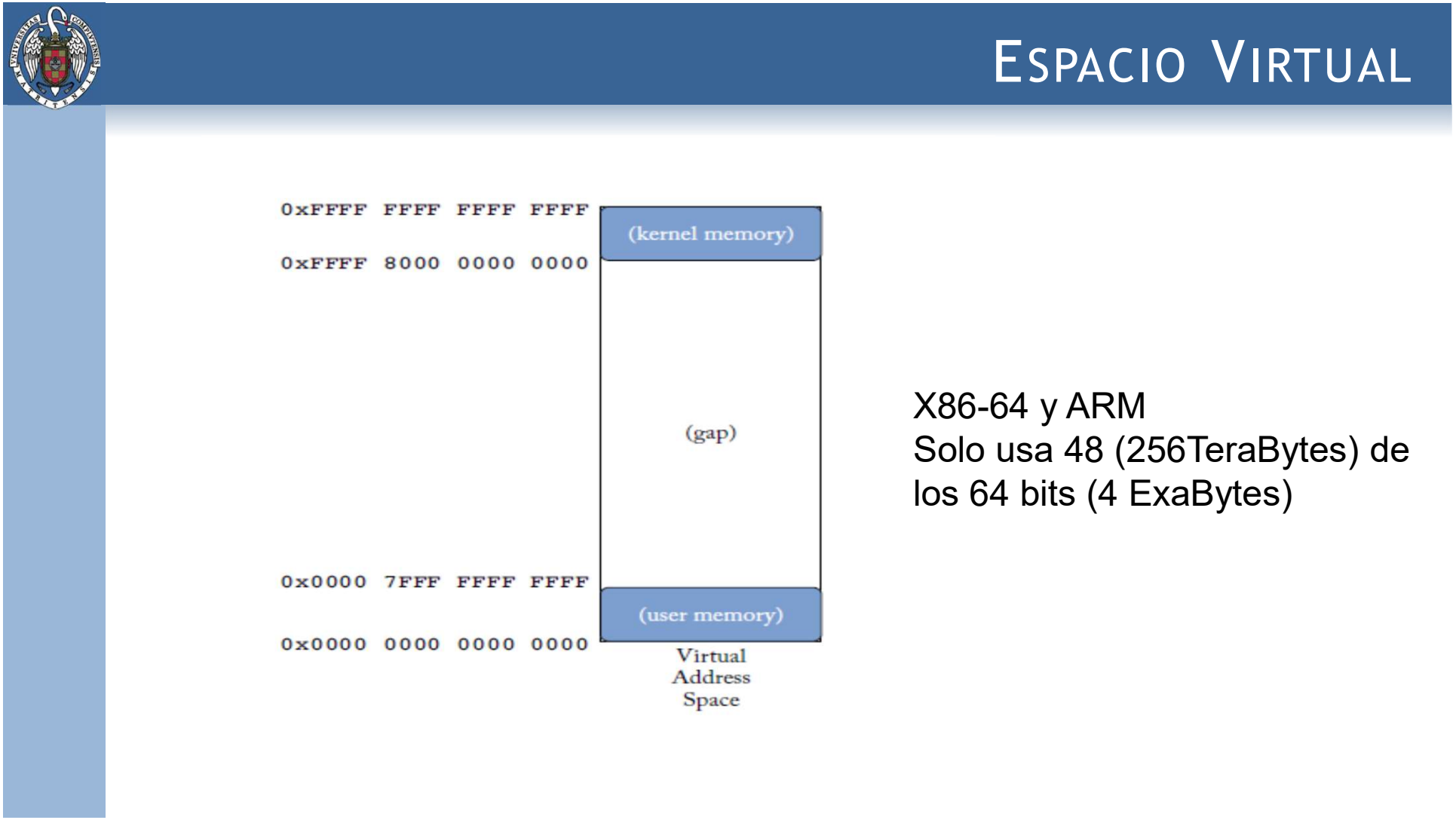
- Cómo funciona
  - Se divide la memoria principal en bloques del mismo tamaño llamados **páginas** físicas
  - Se divide el espacio de direcciones de un proceso en **páginas** virtuales del mismo tamaño que las físicas (el proceso está almacenado en Memoria Secundaria (MS))
  - Se carga en la MP un 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 (bajo demanda)
- 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
  - Lo 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 a qué 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 página 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 de un proceso puede ser mayor, menor o igual que el número de páginas físicas de la memoria

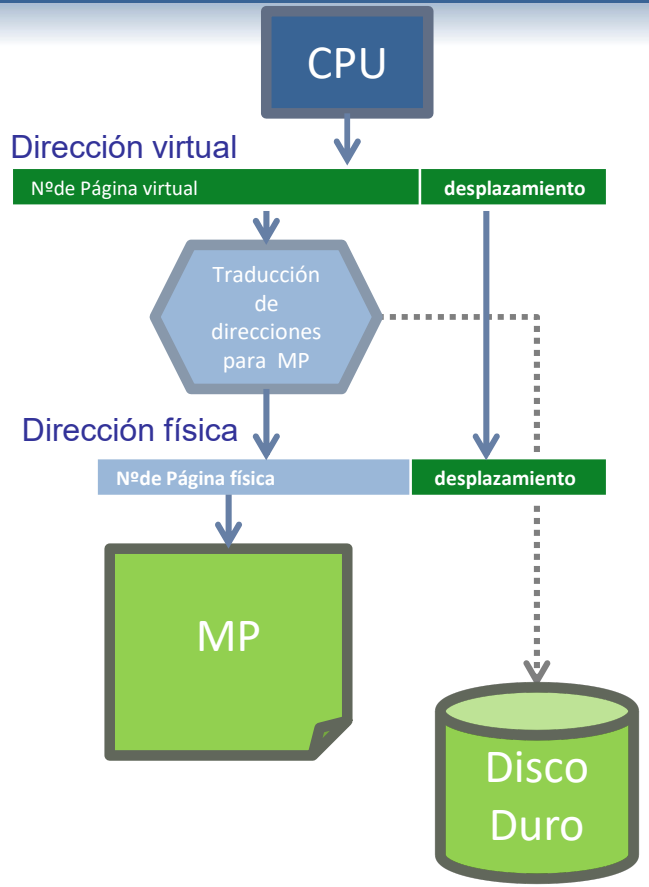


Espacio Virtual

X86-64 y ARM
Solo usa 48 (256TeraBytes) de los 64 bits (4 ExaBytes)

MECANISMO DE TRADUCCIÓN

- Traduce la dirección virtual a la dirección física
  - se traduce el número de página virtual al número de página 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 para cada acceso a memoria
- Tiene que implementar un mecanismo para localizar/direccionar las páginas virtuales que se encuentran en MS



Mecanismo de traducción y emplazamiento

- El emplazamiento determina en qué página física se puede ubicar una página virtual
  - Determina el mecanismo de traducción
- Debido a la 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 página 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 página virtual puesto que ésta puede estar en cualquier página física
  - 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

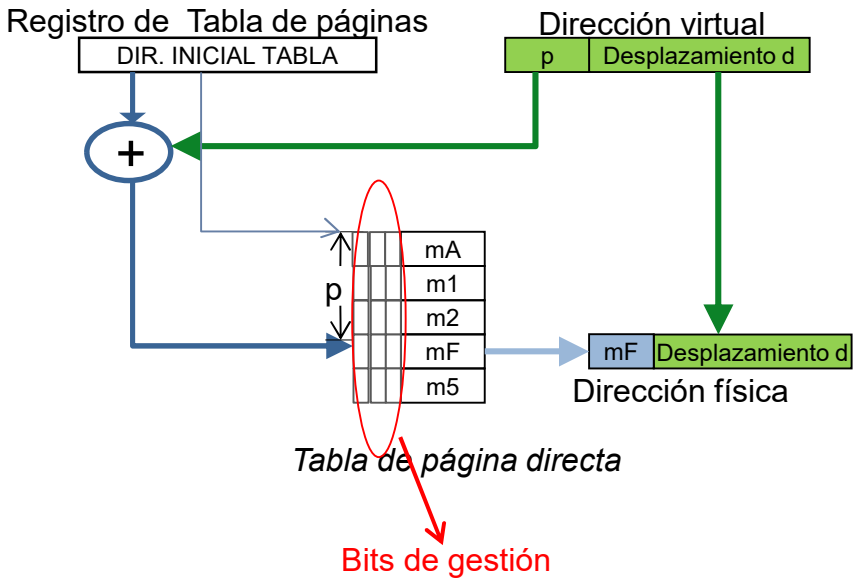
TABLA DE PÁGINAS

- La tabla tendrá, como máximo, tantas entradas como posibles páginas virtuales tenga nuestra arquitectura:
  - 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 (Tamaño)
- La crea el SO cuando se crea un proceso por primera vez
- Otra información que puede contener una entrada de la tabla de páginas:
  - 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 etiquetas

TABLA DE PÁGINAS

- Para su implementación necesita un **Registro de tabla de páginas**
  - Contiene la dirección de MP en la que comienza la tabla
  - El número de página virtual actúa como un desplazamiento que se suma al contenido del registro de tabla de páginas 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

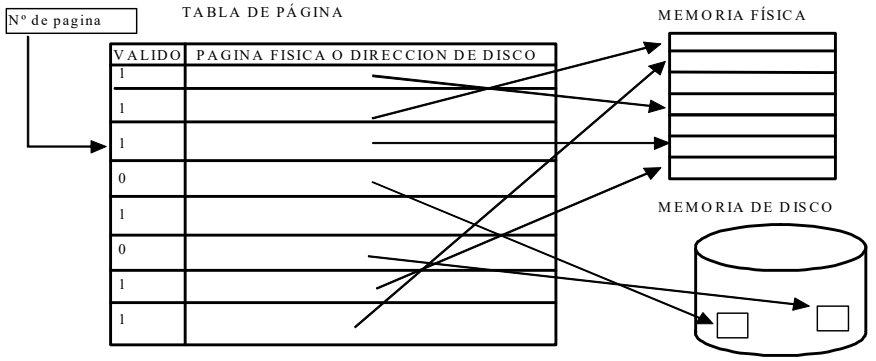


Fallo de página

- ¿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 dónde ubicar la nueva página en Memoria Principal → reemplazamiento

FALLO DE PÁGINA

- ¿ Cómo se busca la página en el Disco Duro?
  - La dirección virtual no nos indica dónde está la página en el disco duro
  - El SO conoce la localización en disco duro de cada página virtual
  - Como no se conoce de antemano cuándo una página en MP será elegida para ser reemplazada el SO reserva un espacio en disco para todas las páginas del proceso cuando lo crea. Espacio de swap.
  - Al mismo tiempo crea una estructura de datos para guardar en qué 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

- Reemplaza 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, y no hay hueco para más páginas, se reemplaza la 12
- Implementar un sistema exacto de LRU es muy caro puesto que hay que actualizar la estructura de datos en 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 qué páginas fueron accedidas en un periodo de tiempo
    - El SO puede seleccionar una de las páginas menos recientemente usadas (bit de referencia desactivado)

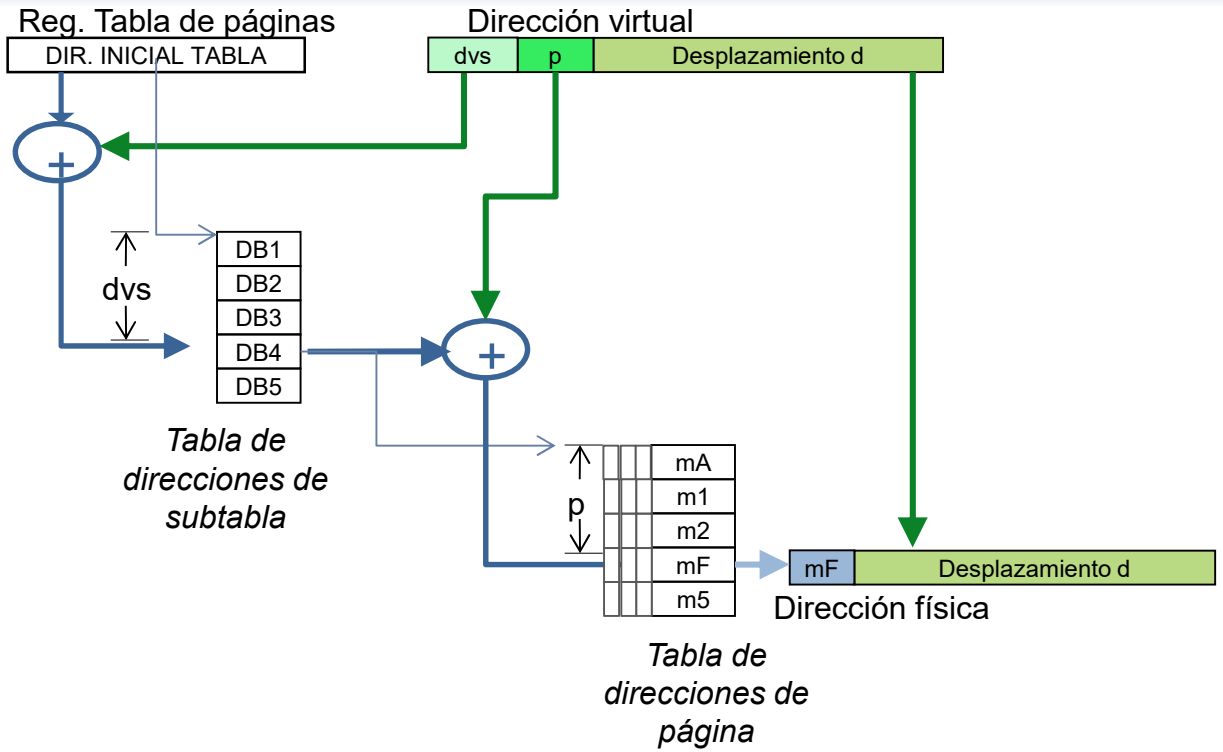
PROBLEMA DE LA TABLA DE PÁGINAS

- Sea un 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 entradas * Nº bytes 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

TABLA DE PÁGINAS DE DOS NIVELES

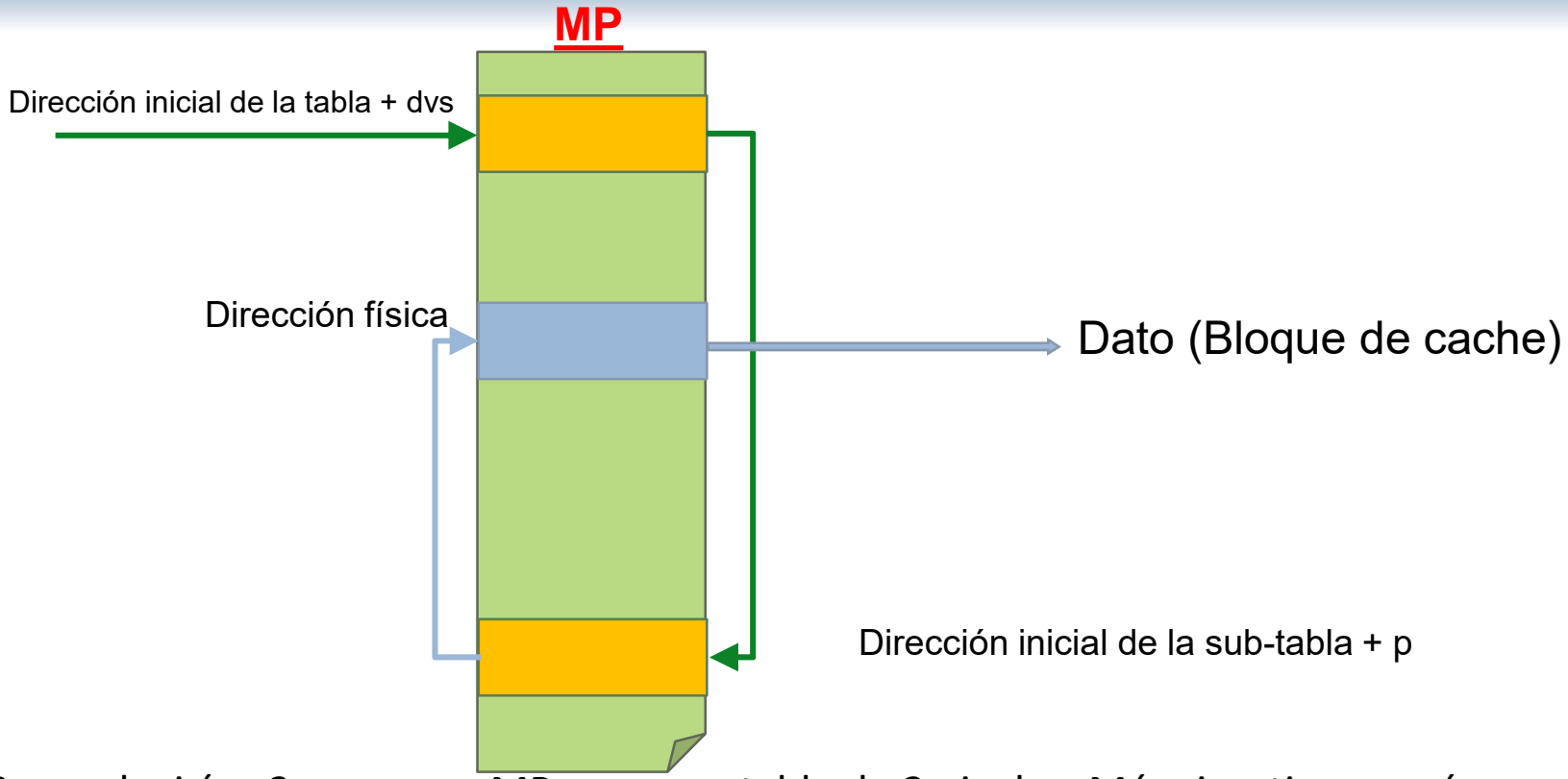


Ejemplo:

X86-32bits. DV 32bits, DF 40bits. Tabla de 2 niveles

X86-64bits. DV 48bits, DF 52bits. Tabla de 4 niveles.

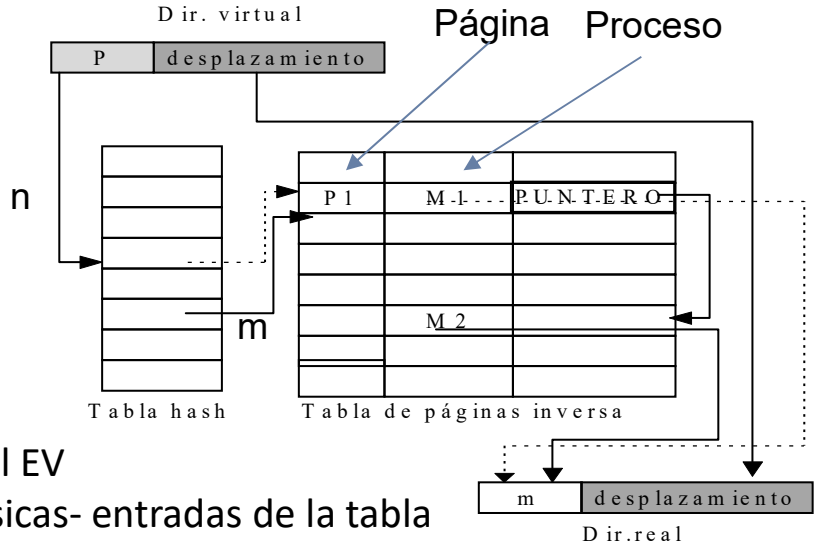
TABLA DE PÁGINAS DOS NIVELES



Se producirían 3 accesos a MP para una tabla de 2 niveles. Más si se tienen más niveles (5,6)

TABLA INVERSA

- En lugar de una entrada por página virtual de cada proceso tiene una entrada por cada página física
- Ventaja: tamaño de la tabla constante (memoria física) independientemente del tamaño del proceso y número de procesos activos
- La página virtual se asocia a la página inversa a través de una función hash
- PowerPC, SPARC



2^n número de páginas del EV

2^m número de páginas físicas- entradas de la tabla

TABLA DE PÁGINAS

- Problema de las tablas de páginas: Tiempo de acceso a un dato
  - Puesto que las tablas están almacenadas en MP para obtener un dato se necesitan al menos 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

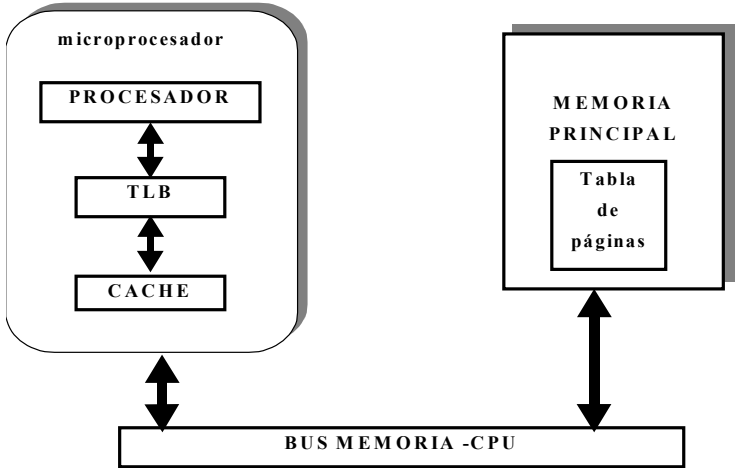
Translation Lookaside Buffer (TLB)

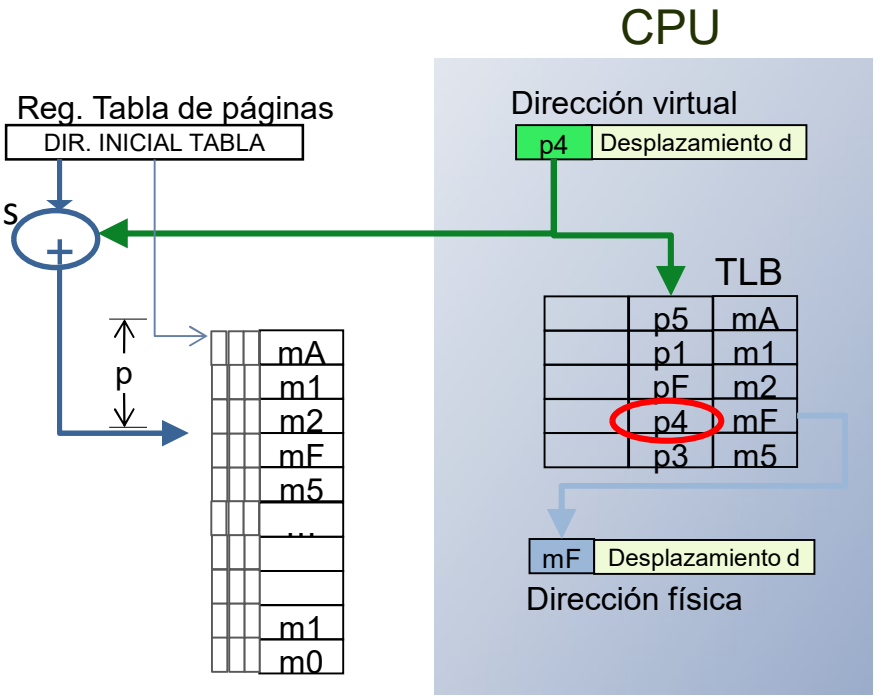
- Solución: Translation Lookaside Buffer (TLB)
  - Cache de traducciones que guarda información sobre las traducciones recientes
  - Muy pequeña y muy rápida
  - Se basa en la localidad de referencias
    - Si una dirección se ha traducido, la probabilidad de que se vuelva a traducir en un futuro cercano es muy elevada
  - Está situada en el chip del procesador
  - La etiqueta almacena el 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 otros bits de información necesarios para la gestión
  - Bit dirty: ¿se ha escrito?
  - Bit de referencia: ¿se ha accedido?
  - Bit de validez: ¿Tiene información válida?

TRANSLATION LOOKASIDE BUFFER TLB

- El sistema de memoria virtual está formado por dos tablas :

1. TLB
   - Contiene las últimas traducciones
   - En el chip procesador
2. Tabla de páginas
   - Contiene todas las direcciones de páginas
   - En Memoria Principal





Translation Lookaside Buffer TLB

- El TLB tiene la siguiente estructura:

V	R	D	Página Virtual	Página Física

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

- 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

TLB-ALTERNATIVAS DE DISEÑO

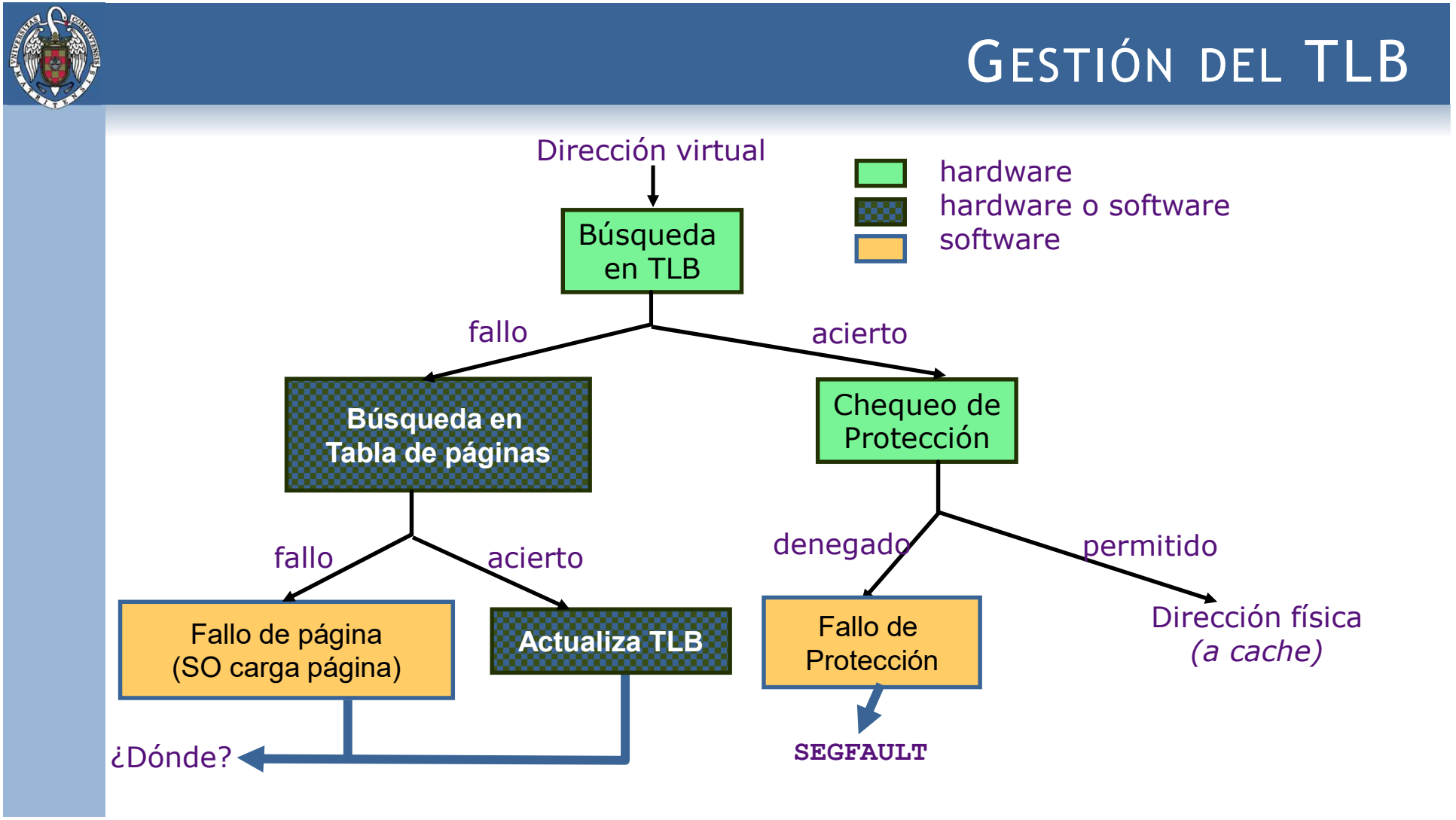
- Existen dos alternativas en el diseño de una TLB
  - Con identificadores de proceso
  - Sin identificadores de proceso
- TLB sin identificador de proceso
  - Se accede al TLB sólo con el número de página virtual
  - Cada vez que hay un cambio de proceso el sistema operativo debe invalidar la TLB ya 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
  - El SO 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
- Bit de validez del TLB se utiliza para invalidar la información

GESTIÓN DEL 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 acceso 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 página está en memoria MP es un fallo de TLB
      - Acción: se copia la información con la traducción de DV a DF 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 páginas en MP, un fallo de TLB será mucho más frecuente que un fallo de página

GESTIÓN DEL TLB

- Los fallos de TLB pueden tratarse:
  - SW (excepción→ SO→ reejecutar). (ciclo de fallo TLB)
  - 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
  - Tiempo de acierto: 0,5 -1 ciclo de reloj
  - Penalización de fallo: 10-100 ciclos de reloj
  - Tasa de fallos: 0,01-1%
- Mejora. TLB multinivel. Ejemplo: Intel Skylake páginas 4KB
  - L1 TLB 64 entradas. 64x4KB= 256KB
  - L2 TLB 1536 entradas. 1536x4KB= 6MB
  - L1 separado datos e instrucciones. L2 unificado



Gestión del 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 (permite cache con direcciones virtuales)
  - 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 puede ser más 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. (text, heap, stack)
      - 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 de 32KB
    - Pueden aparecer problemas de almacenamiento principal y secundario
    - 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.
- Existen “huge pages”, para aplicaciones con muchos datos.

ESCRITURA

- Los órdenes de magnitud de los tiempos de acceso a cada nivel son:
  - Cache: Varios ciclos
  - Memoria principal: decenas 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 si ha sido escrita
    - 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

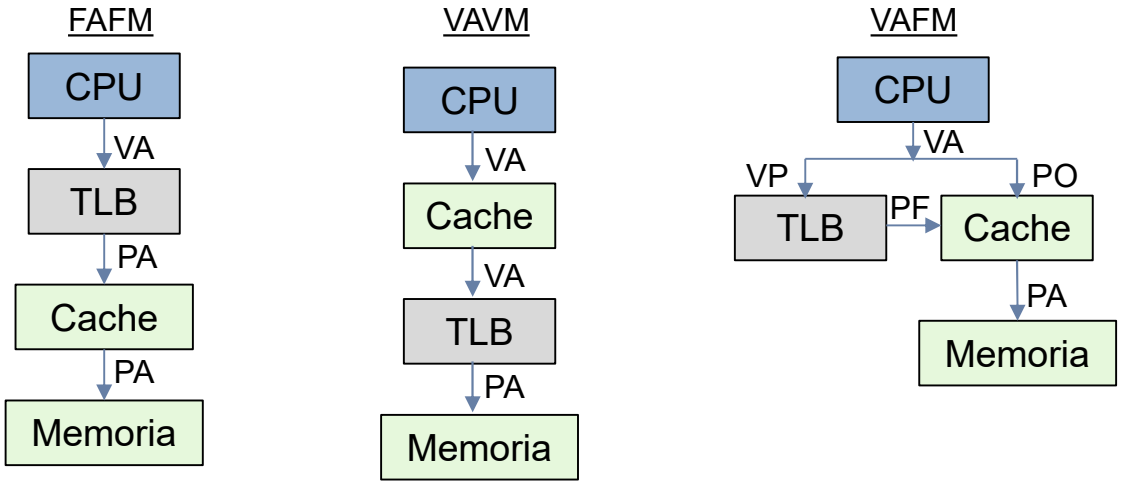
- Multiprogramación: varios programas se ejecutan concurrentemente compartiendo el procesador
  - Necesita mecanismos de protección y compartición
  - Concepto de proceso: un programa en ejecución más el estado que necesita para seguir ejecutándose
- Memoria virtual es el principal mecanismo que protege los procesos unos de otros
  - Existen diferentes niveles de protección
  - Las restricciones de protección se incluyen en cada entrada de la tabla de páginas
    - Permiso de lectura de la página
    - Permiso de escritura de la página
    - Permiso de ejecución del código de esa página
  - Dado que el SO es el que genera y maneja las tablas es el SO el que fija los niveles de seguridad (soporta compartición de datos)

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 Memoria Cache
  - Modificando la tabla de páginas y el 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 accedidas, físicamente marcadas

Integración de la Memoria virtual y cache

- FAFM: Físicamente accedidas, físicamente marcadas
- VAVM: Virtualmente accedidas, Virtualmente marcadas
- VAFM: Virtualmente accedidas, físicamente marcadas

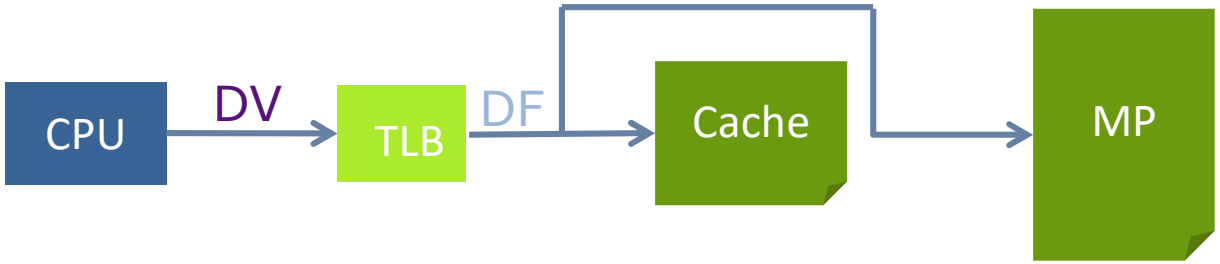


VA: Virtual address PA: Physical address

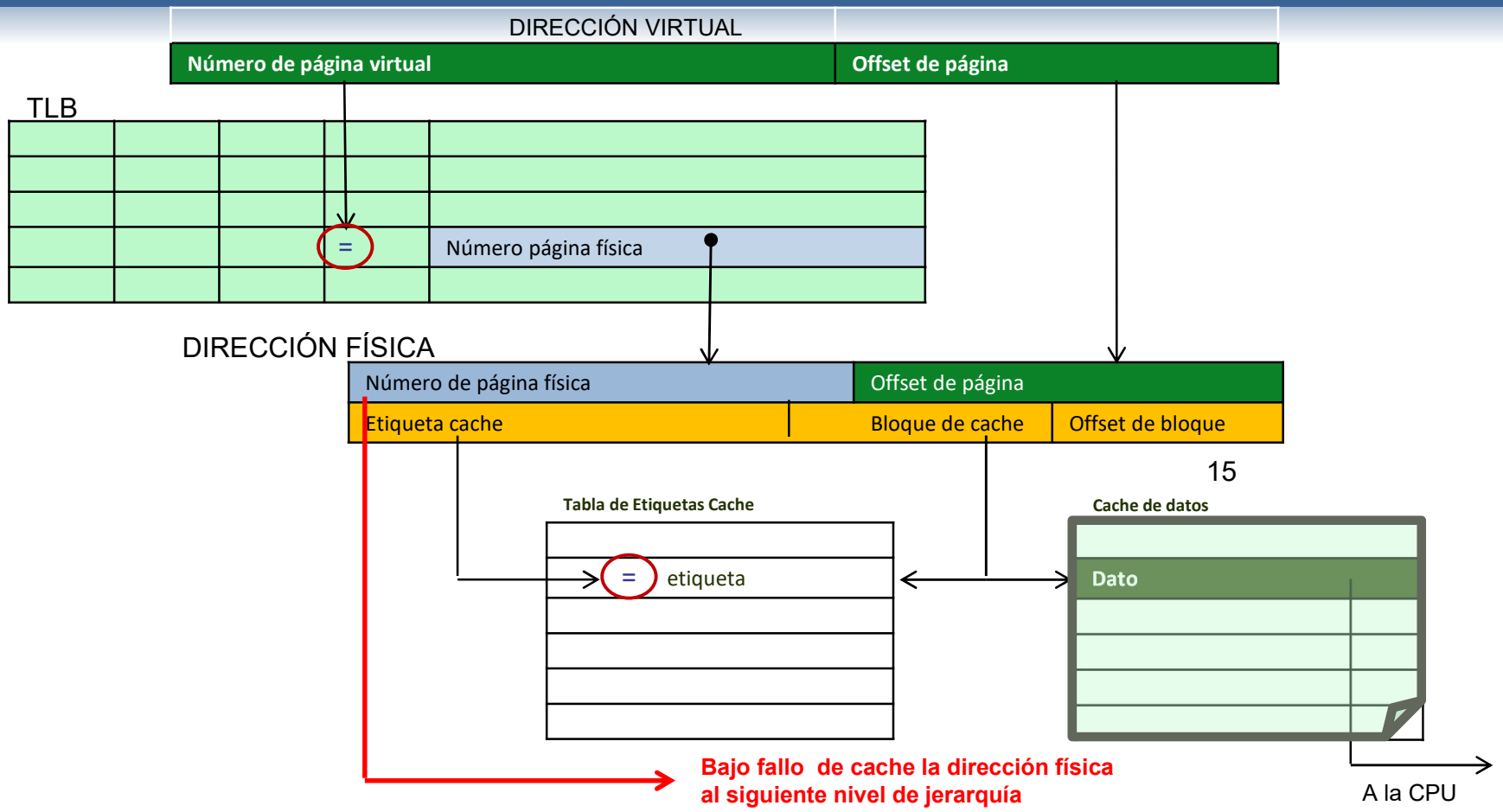
VP: Virtual page PF: Physical page PO: Page offset

CACHE DE DIRECCIONES FÍSICAS

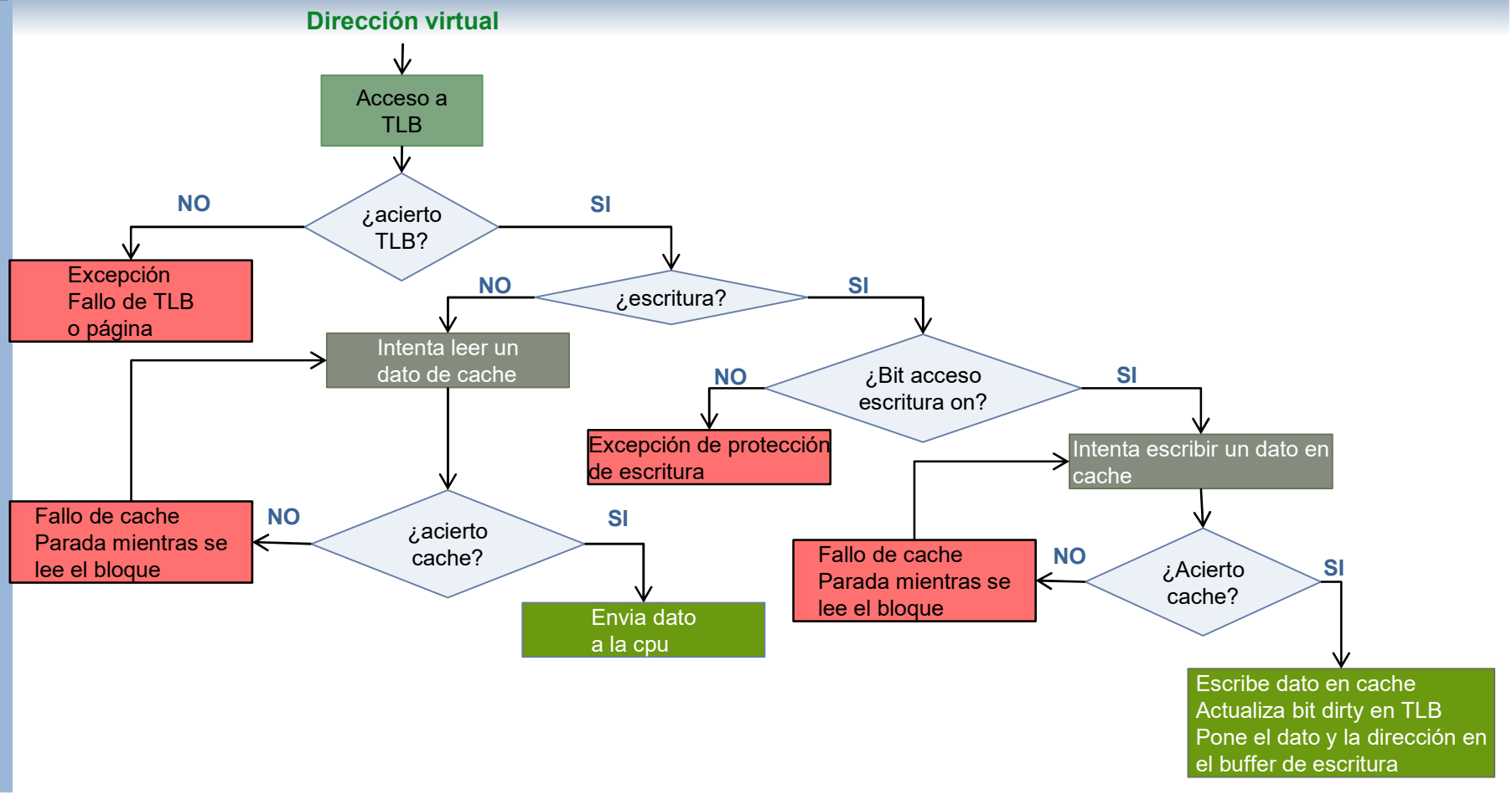
- La cache utiliza direcciones físicas
  - Utiliza la dirección física tanto para indexar como para comparar etiquetas
- La dirección virtual generada por el procesador se traduce siempre a través del TLB
- Permite que en la memoria cache convivan varios contextos es decir varios procesos activos.
 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



CACHE DE DIRECCIONES FÍSICAS

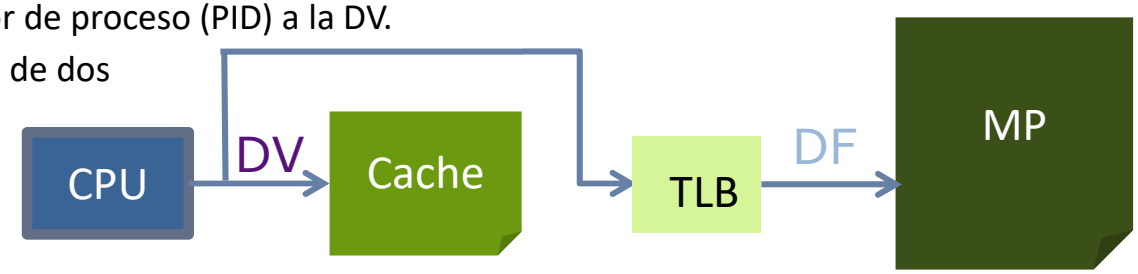


CACHE DE DIRECCIONES FÍSICAS

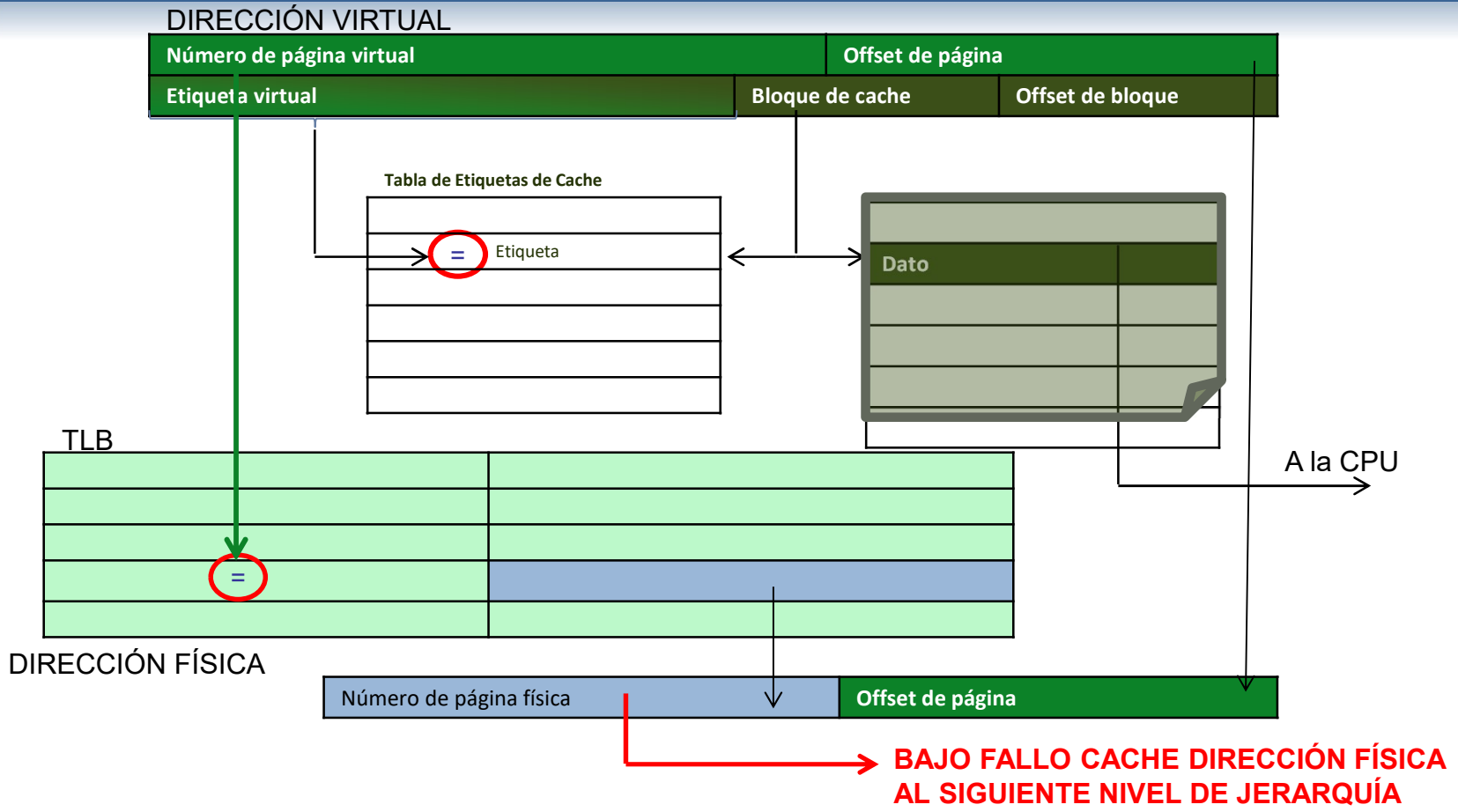


CACHE DE DIRECCIONES VIRTUALES

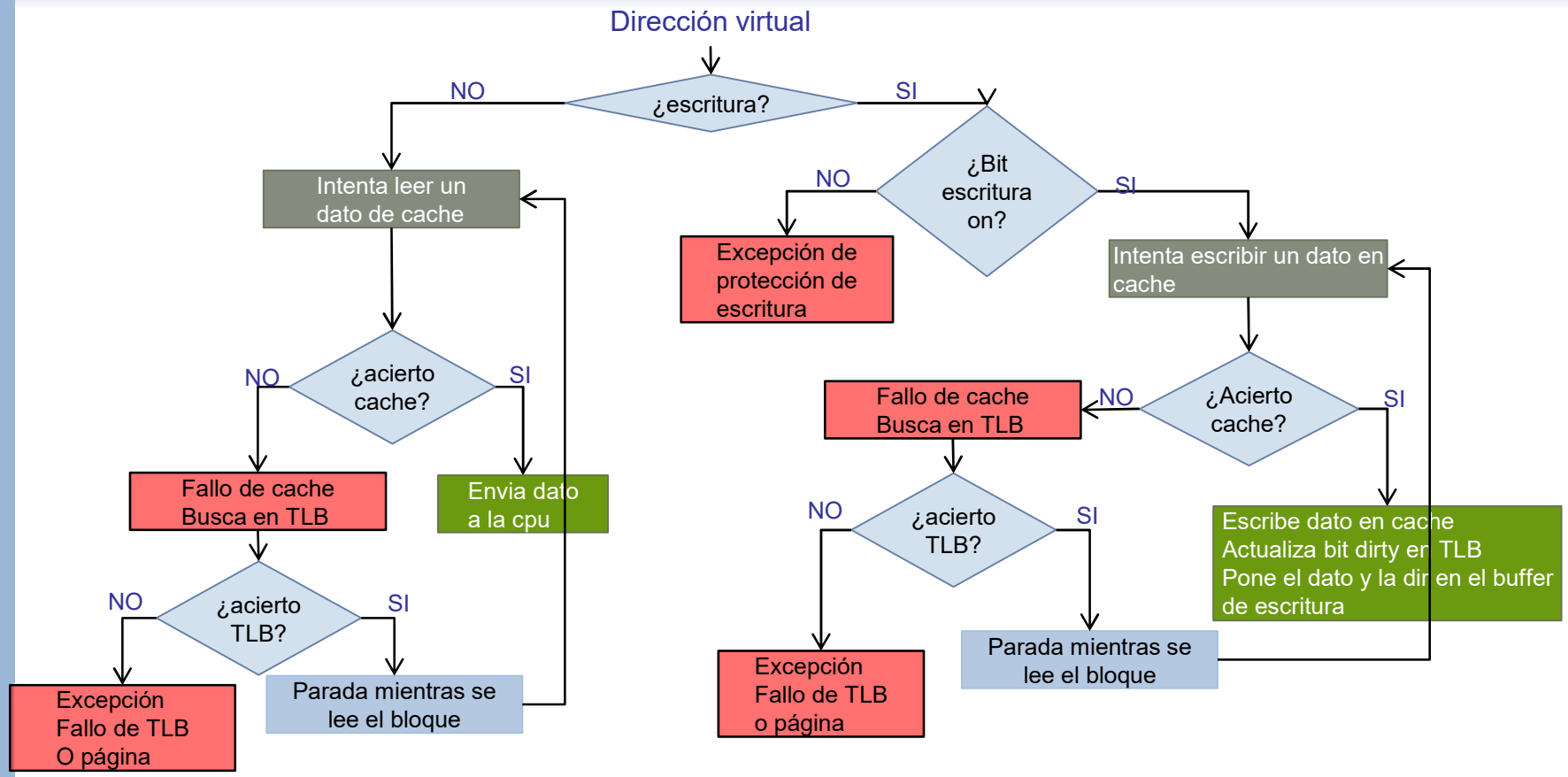
- Se accede a la cache con la dirección virtual
  - Utiliza la dirección virtual tanto para indexar como para comparar etiquetas
  - La dirección virtual se reinterpreta NO se traduce
- Ventaja: acceso a la cache más rápido puesto que no se pasa por el TLB
  - Sólo se pasa por el TLB cuando hay un fallo 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 (paso de la ejecución de un proceso a otro distinto):
  - Hay que borrar la cache.
    - De lo contrario: falsos aciertos ya que diferentes procesos pueden generar las mismas direcciones virtuales.
  - 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.



CACHE DE DIRECCIONES VIRTUALES



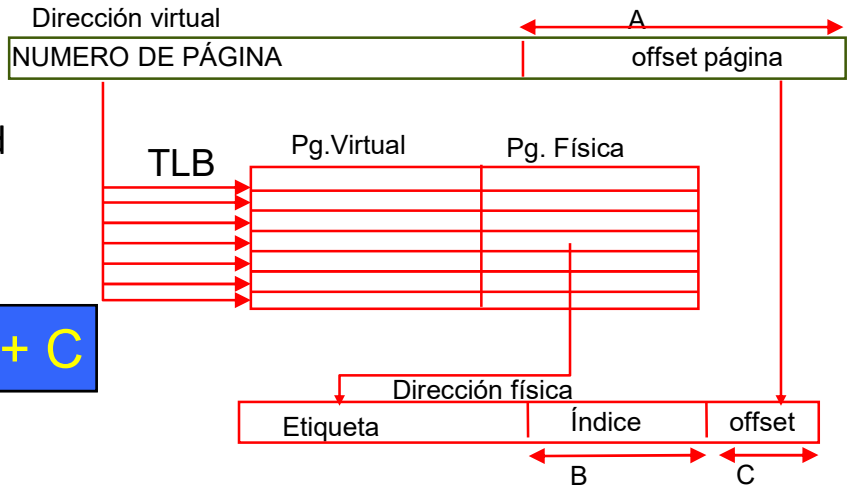
CACHE DE DIRECCIONES VIRTUALES



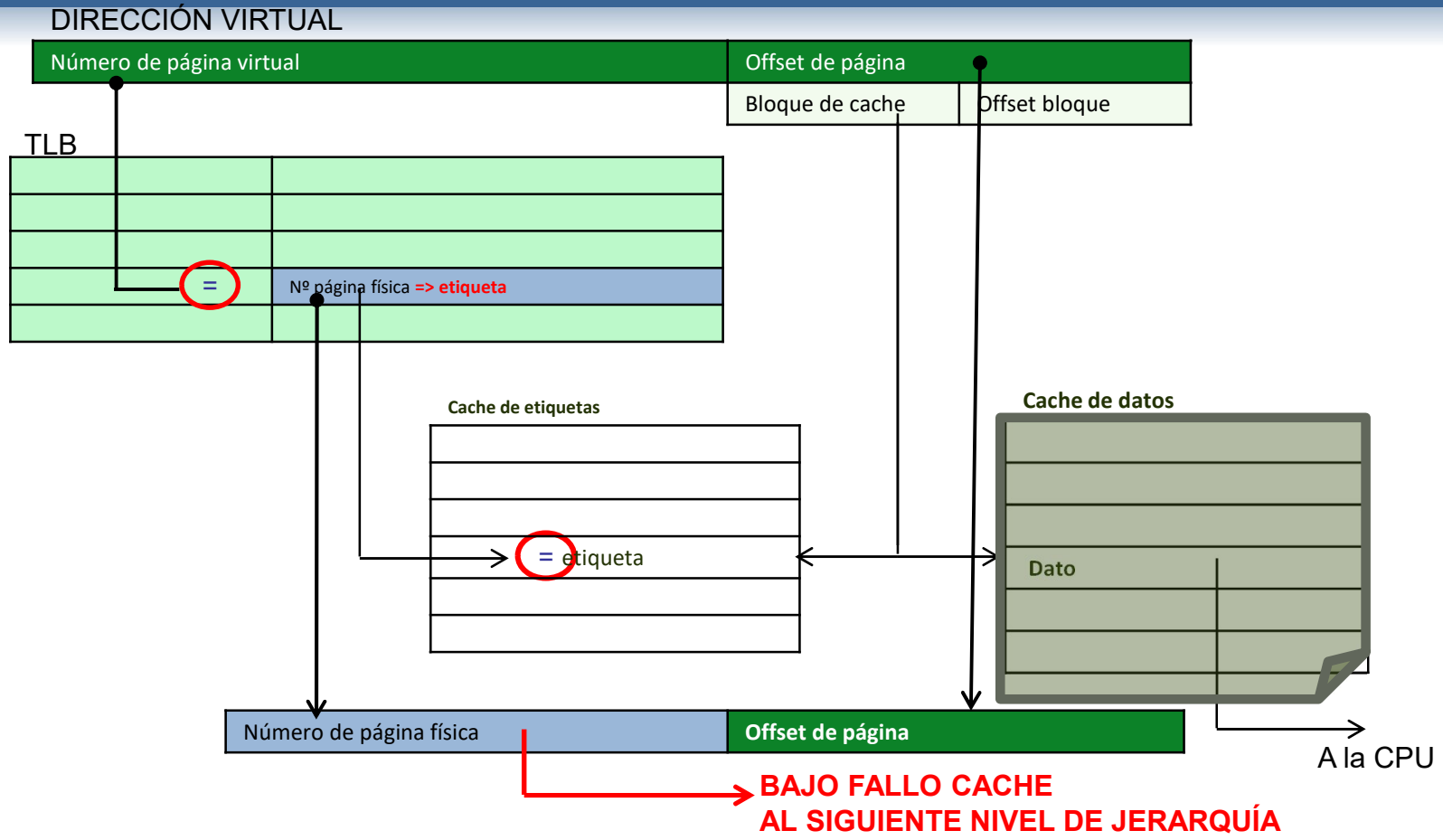
CACHE FÍSICAMENTE MARCADA VIRTUALMENTE ACCEDIDA

- Utiliza el offset de la dirección virtual, que es igual para la dirección física, para indexar la cache
- Se realizan en paralelo las siguientes acciones:
  - Lectura del dato usando el offset de la dirección 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

$A \geq B + C$

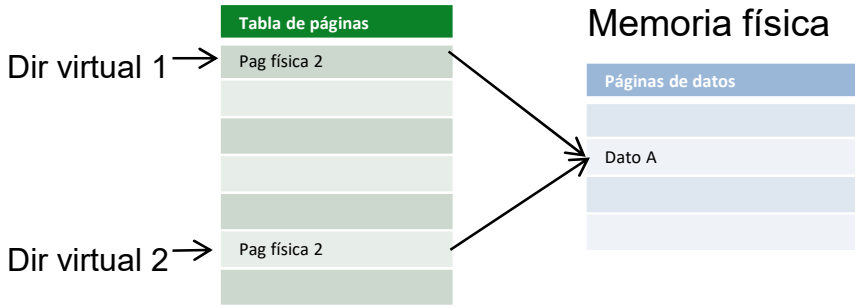


CACHE FÍSICAMENTE MARCADA VIRTUALMENTE ACCEDIDA



PROBLEMA DE LOS SINÓNIMOS

- Se puede dar en sistemas con memoria compartida con:
  - Caches virtuales
  - Virtualmente accedidas físicamente marcadas
- Dos direcciones virtuales apuntan a 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 la traducción se produce antes del 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.



Cache virtual

tag	dato
Dir virtual 1	Primera copia del dato A
Dir virtual 2	Segunda copia del dato A

MEMORIA VIRTUAL EN LOS SISTEMAS ACTUALES

- Servidores/PCs/portátiles/smartphones tienen memoria virtual
  - Portabilidad entre modelos con diferentes tamaños de memoria
  - Protección entre múltiples usuarios o múltiples tareas
  - Compartir memoria física (no muy grande) entre diferentes tareas (activas)

- La mayoría de los procesadores empotrados y DSPs no tienen memoria virtual
  - No pueden permitirse el *lujo* de tener memoria virtual (área/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

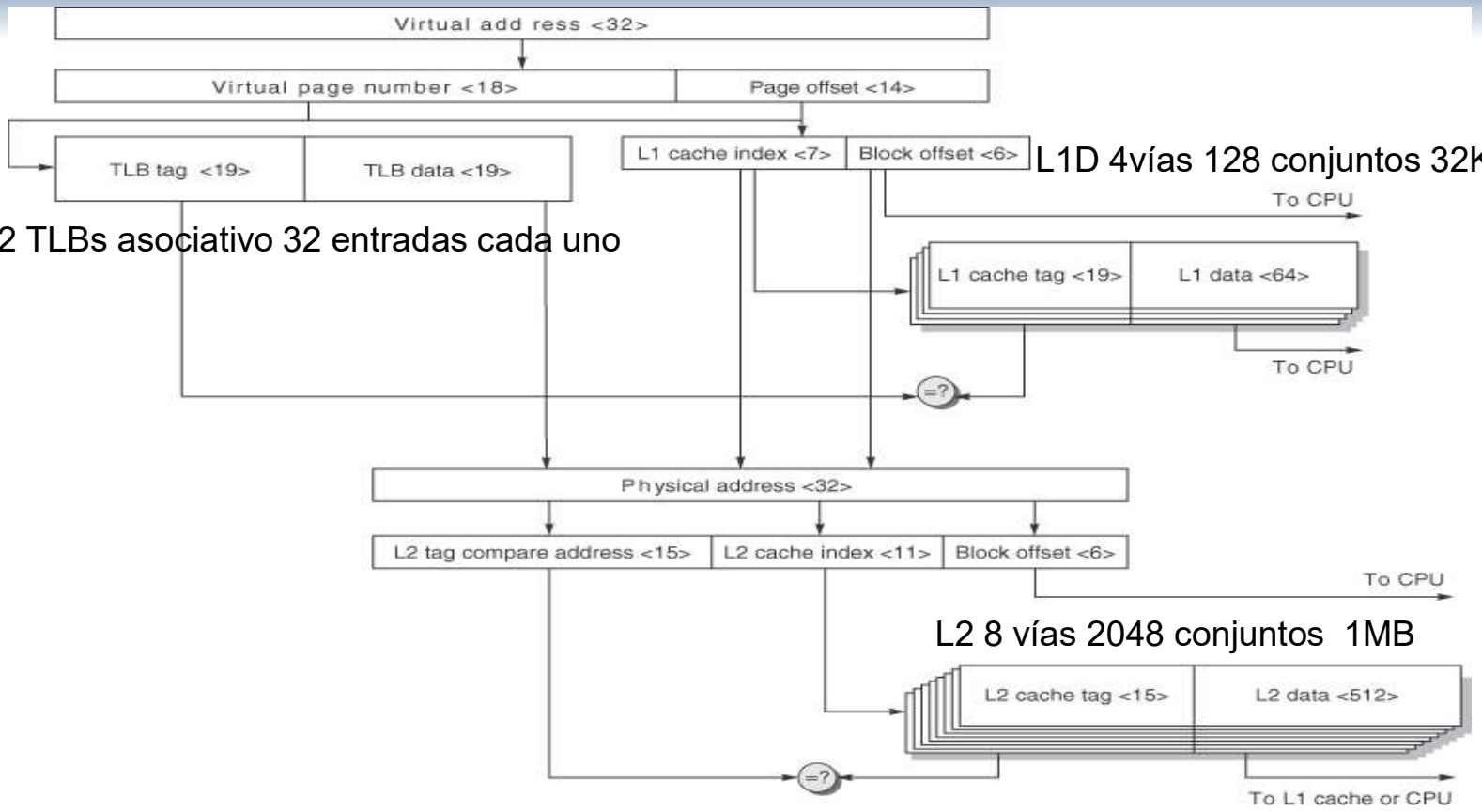
VISIÓN GLOBAL: ARM CORTEX A8 Y CORE I7

Characteristic	ARM Cortex-A8	Intel Core i7
Virtual address	32 bits	48 bits
Physical address	32 bits	44 bits
Page size	Variable: 4, 16, 64 KiB, 1, 16 MiB	Variable: 4 KiB, 2/4 MiB
TLB organization	1 TLB for instructions and 1 TLB for data Both TLBs are fully associative, with 32 entries, round robin replacement TLB misses handled in hardware	1 TLB for instructions and 1 TLB for data per core Both L1 TLBs are four-way set associative, LRU replacement L1 I-TLB has 128 entries for small pages, 7 per thread for large pages L1 D-TLB has 64 entries for small pages, 32 for large pages The L2 TLB is four-way set associative, LRU replacement The L2 TLB has 512 entries TLB misses handled in hardware

Simple

Complejo

VISIÓN GLOBAL: ARM CORTEX A8



L1D 4vías 128 conjuntos 32K

2 TLBs asociativo 32 entradas cada uno

L2 8 vías 2048 conjuntos 1MB

VISIÓN GLOBAL: CORE I7 2010