Tema 7. Memoria virtual

Índice

1. Gestión de memoria

➢ Introducción

• El sistema de memoria virtual de los actuales computadores surgió para liberar al programador de una serie de tareas relacionadas con la gestión que los programas deben realizar con la memoria.

• La memoria virtual automatiza la gestión entre los dos niveles principales de la jerarquía de memoria: memoria principal y secundaria (disco).

• Las principales funciones que automatiza el sistema de memoria virtual son las siguientes:

- **Aumento del espacio de direcciones físicas**
  - Permite que se ejecuten programas de tamaño superior a la memoria principal
- **Reubicación de programas en memoria**
  - Permite que un programa se ejecute en cualquier ubicación de la memoria
- **Protección de memoria**
  - Impide que un programa acceda a la zona de memoria de otro programa
- **Compartición de memoria**
  - Permite que varios programas accedan a otros programas o datos comunes

1. Gestión de memoria

➢ Aumento del espacio de direcciones físicas (overlaying)

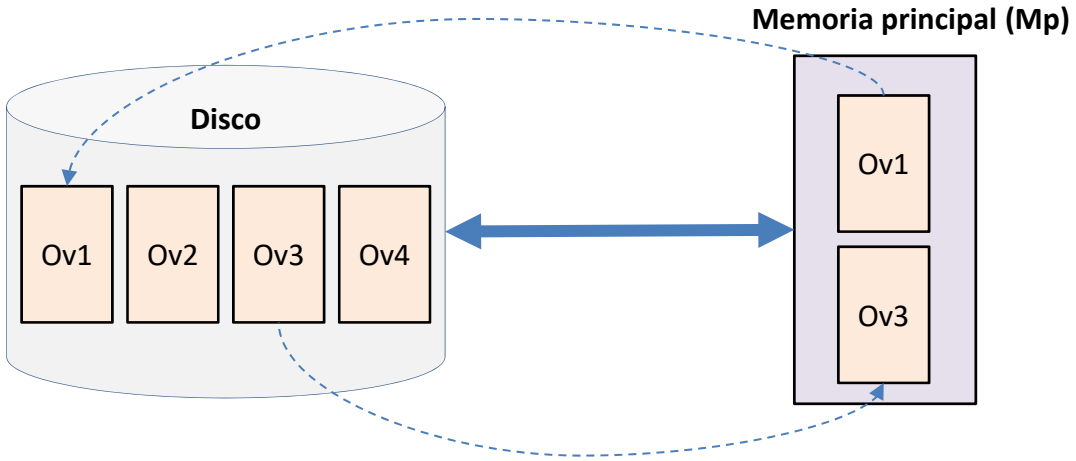
• La capacidad de la memoria principal disponible en los computadores ha aumentado de forma sostenida desde sus orígenes.

• Pero el tamaño de los programas ha crecido más rápidamente, por lo que la necesidad de ejecutar programas de tamaño mayor que la memoria principal ha sido una constante en la historia de los computadores.

• Una primera forma de superar esta limitación fue el uso de la técnica de solapamiento (*overlay*).

• Esta técnica consiste en dividir manualmente el programa en módulos de código denominados *overlays* residentes parte de ellos en memoria principal y la totalidad en memoria secundaria.

• Desde un *overlay* en ejecución se carga otro *overlay* desde la memoria secundaria cuando necesita continuar la ejecución por el código contenido en el segundo overlay**.**



• El Sistema de Memoria Virtual automatiza el mecanismo de intercambio entre memoria principal y secundaria, consiguiendo un espacio virtual de memoria mucho mayor que el de la principal.

1. Gestión de memoria

➢ Reubicación de programas en memoria principal

- **Proceso de ejecución de un programa en código fuente**

  Código fuente --> (Compilación) -- > Código objeto

  Código objeto -- > (Enlace) -- > Ejecutable

  Ejecutable --> (Carga) --> Programa en memoria (Ejecución)

- **Código absoluto**: las direcciones se generan en tiempo de compilación (y/o enlace)
  - Hay que conocer las direcciones de memoria donde se va a ejecutar el programa en tiempo de compilación/enlace.
  - El ejecutable no es reubicable.
- **Reubicación estática**: las direcciones se generan en tiempo de carga (el ejecutable tiene referencias relativas)
  - Una vez cargado no puede moverse a otro lugar de la memoria.
  - Solo puede haber intercambio si el programa vuelven a la misma ubicación de memoria.
- **Reubicación dinámica:** las direcciones se generan en tiempo de ejecución (el programa al ejecutarse maneja unas referencias que no son las direcciones de memoria reales a las que accede)
  - No hay problema con el intercambio, los programas pueden salir de memoria y volver a ella en cualquier ubicación
  - Aparece una distinción entre el espacio virtual o lógico de direcciones que maneja el programa y el espacio físico de direcciones al que realmente se accede.

- **Los mecanismos de paginación y segmentación del Sistema de Memoria Virtual implementa la reubicación dinámica.**

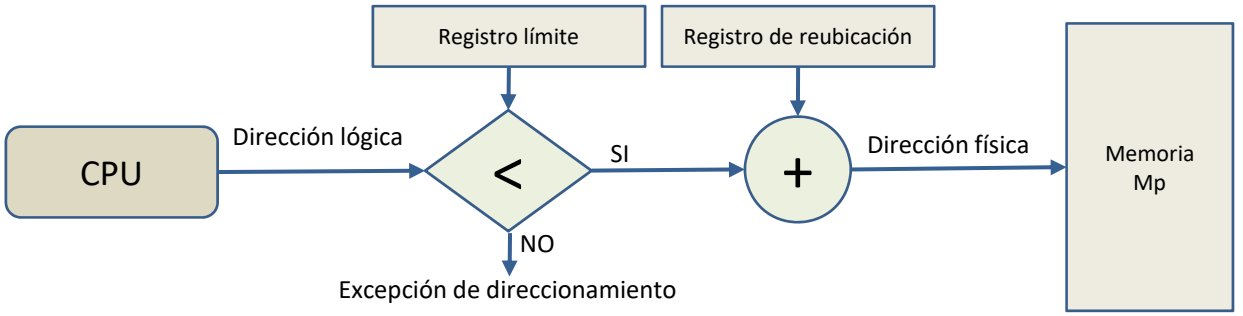
1. Gestión de memoria

➢ Protección de memoria

• La memoria debe estar protegida para que un programa no pueda acceder directamente a la memoria de otros programas.

• Se utilizaron dos alternativas para implementar la protección:

1. D**os registros límite**
   - o Cada vez que un programa genera una dirección se comprueba si es mayor que el registro límite inferior y menor que el superior.
   - o De no ser así se genera una excepción
2. **Un registro base y un registro límite**
   - o Cuando se genera una dirección se comprueba si es menor que el registro límite.
   - o En caso de serlo, la dirección física se obtiene sumando el valor del registro base.
   - o De no ser así se genera una excepción.



• **Los mecanismos de paginación y segmentación del Sistema de Memoria Virtual resuelven el problema de la protección de memoria**

1. Gestión de memoria

➢ Compartición de memoria

• Debe haber flexibilidad para permitir que varios programas accedan a un misma zona de memoria para compartir código o para utilizar una misma estructura de datos.

• Se trata de permitir que direcciones lógicas de dos o más programas, posiblemente distintas entre sí, se correspondan con la misma dirección física.

• La compartición de memoria no debe comprometer su protección básica

• **Los mecanismos de paginación y segmentación del Sistema de Memoria Virtual resuelven el problema de la compartición**

2. Memoria Virtual

➢ Principio de funcionamiento de la memoria virtual

• En un computador con memoria virtual (Mv) las direcciones de los programas (generadas por la CPU) hacen referencia a un espacio mayor que el espacio físico realmente disponible en la memoria principal o memoria física (Mp).

• Los programas operan *virtualmente* con un tamaño físico de memoria principal mucho mayor que el realmente disponible.

• Hay que diferenciar entre el espacio de direcciones virtuales generado por la CPU y el espacio de direcciones físicas o reales determinado por el tamaño de la memoria física.

• El espacio virtual se soporta sobre un disco con la ayuda de un mecanismo de traducción que genera la dirección física de memoria principal a partir de la virtual.

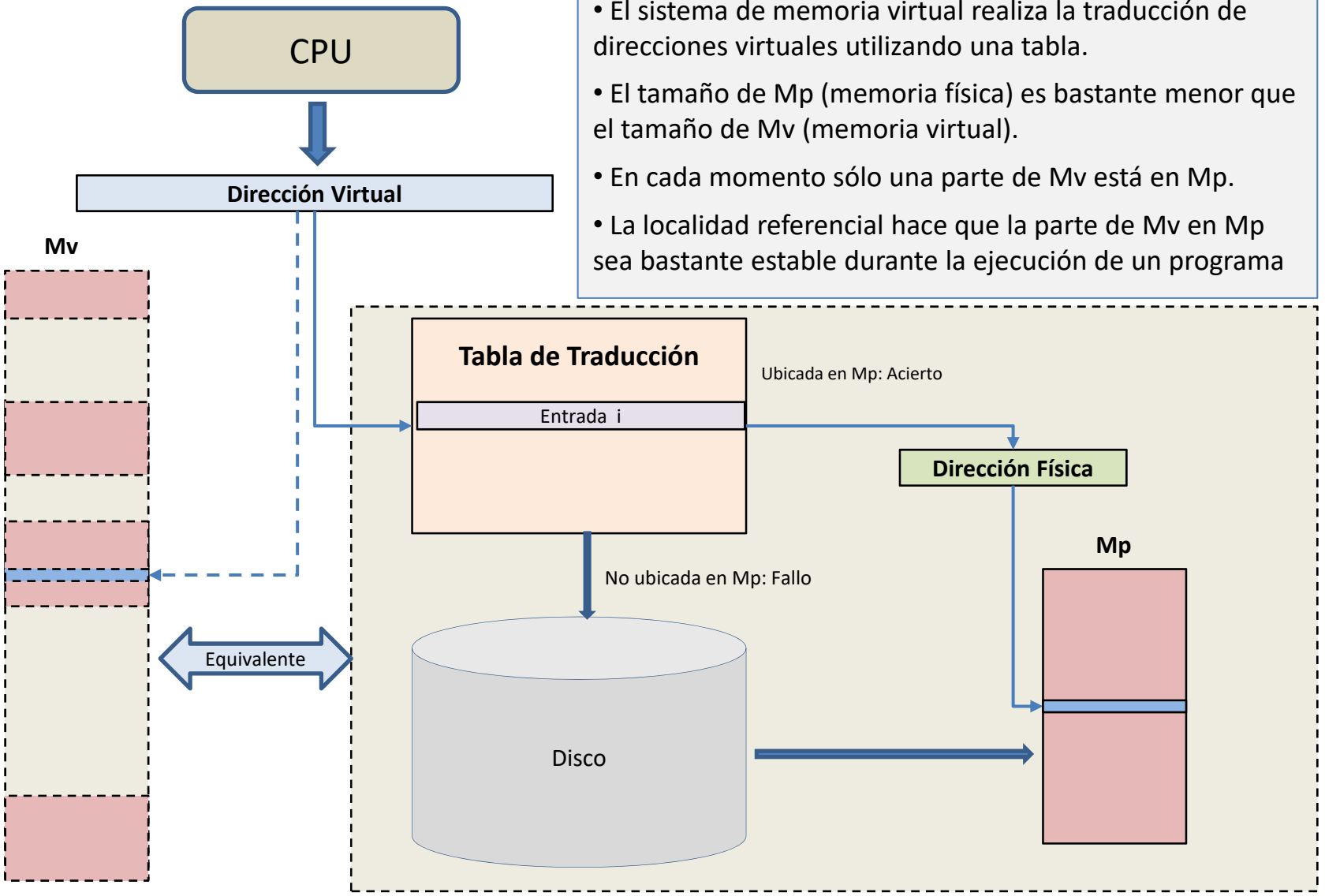
• Si la palabra referenciada no está en Mp habrá que trasladar un bloque de Mv (disco) a Mp.

• Los sistemas de Mv pueden clasificarse atendiendo al tamaño del bloque que se transfiere entre Mp y memoria secundaria:

1. **Memoria virtual paginada**: bloques de igual tamaño, la página
2. **Memoria virtual segmentada:** bloques de distinto tamaño, el segmento
3. **Memoria virtual segmentada/paginada:** segmentos de un número variable de páginas

2. Memoria Virtual

➢ Esquema de funcionamiento de la memoria virtual

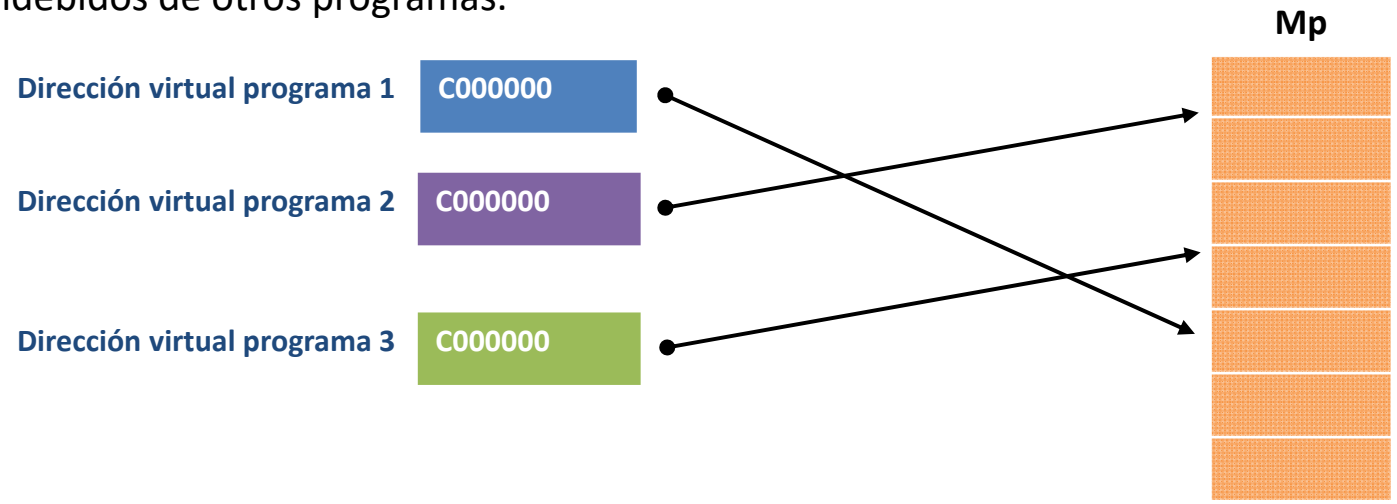


- El sistema de memoria virtual realiza la traducción de direcciones virtuales utilizando una tabla.
- El tamaño de Mp (memoria física) es bastante menor que el tamaño de Mv (memoria virtual).
- En cada momento sólo una parte de Mv está en Mp.
- La localidad referencial hace que la parte de Mv en Mp sea bastante estable durante la ejecución de un programa

2. Memoria Virtual

➢ Protección de memoria con MV

- Con un mecanismo de memoria virtual un programa sólo puede leer y escribir en la zona de Mp que le es asignada.
- Cada programa se compila con su propio espacio de direcciones, lo que implica que la misma dirección virtual (la proporcionada por el procesador) de dos programas diferentes se cargue en direcciones físicas diferentes.
- La memoria virtual traduce el espacio de direcciones virtuales 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.



3. Memoria virtual paginada

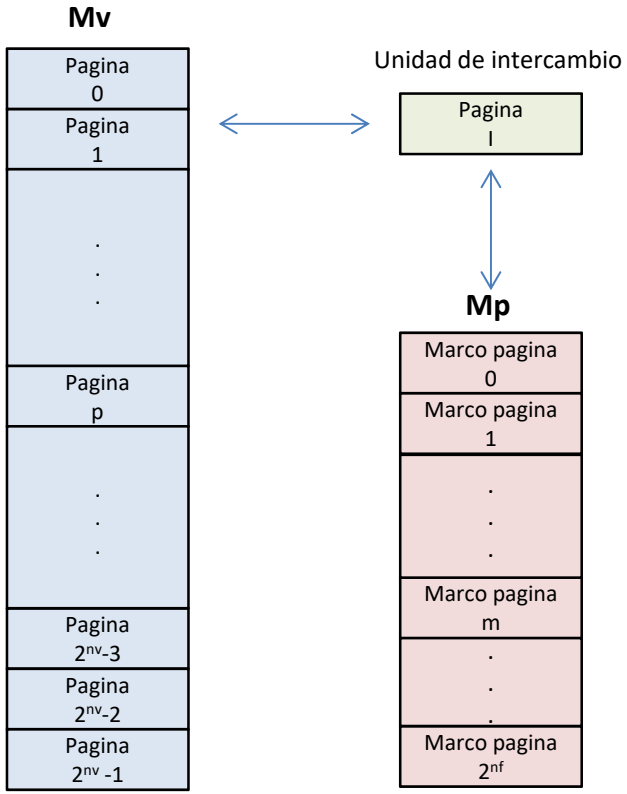
➢ Principio de funcionamiento

• Tanto la memoria principal como la memoria del disco se dividen en páginas de igual tamaño (Página en Mv y marco de página en Mp).

• El número de páginas de la memoria virtual es mayor que el número de marcos de página de la memoria física.

• En cada momento sólo las copias de un conjunto de páginas virtuales del programa residen en la memoria física: *conjunto de trabajo* o *conjunto activo*

• El conjunto activo resulta relativamente estable a lo largo del tiempo, debido a la localidad referencial que manifiestan los programas.



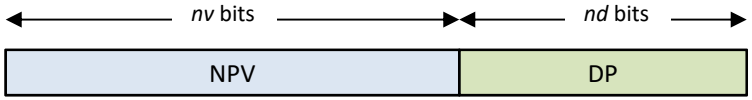
3. Memoria virtual paginada

➢ Direcciones virtuales y físicas

• Los bits de una DV se consideran divididos en dos campos, el *número de página virtual* (NPV) los más significativos, y el *desplazamiento dentro de la página* (DP), los menos significativos.

• El número de bits *nd* del campo DP lo determina el tamaño de página (*nd = log_2 tamaño de página*).

• El número de bits *nv* del campo NPV lo determina el número de páginas virtuales (*nv = log_2 nº de páginas virtuales*).

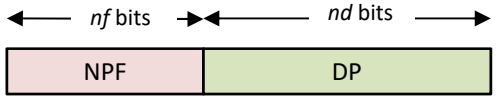


Dirección virtual (Generada por la CPU)

• Los bits *nf* de una DF se consideran divididos también en dos campos, el *número de página física* (NPF) los más significativos, y el *desplazamiento dentro de la página* (DP), los menos significativos.

• El número de bits del campo DP de una DF es el mismo que el de una DV puesto que las páginas tienen igual tamaño que los marcos de página.

• El número de bits del campo NPF lo determina el número de páginas físicas de Mp (*nº de bits de NPF = log_2 nº de páginas físicas*).



Dirección física

3. Memoria virtual paginada

➢ Tabla de páginas

• Las DVs generadas por la CPU se traducen a DFs con la ayuda de una Tabla de Páginas (TP).

• Esta tabla contiene en principio tantas entradas como páginas existen en la Mv, y la posición en la tabla de una entrada correspondiente a una página virtual concreta coincide con su NPV.

• Cada entrada contiene los siguientes campos:

H : bit de acierto (Hit)

H = 1 → Acierto de página

H = 0 → Fallo de página

C: Bit de limpieza de página (Clear)

C = 1 → Marco no modificado → no necesita actualizarse la página en el disco

C = 0 → Marco modificado → necesita actualizarse

RWX: Tipo de acceso permitido a la página: Lectura, escritura, ejecución

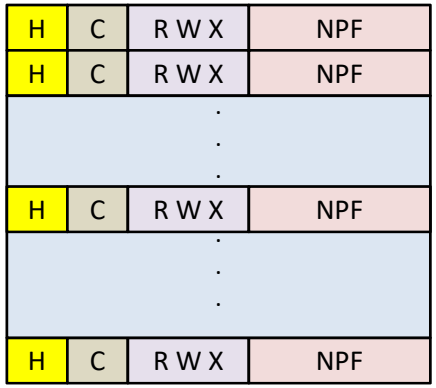
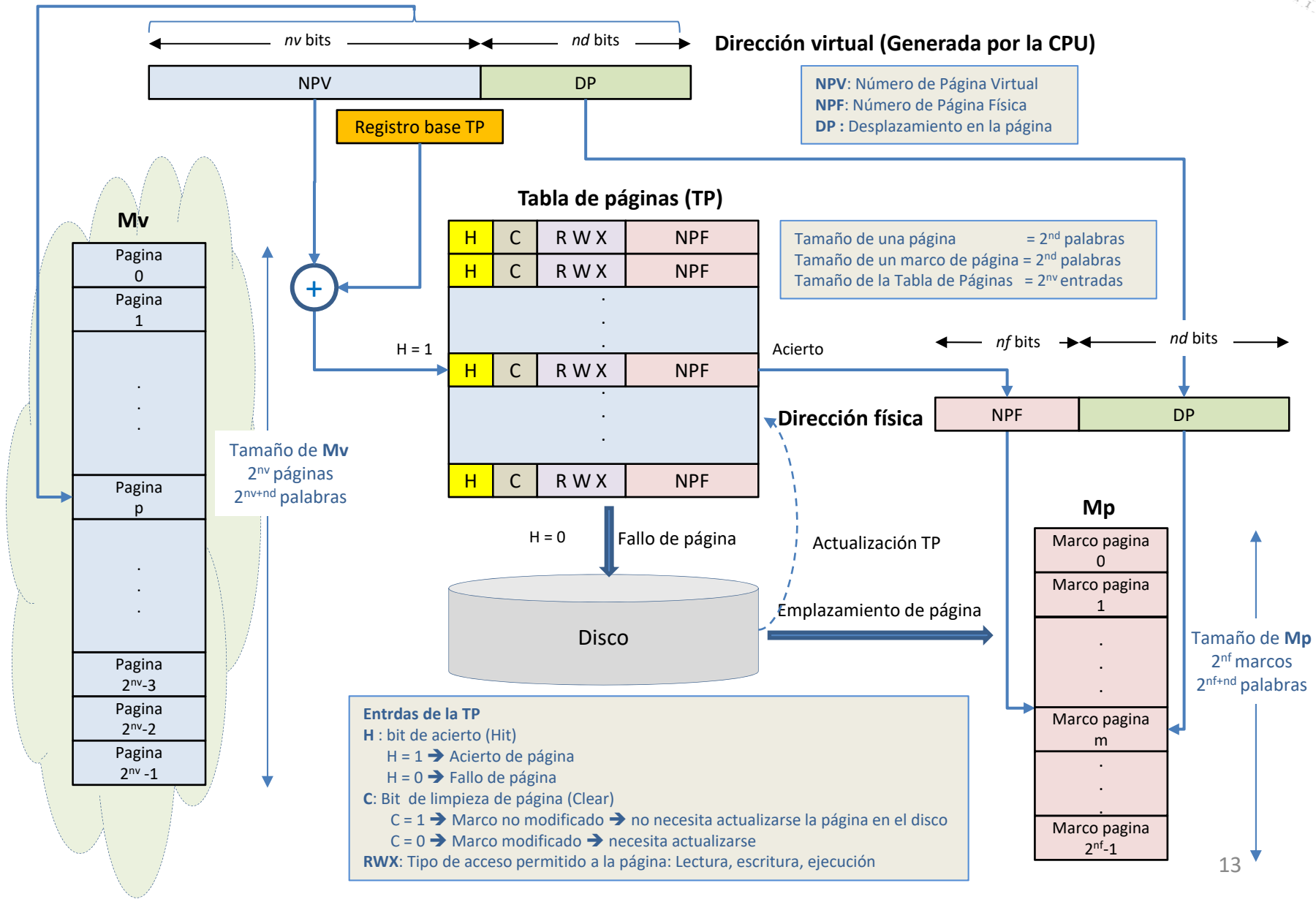


Tabla de páginas (TP)

3. Memoria virtual paginada

➢ Estructura funcional de la memoria virtual paginada

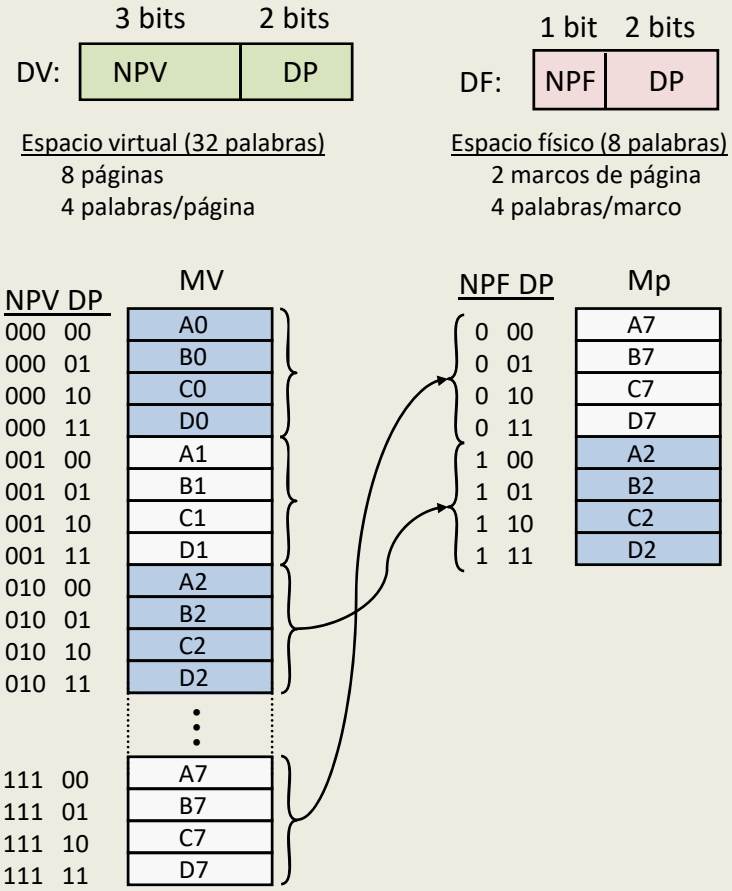
- Mecanismo de traducción de Dirección virtual a Dirección física



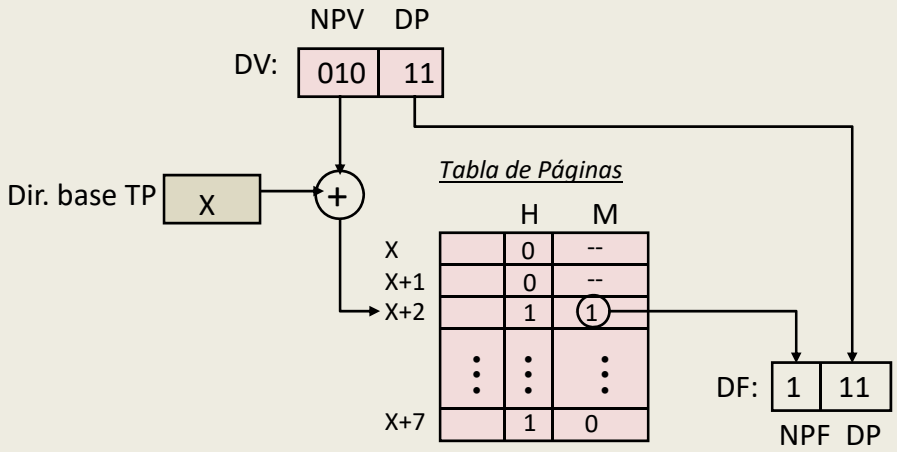
3. Memoria virtual paginada

➢ Ejemplo de traducción de direcciones

Estado de Mv y Mp



Traducción



3. Memoria virtual paginada

➢ Elección del tamaño de la página

- Es necesario tener en cuenta diversos factores:
  - Tamaño de las unidades de transferencia con memoria secundaria.
    - Es conveniente que el tamaño de la página coincida con un nº entero de sectores.
    - De esta forma se optimiza tanto el uso como el tiempo de acceso a MS (disco)
  - Tamaño de la tabla de páginas.
    - Si el tamaño de la página aumenta ⇒ el tamaño de la TP disminuye
  - Tamaño medio de las entidades lógicas del programa.
    - Si el tamaño de la página es similar al tamaño medio de las entidades lógicas del programa ⇒ se aprovecha al máximo la localidad espacial
  - Fragmentación interna
    - ⇨ Cuando aumenta el tamaño de página la cantidad de memoria no usada en la última página crece
- Ejemplos

Computador	Tamaño Página
IBM360/67	4KB
MC68020	256 bytes a 32 KB
MIPS R2000/3000	4KB
Alpha 21064	8KB
Pentium	4KB o 4 MB

3. Memoria virtual paginada

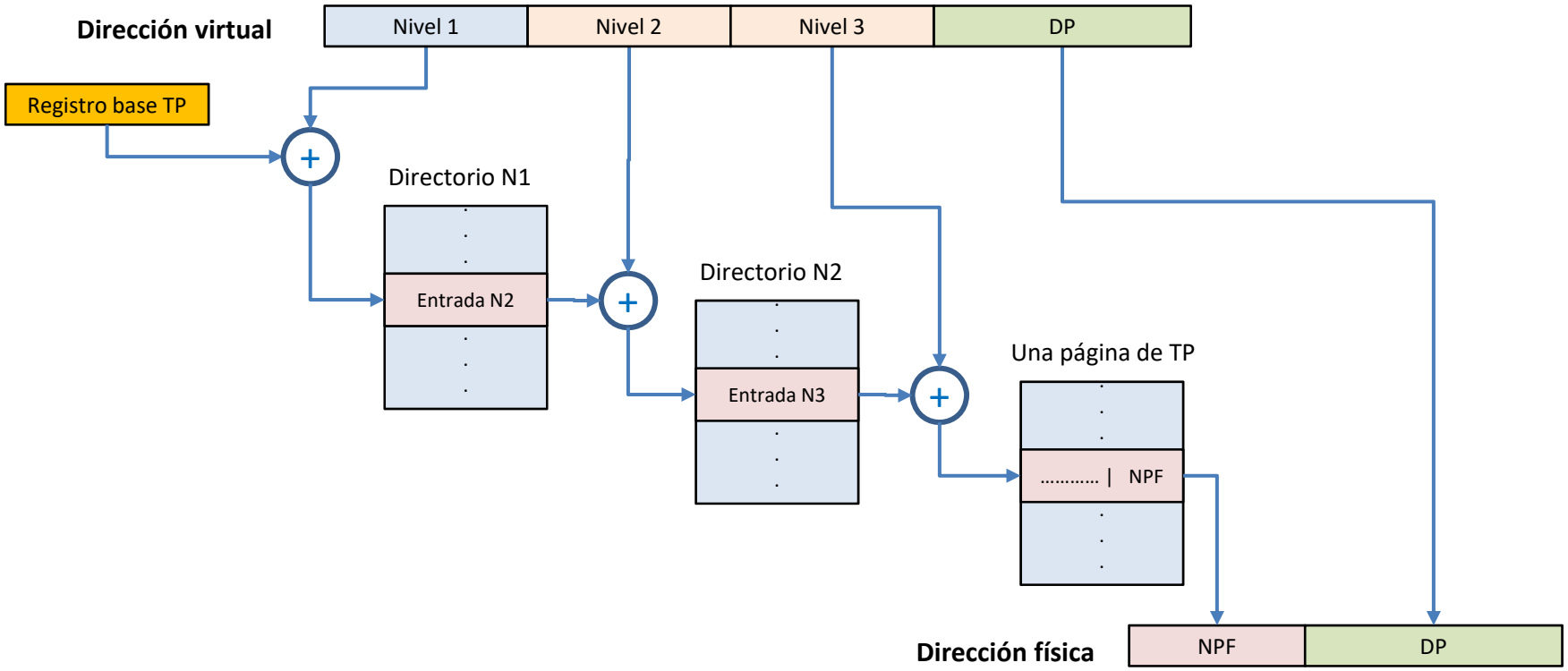
➢ Problemas con la tabla de páginas

- Sistema de memoria virtual con las siguientes características
  - Una dirección virtual de 32 bits
  - Un tamaño de página de 4 K = 2^{12} bytes
  - 4 bytes por entrada de página
- ¿Cuál es el tamaño de la tabla de páginas?
  - Número de páginas = Número de entradas de la tabla de páginas =2^{32} / 2^{12} = 2^{20}
  - Tamaño de la tabla de páginas = Nº de entradas x 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 varios niveles
  - Tabla de páginas invertida

3. Memoria virtual paginada

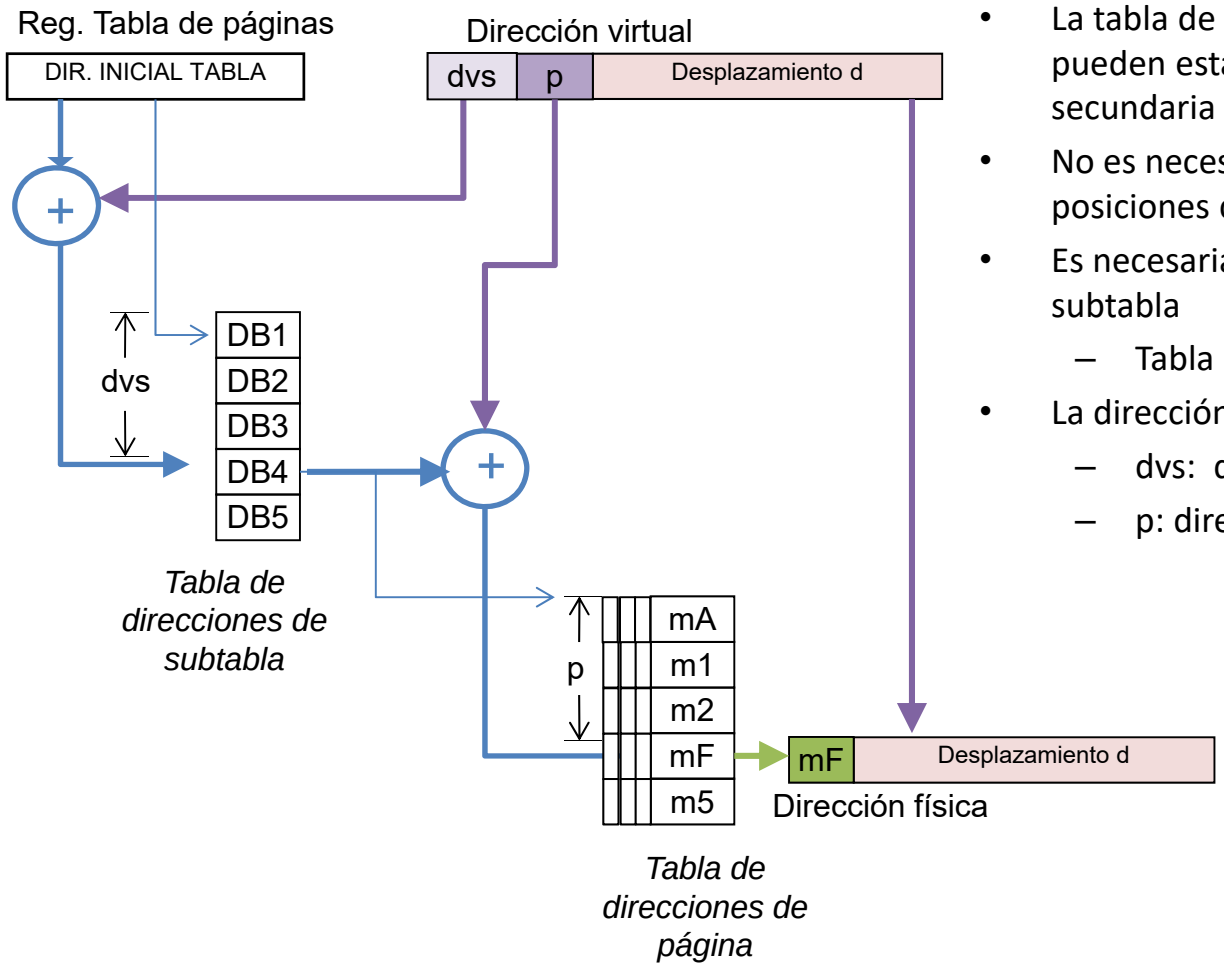
➢ Tabla de páginas de varios niveles

- Existe una página de directorio de nivel 1 en la que cada elemento apunta a una tabla de directorios de nivel 2, y así sucesivamente hasta llegar a las páginas de la TP.
- La longitud máxima de una tabla de páginas se restringe al tamaño de una página
- Ejemplo: TP organizada en tres niveles: N1, N2 y N3.



3. Memoria virtual paginada

➢ Ejemplo de tabla de páginas de dos niveles: Pentium

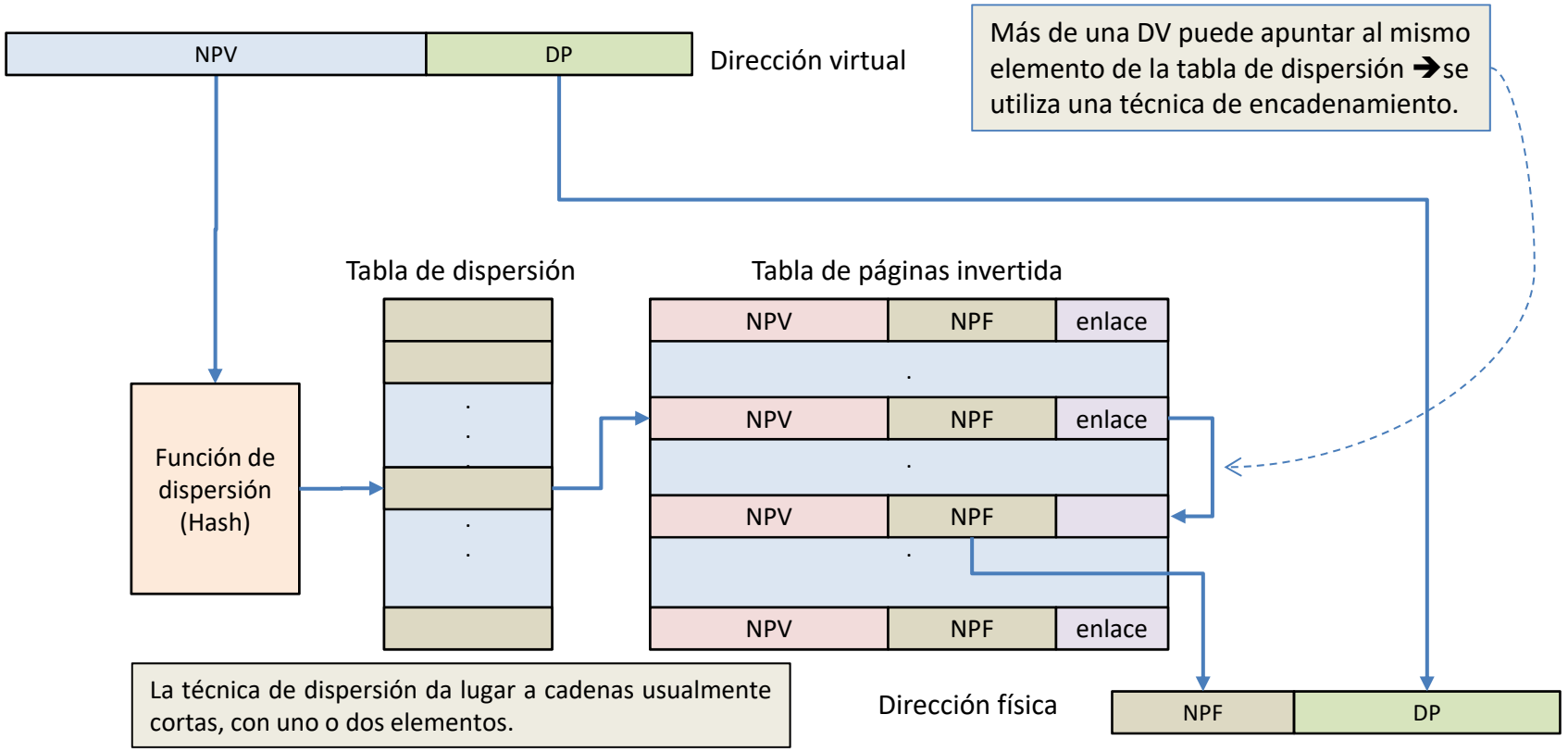


- La tabla de páginas se divide en subtablas que pueden estar en Memoria Principal o secundaria -> tabla virtual.
- No es necesario que las subtablas estén en posiciones consecutivas en memoria
- Es necesaria una dirección base para cada subtabla
  - Tabla de direcciones de subtabla
- La dirección de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

3. Memoria virtual paginada

➢ Tabla de páginas invertida (HASH)

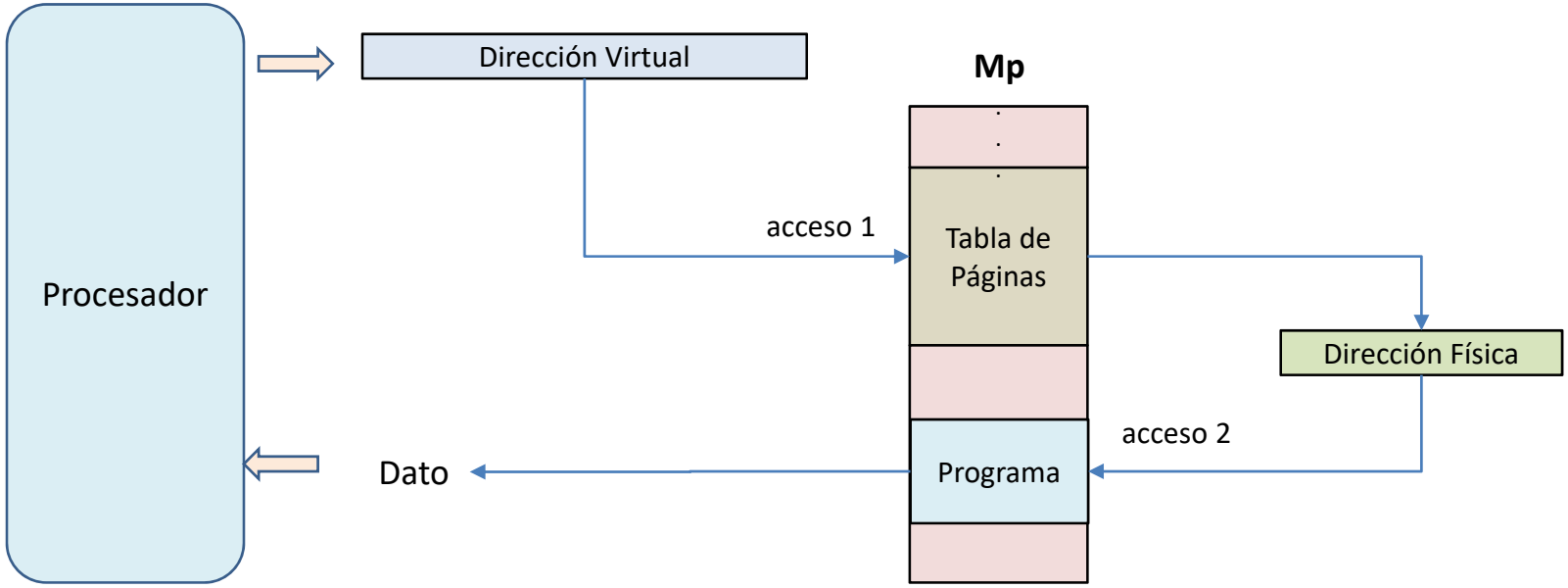
- Elimina de la TP las entradas que no apuntan a una página física en Mp: nº entradas = nº paginas físicas.
- Se aproxima a un comportamiento asociativo de la TP.
- El campo NPV de la DV se hace corresponder sobre una *tabla hash* con una función de dispersión.
- La tabla de dispersión incluye un puntero a una TP invertida, que contiene los elementos de la TP.
- Hay un elemento en la tabla de dispersión y en la tabla invertida para cada marco de página .
- Requiere una zona fija en MF para las tablas, con independencia del número de programas o páginas.
- Esta alternativa de TP se utiliza en el PowerPC.



3. Memoria virtual paginada

➢ Tiempo de acceso en memoria virtual paginada

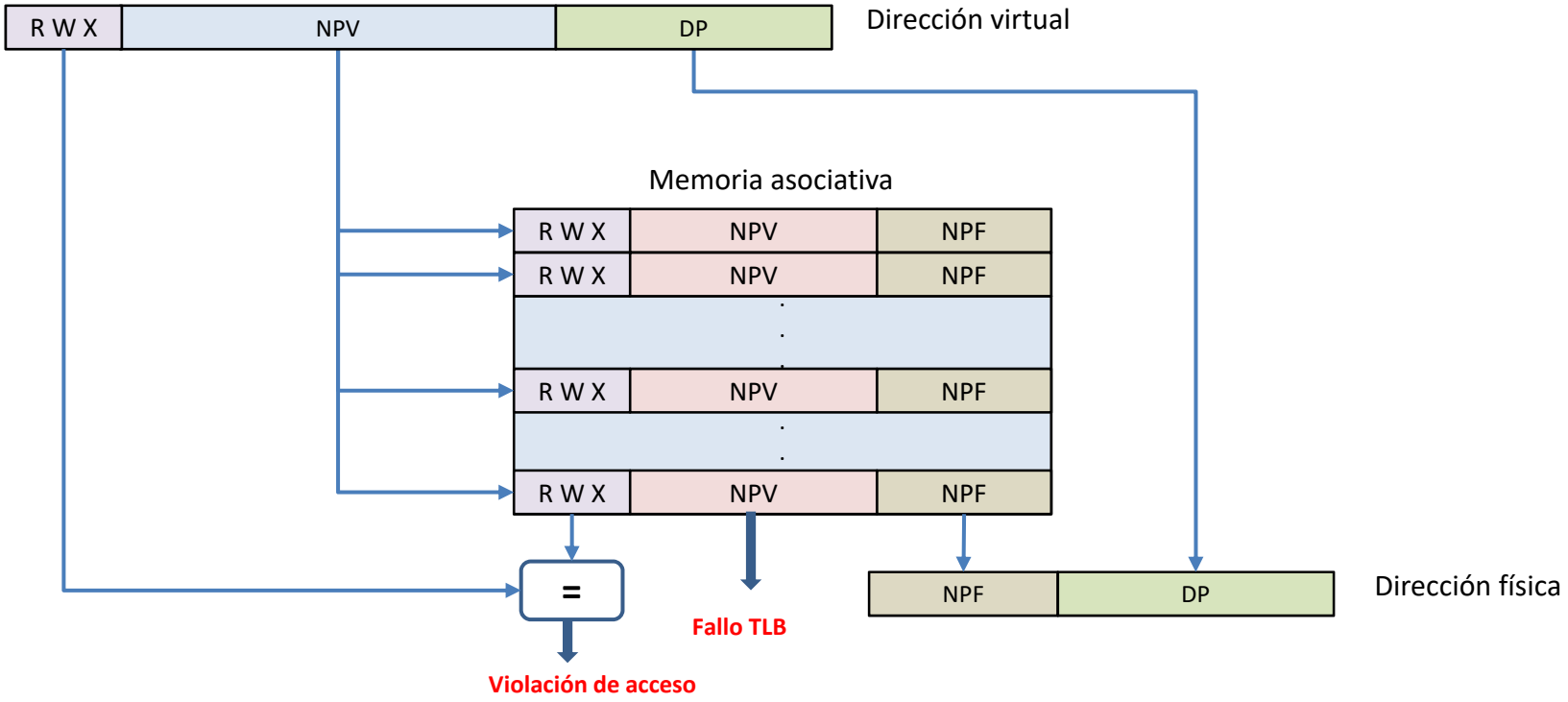
- Puesto que la TP se almacena total o parcialmente en Mp, para obtener un dato se necesitan dos accesos a memoria:
  - Un acceso para obtener la dirección física.
  - Otro acceso para obtener el dato.
- En tablas multinivel el tiempo de acceso es mayor: se necesita un acceso más por nivel de la tabla.



3. Memoria virtual paginada

➢ Buffer de traducción anticipada (TLB: *Translation Lookaside Buffer*)

- Se reduce el tiempo de acceso a Mv utilizando una cache asociativa para la TP denominada *buffer de traducción anticipada* (TLB).
- El TLB contiene aquellas entradas de la TP a las que se ha accedido recientemente.
- Por localidad temporal, la mayoría de las referencias a memoria corresponderán a posiciones incluidas en páginas recientemente utilizadas.
- TLB de pequeño tamaño ⇒ implementar con tecnología rápida ⇒ menor tiempo de acceso.
- Tamaño típico de la TLB: 32 - 4K entradas

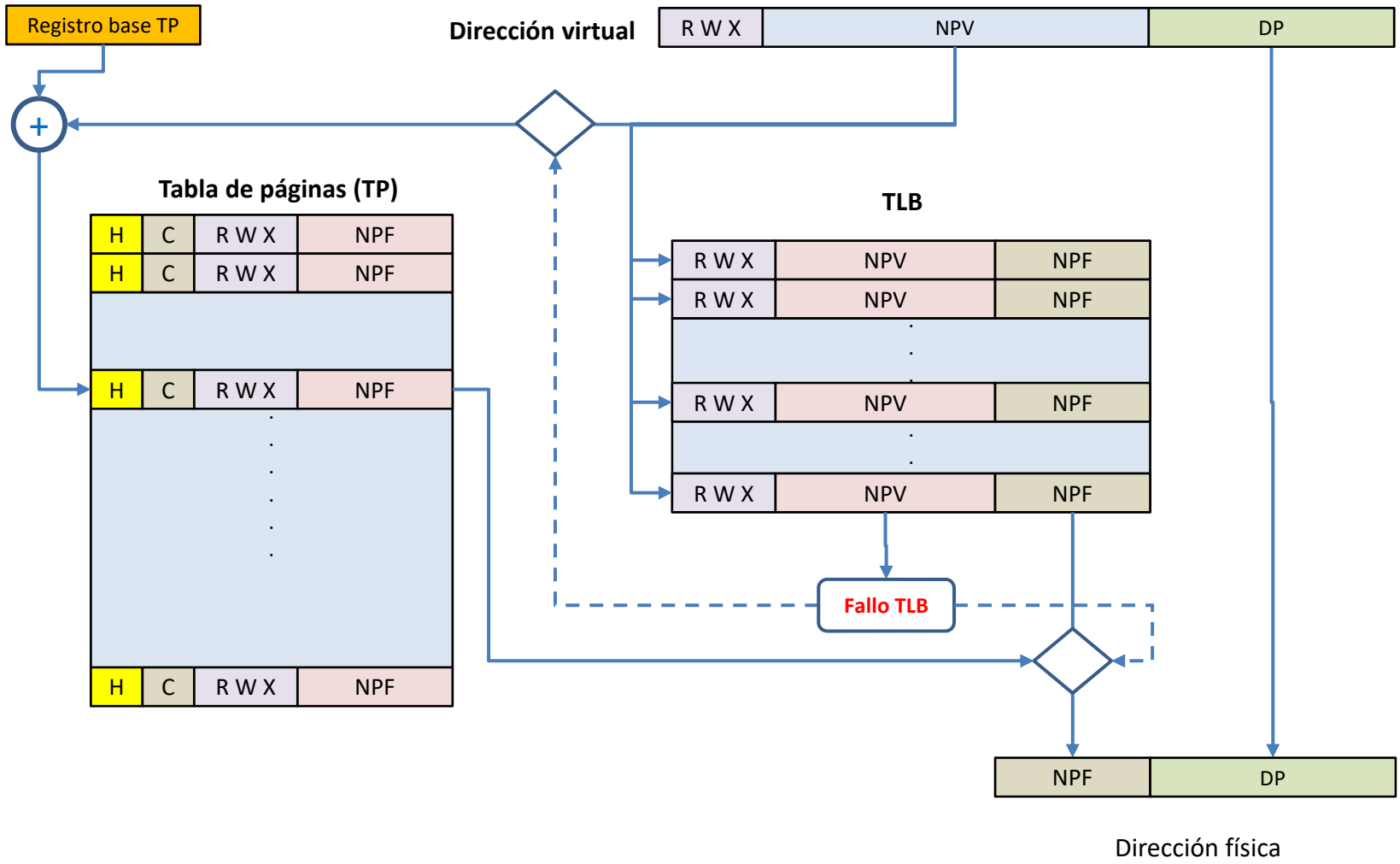


Buffer de Traducción Anticipada (TLB)

3. Memoria virtual paginada

➢ Interacción entre la TP y el TLB

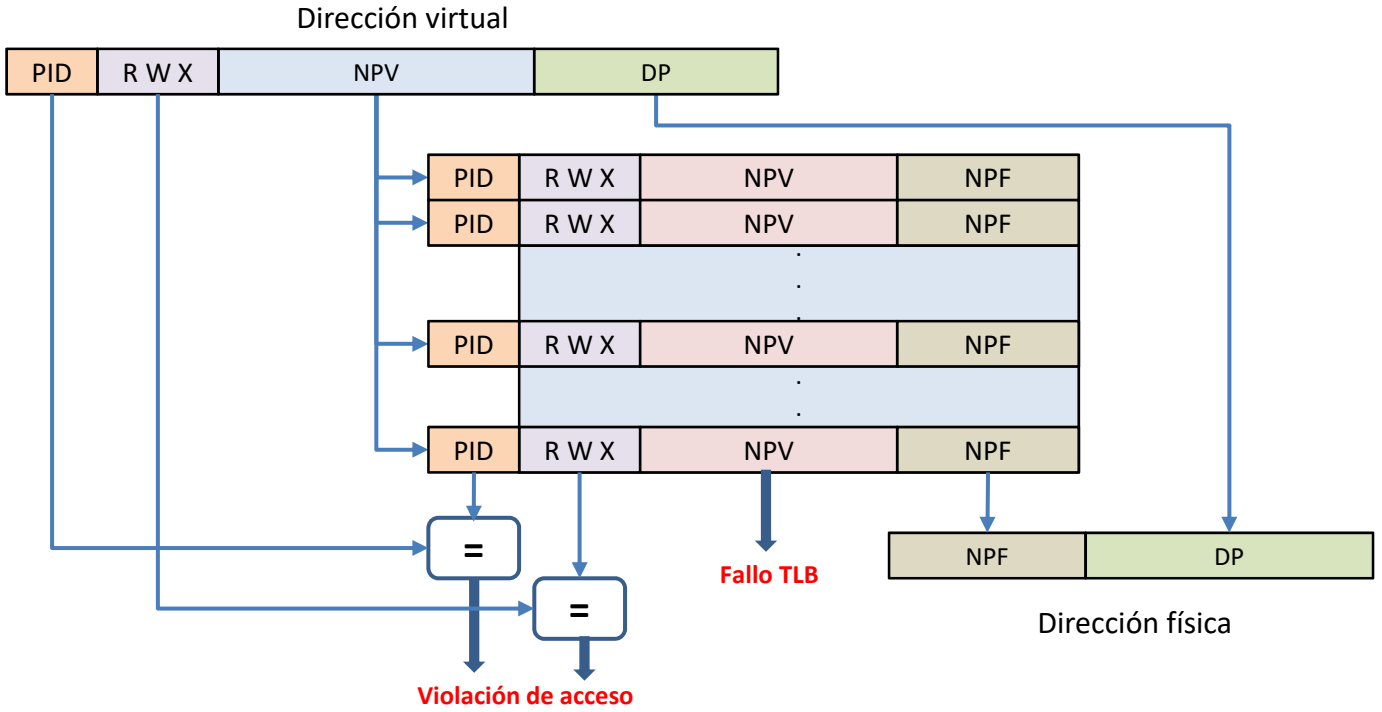
- Para traducir una dirección se accede en primer lugar al TLB
- Si la dirección no se encuentre en el TLB se accede a la TP y se actualiza el TLB



3. Memoria virtual paginada

➢ TLB e Identificador de proceso (PID)

- Existen dos alternativas en el diseño de una TLB : con identificador y sin identificador 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 (programa) se debe invalidar el TLB ya que cada proceso tiene su propio espacio virtual
- TLB con identificador 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 sistema operativo se encarga de asignar un identificador a cada proceso
  - En los cambios de proceso no es necesario invalidar todo el TLB.



3. Memoria virtual paginada

➢ Interacción entre la memoria virtual y la memoria cache

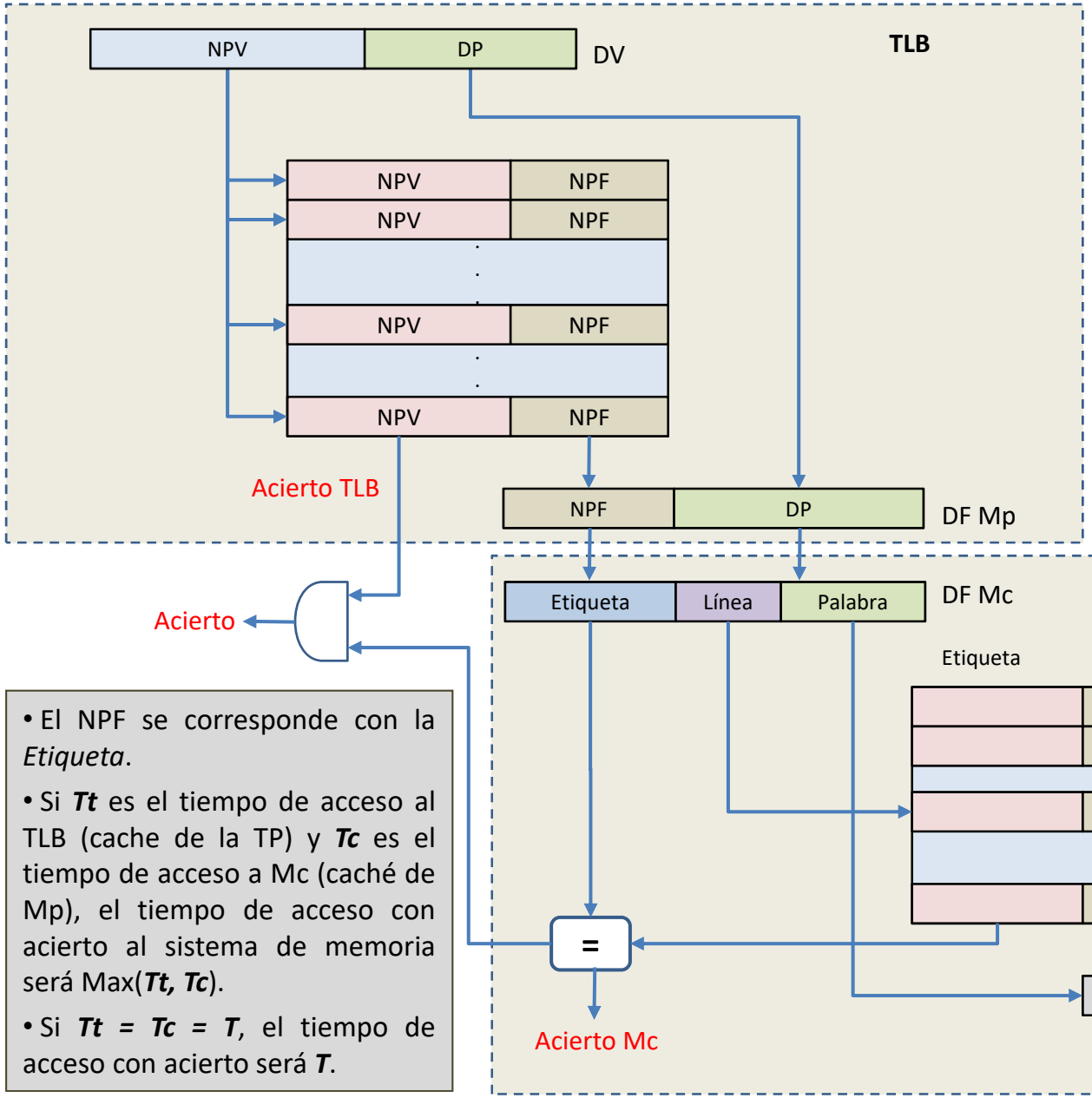
- La memoria virtual debe interactuar con la memoria caché
- Para traducir una DV se consulta el TLB para comprobar si la entrada se encuentra en él.
  - Si es así, se genera la DF combinando el NPF con el DP.
  - Si no, se accede a la TP en busca del elemento correspondiente.
- Con DF se consulta la caché para ver si el bloque se encuentra en el conjunto.
  - Si es así, se envía al procesador.
  - Si no, se produce un fallo de cache y se busca la palabra en memoria principal.
- La DV debe pasar primero por el TLB antes de que la DF pueda acceder a la cache.
- Este mecanismo se puede acelerar utilizando dos alternativas:
  - Acceder simultáneamente al TLB para buscar el NPF y a la cache para buscar el bloque.
  - Utilizar caches con direcciones virtuales
- La primera alternativa requiere que la longitud del campo de desplazamiento DP de la DV (igual al DP de la DF) sea mayor o igual que los campos de *conjunto* y *palabra* del formato de DF de la cache

  De esta forma será posible realizar en paralelo

  1) La búsqueda en el TLB del NPF que se corresponde con la etiqueta.
  2) La búsqueda en el directorio de la cache del nº de bloque (línea).

3. Memoria virtual paginada

➢ Acceso paralelo al TLB y al directorio de la cache



• Si la longitud del campo de desplazamiento DP de la DV (igual al DP de la DF) es mayor o igual que los campos de *Línea* y *Palabra* del formato de la DF para la memoria cache Mc, tal como se muestra en la figura, se puede acceder en paralelo al TLB para buscar el NPF y al directorio de la cache para buscar el bloque,

• El NPF se corresponde con la *Etiqueta*.

• Si ***Tt*** es el tiempo de acceso al TLB (cache de la TP) y ***Tc*** es el tiempo de acceso a Mc (caché de Mp), el tiempo de acceso con acierto al sistema de memoria será Max(***Tt, Tc***).

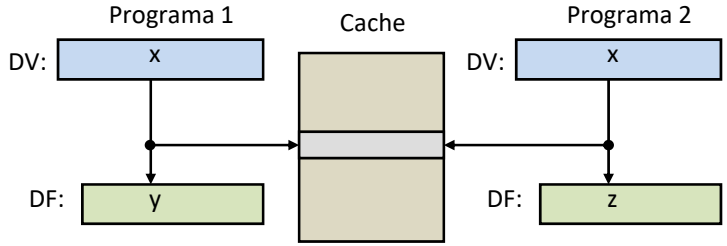
• Si ***Tt = Tc = T***, el tiempo de acceso con acierto será ***T***.

3. Memoria virtual paginada

➢ Acceso a caché con direcciones virtuales

- Idea básica
  - Permite acceder a la cache usando directamente la DV
  - Acelera el acceso a la cache eliminando el tiempo de traducción DV → DF

- Problema
  - Colisiones
    - Cada programa tiene su propio espacio de direcciones virtuales
    - Dos programas pueden generar DV idénticas que se refieren a DF distintas
    - Sin embargo, ambas DV accederán a la misma posición de cache

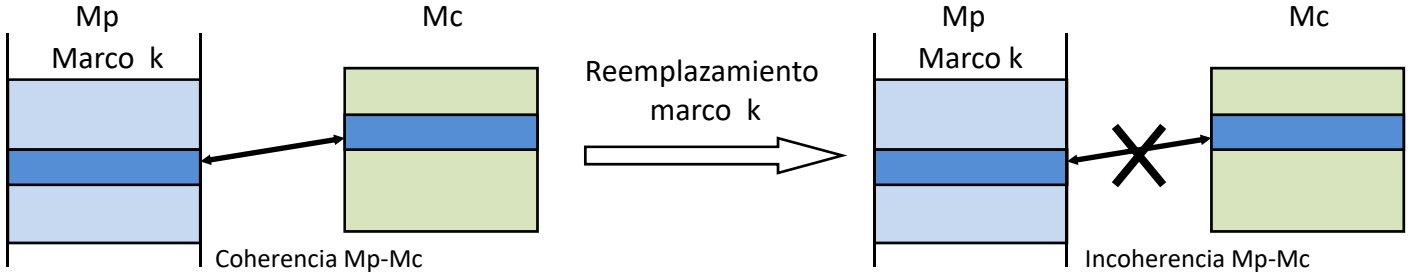


- **Solución**
  - Añadir en la etiqueta de la cache un identificador del programa (proceso) al que pertenece ese bloque

3. Memoria virtual paginada

➢ Mantenimiento de la coherencia entre Mc y Mp

- Cuestiones planteadas
  - ¿Qué ocurre cuando se reemplaza una página en Mp y algún bloque de esa página estaba en Mc?
  - ¿Qué ocurre cuando el DMA ordena escribir sobre un bloque de Mp que está en Mc?
- En ambos casos se produce un problema de incoherencia entre Mp y Mc
  - El contenido de Mp ha variado y por tanto Mp y Mc no contienen la misma información



- Solución: Invalidar todos los bloques de cache que se ven afectados por esa modificación en Mp

➢ DMA y memoria virtual

- ¿Un controlador DMA debe utilizar direcciones virtuales o físicas?
- Solución: debe emplear direcciones virtuales
  - El DMA controla las transferencias de un bloque hacia/desde *posiciones consecutivas de memoria*
  - Para ello emplea registro de direcciones que se incrementa/decrementa después de cada transferencia
  - Si el DMA usa direcciones físicas y el bloque a transferir no cabe en una sola página sería necesario acceder a dos páginas distintas que no tienen por qué estar ubicadas en posiciones consecutivas de memoria

3. Memoria virtual paginada

➢ Políticas de búsqueda (*fetch*)

- **Prebúsqueda** (análoga a la utilizada en memoria cache)
- **Búsqueda por demanda** (análoga a la utilizada en memoria cache)

➢ Políticas de sustitución (*replacement*)

- **Aleatoria:** elije una página aleatoriamente, sin mirar el número de referencias o el tiempo que lleva en Mp.
- **LRU (Least Recently Used)**: análoga a la utilizada en memoria cache.
- **FIFO (First In First Out)**: se sustituye la página que lleva más tiempo residente en memoria.
- **Reloj (FINUFO: First In Not Used First Out)**: es una mejora de la FIFO en la que también se chequea si una página ha sido referenciada → hace mejor uso de la localidad temporal.
- **Optima (MIN)**: es la mejor política posible: sustituir la página que vaya a tardar más tiempo en ser referenciada. No es implementable.
  - Sólo se utiliza como referencia teórica para medir la eficiencia de otras políticas.

3. Memoria virtual paginada

➢ Política LRU: menos recientemente usada

- Se sustituye la página que lleva más tiempo sin ser referenciada.
- Produce buenos resultados puesto que por la localidad temporal, páginas recientemente referenciadas es probable que sean próximamente referenciadas.
- Implementación: se asocian contadores a las páginas que se actualizan de la siguiente forma cuando se referencia la página P_i:

  $\forall P_k$: si contador(P_k) $\leq$ contador(P_i) ==> contador(P_k) := contado(P_k) + 1
  contador(P_i) = 0
  Cuando se produce un fallo se sustituye la página P_j : contador(P_j) = MAXIMO
- Ejemplo:

Referencias	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
		0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
Fallos	f	f	f	F		F		F	F	F	F			F		F		F		

Total: 12 fallos de página (3 forzosos)

3. Memoria virtual paginada

➢ Política FIFO

- Se sustituye la página residente que lleve más tiempo en memoria.
- Fácil de implementar con una cola FIFO de páginas.
- Problema: al no tener en cuenta la historia del uso de una página, puede prescindir de páginas a las que se accede con frecuencia.
- Padece la anomalía de **Belady: fenómeno paradójico** consistente en que aumenta el número de fallos de páginas cuando aumentan los marcos de páginas en Mp.
- Ejemplo:

Referencias	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
		0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
			1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
Fallos	f	f	f	F		F	F	F	F	F	F			F	F			F	F	F

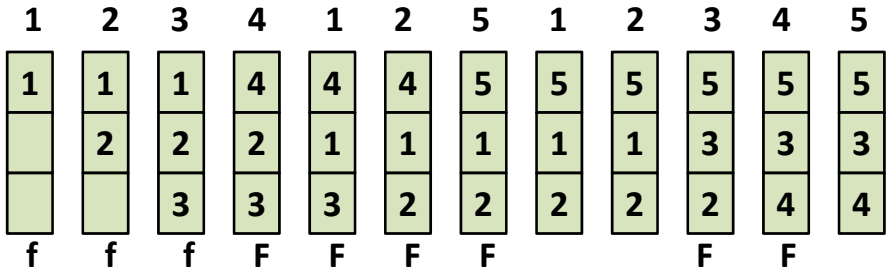
Total: 15 fallos de página (3 forzosos)

3. Memoria virtual paginada

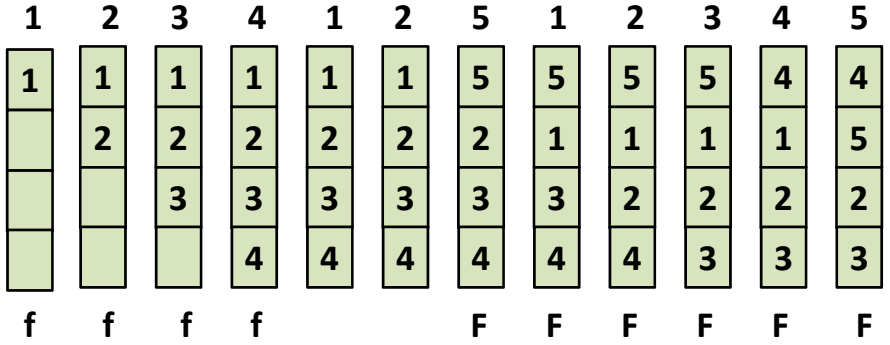
➢ Anomalía de Belady:

- Consiste en la posibilidad de que aumentando el número de marcos de página puede aumentar el número de fallos.

- Por ejemplo, las siguientes referencias a las páginas:
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
  dan lugar a 9 fallos con 3 marcos y a 10 fallos con 4 marcos



Total: 9 fallos

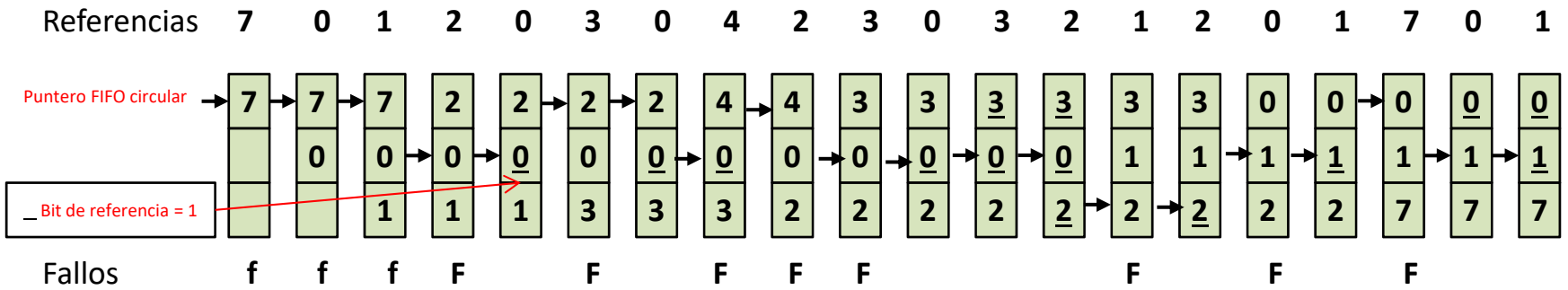


Total: 10 fallos

3. Memoria virtual paginada

➢ Política del reloj o FINUFO

- Política FIFO que además tiene en cuenta la última referencia.
- Se mantiene una cola como en la FIFO pero **circular**.
- Se introduce un *bit de referencia* asociado a cada página para anotar que ha sido referenciada.
- Cuando hay que sustituir una página se aplica la política FIFO con la siguiente modificación:
  - Si el valor del bit de referencia de la página seleccionada es 0, se sustituye la correspondiente página, como en la política FIFO pura.
  - Si el valor del bit de referencia de la página seleccionada es 1, la página no es sustituida, se pone su bit de referencia a 0 y se aplica la política FIFO
- Ejemplo:



Total: 11 fallos de página (3 forzosos)

3. Memoria virtual paginada

➢ Política óptima

- Se sustituye la página que más tiempo tarde en volver a ser referenciada
- Presenta la frecuencia de fallos de página más baja de todos
- No es implementable (requiere adivinar el futuro))
- Útil como referencia de comparación para otras políticas
- Ejemplo:

Referencias	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
			1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
Fallos	f	f	f	F		F		F			F			F				F		

Total: 9 fallos de página (3 forzosos)

4. Memoria virtual segmentada

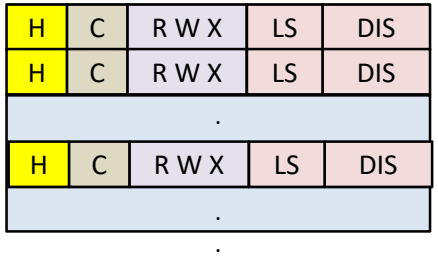
➢ Concepto de memoria virtual segmentada

- Proporciona una forma lógica de organizar los programas y datos.
- Permite gestionar la memoria como múltiples espacios de direcciones o segmentos.
- Los segmentos tienen un tamaño variable, dinámico.
- Se puede asignar a cada segmento derechos de acceso y uso (privilegios y protección).
- Las DVs se componen de Número de Segmento (NS) y Desplazamiento en el Segmento (DS).

Dirección virtual

- La traducción de DV a DF se realiza con la ayuda de la Tabla de Segmentos (TS) que contiene una entrada por segmento.

Tabla de segmentos (TS)

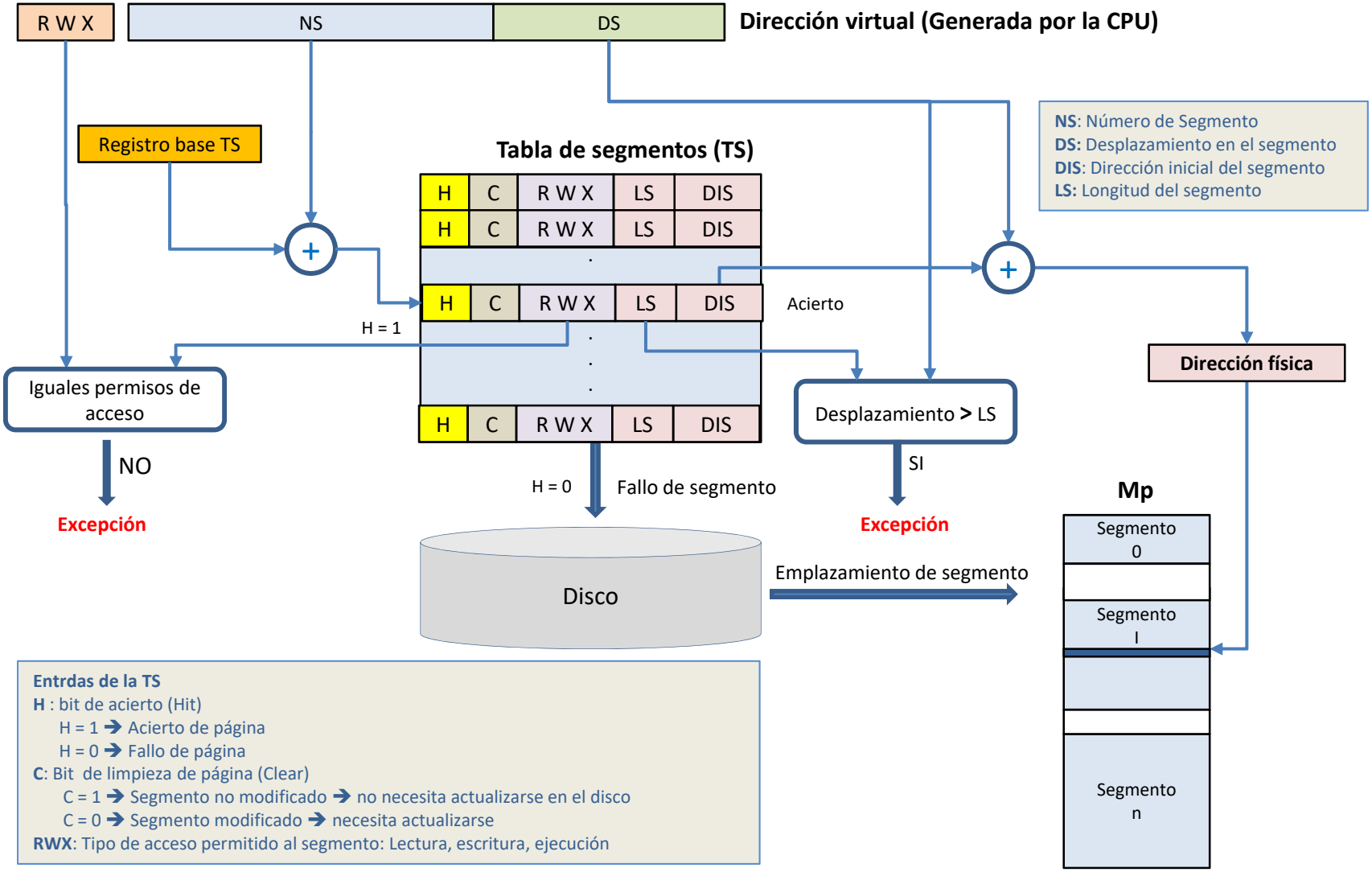


- Cada entrada de la TS contiene:
  - Los mismos bits de control y protección que la Tabla de Páginas.
  - La Longitud del Segmento (LS)
  - La Dirección Inicial del Segmento (DIS)

4. Memoria virtual segmentada

➢ Estructura funcional de la memoria virtual segmentada

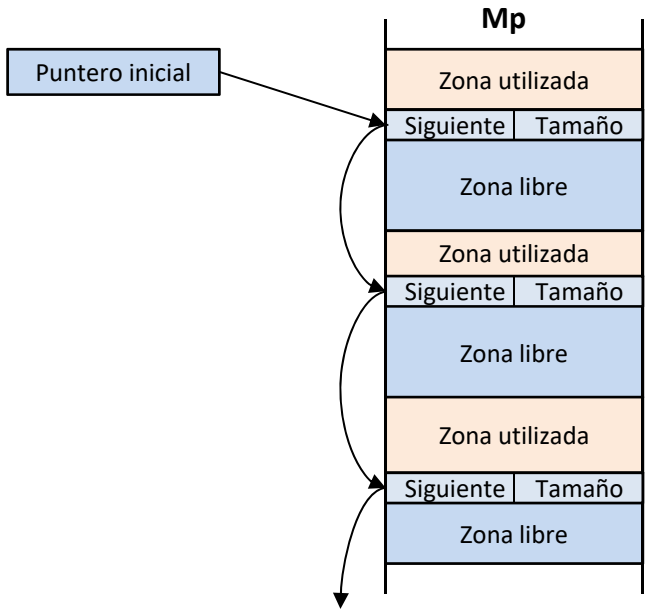
- Mecanismo de traducción de Dirección virtual a Dirección física



4. Memoria virtual segmentada

➢ Políticas de ubicación (*placement*) para memorias segmentadas (1)

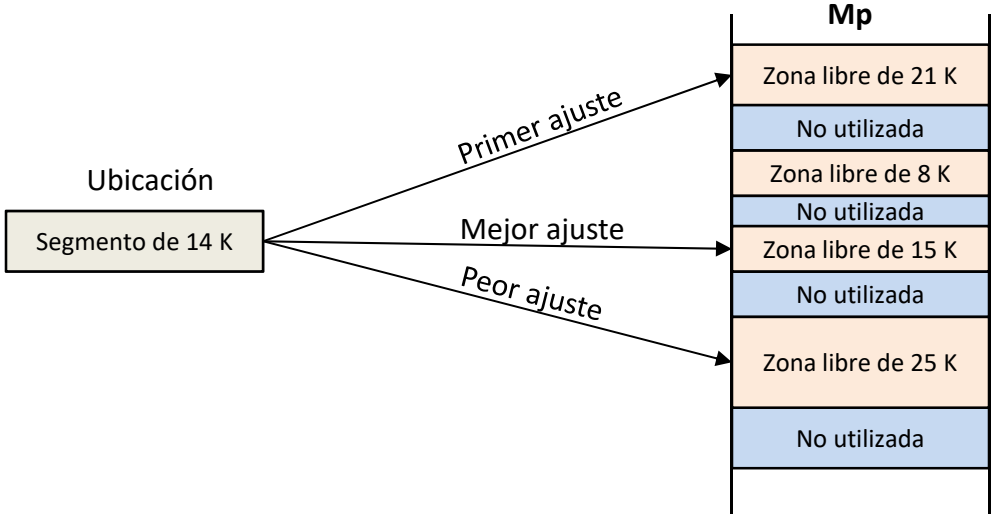
- Cuando se produce un fallo de segmento hay que buscar en Mp una ubicación adecuada.
- El problema no es trivial porque tanto los segmentos como los huecos en Mp son de tamaño variable.
- Para gestionar las zonas libres de Mp:
  1. Se crea una lista enlazada con identificación del tamaño de cada hueco.
  2. Se aplica la política de ubicación sobre la lista.
  3. Determinado el hueco donde ubicar el segmento, se actualiza la lista.
  4. Si no se encuentra un hueco apropiado interviene la política de sustitución.



4. Memoria virtual segmentada

➢ Políticas de ubicación (*placement*) para memorias segmentadas (2)

- Las tres políticas de ubicación de segmentos más utilizadas son las siguientes:
  - **Mejor ajuste (*best fit*)**
    - La lista de huecos se mantiene ordenada en orden creciente de tamaño
    - Se ubica el segmento en el primer hueco con capacidad suficiente para albergarlo.
  - **Peor ajuste (*worst fit*)**
    - La lista de huecos se mantiene ordenada en orden decreciente de tamaño
    - Se ubica el segmento en el primer hueco con capacidad suficiente para albergarlo.
  - **Primer ajuste (*first fit*)**
    - La lista de huecos se ordena según las direcciones iniciales de los huecos
    - Se ubica el segmento en el primer hueco con capacidad suficiente para albergarlo.
    - Cuando transcurre un cierto tiempo, se acumulan un número elevado de huecos pequeños próximos a la cabeza de la lista, penalizando las búsquedas.
    - Se evita adelantando cíclicamente, después de cada búsqueda, la posición inicial de la lista.



5. Memoria con segmentos paginados

➢ Justificación

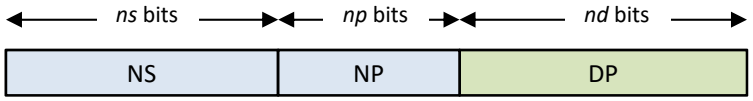
• Al ser los segmentos unidades lógicas completas de programa o datos, la memoria virtual segmentada presenta ventajas para el programador, pero una mala gestión de Mp.

• La paginación por el contrario proporciona una forma más eficiente de gestionar la Mp, pero resulta más alejada de los intereses del programador.

• La memoria virtual con segmentos paginados (segmentada paginada) combina las ventajas de ambas: unidades lógicas para el programador y eficiencia en la gestión de Mp.

➢ Espacio virtual de direcciones

• Se divide en segmentos de tamaño variable.
• Cada segmento está compuesto por un número variable de páginas de igual tamaño.
• **Dirección virtual**:

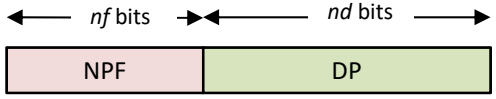


- Número máximo de segmentos (por programa)= 2^{ns}
- Tamaño máximo de un segmento = 2^{np} páginas
- Tamaño de una página = 2^{nd} palabras

• Los campos NS, NP y DP son de longitud fija pero un segmento puede tener una longitud variable entre 0 y 2^{np} páginas.

➢ Espacio físico de direcciones

• Se divide en marcos de página (páginas físicas) del mismo tamaño que la página.
• **Dirección física:**

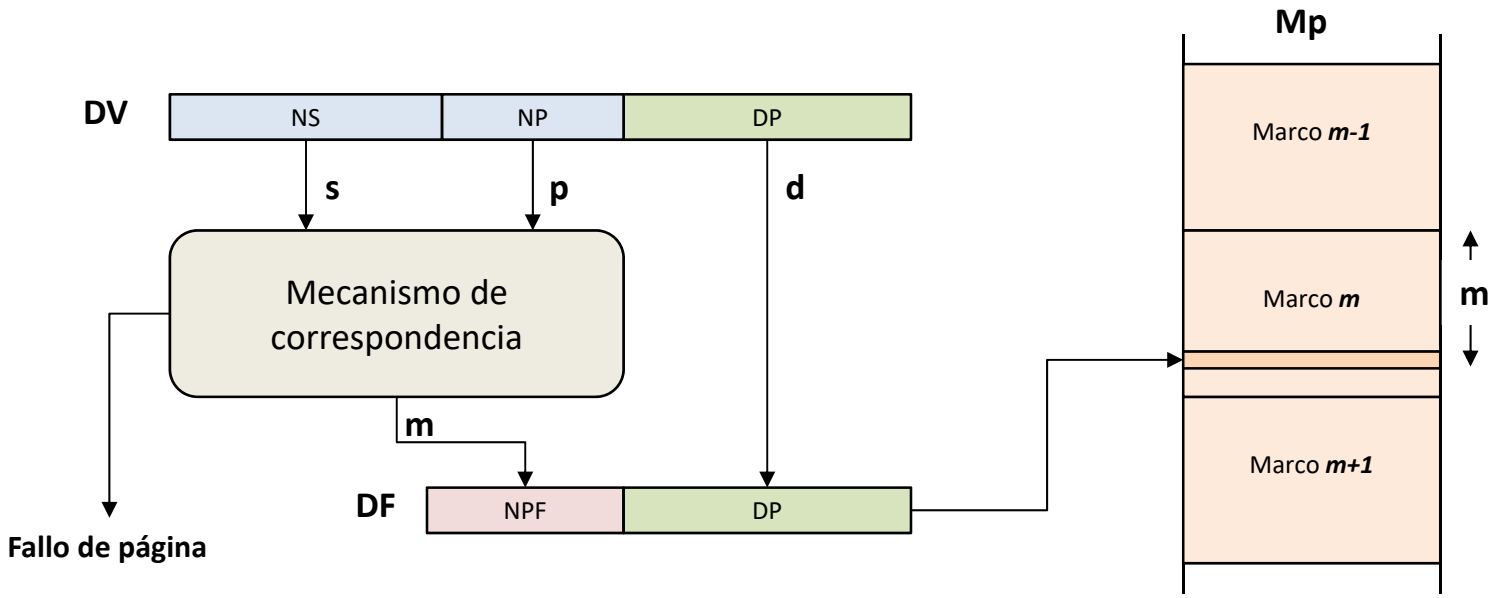


- Tamaño de un marco de página = 2^{nd} palabras
- Número total de marcos de página en Mp = 2^{nf}

5. Memoria con segmentos paginados

➢ Traducción de direcciones

• Se necesita un mecanismo de correspondencia para determinar en qué página física (marco de página) ***m*** de **Mp** se ubica la página ***d*** del segmento ***s***.



• El mecanismo de correspondencia determina para cada dirección virtual ***DV*** de un programa en ejecución ***P*** la dirección física ***DF*** si esta se encuentra en **Mp**, o fallo de página en caso contrario.

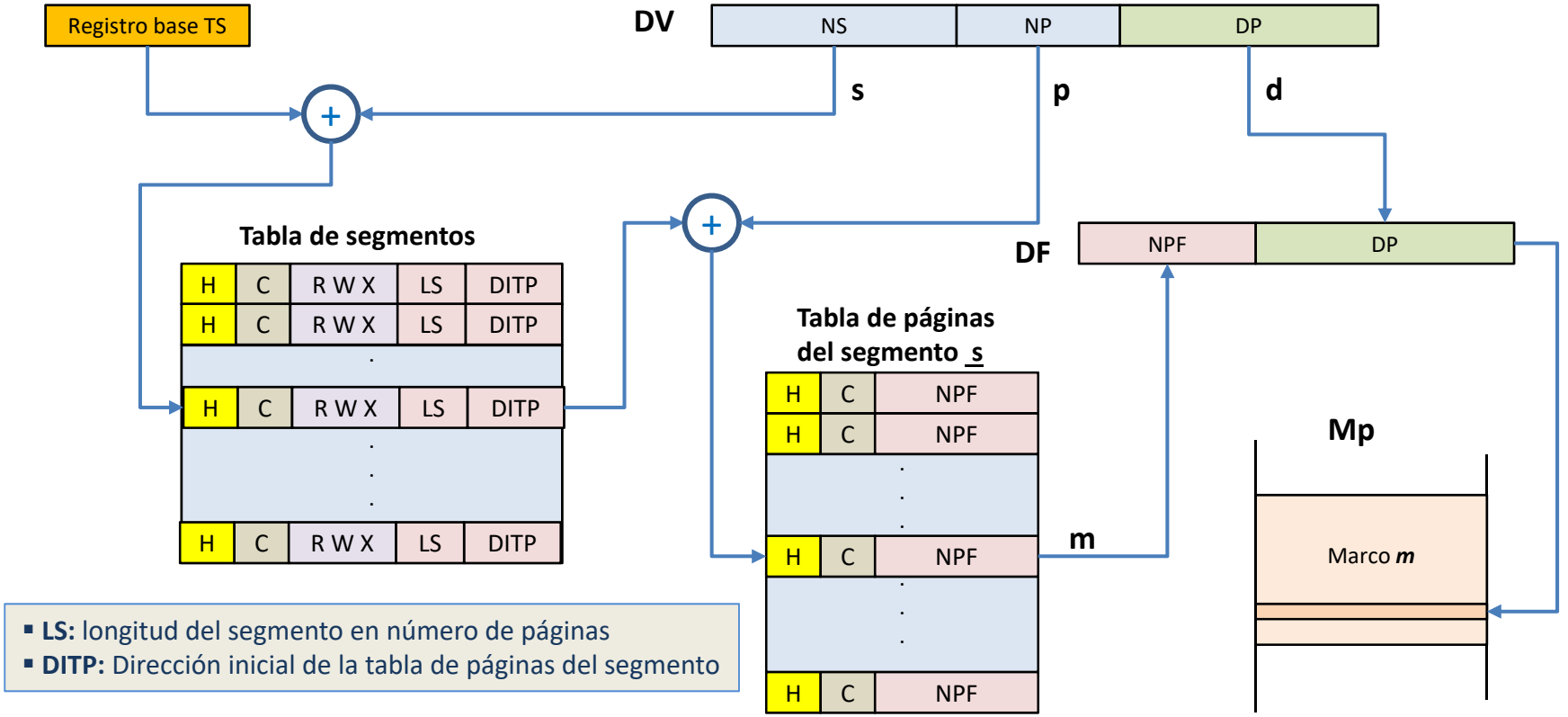
• Se suelen utilizar dos mecanismos de traducción:

- Traducción directa
- Traducción mixta

5. Memoria con segmentos paginados

➢ Traducción directa

- Existe una tabla de segmentos (TS) para cada programa P.
- Cada entrada de TS almacena la dirección inicial de la tabla de páginas de ese segmento.
- La tabla de páginas de un segmento contiene una entrada por cada página de dicho segmento.
- Las páginas de un segmento no tienen por qué almacenarse de forma consecutiva en Mp.

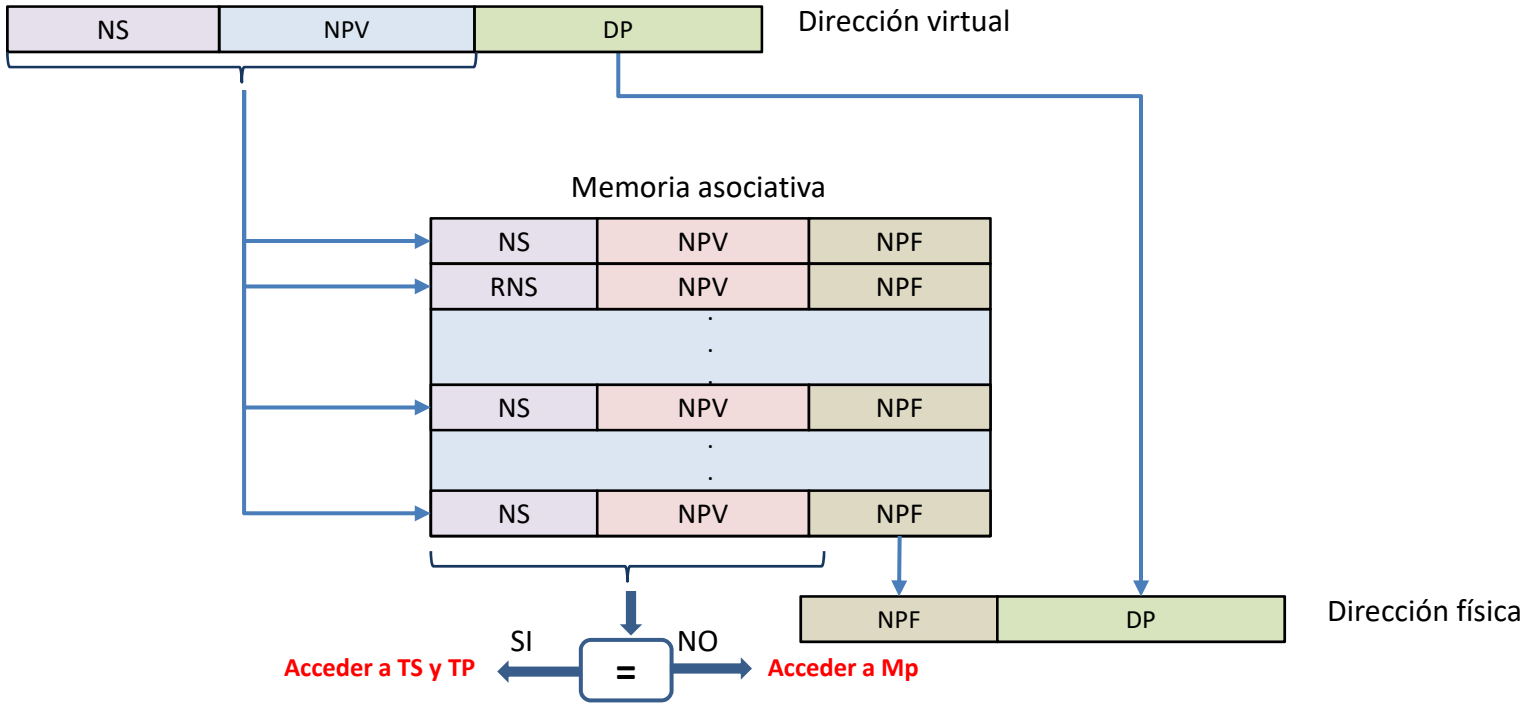


- **LS:** longitud del segmento en número de páginas
- **DITP:** Dirección inicial de la tabla de páginas del segmento

5. Memoria con segmentos paginados

➢ Traducción mixta

- Se utiliza un TLB con las páginas más activas
- Se debe guardar la TS en Mp y las TPs de los distintos segmentos en Mp o disco
- El proceso de traducción es el siguiente:



6. Memoria virtual del Pentium II

➢ **El Pentium II dispone de un sistema de gestión de memoria virtual con posibilidad de segmentación y paginación:**

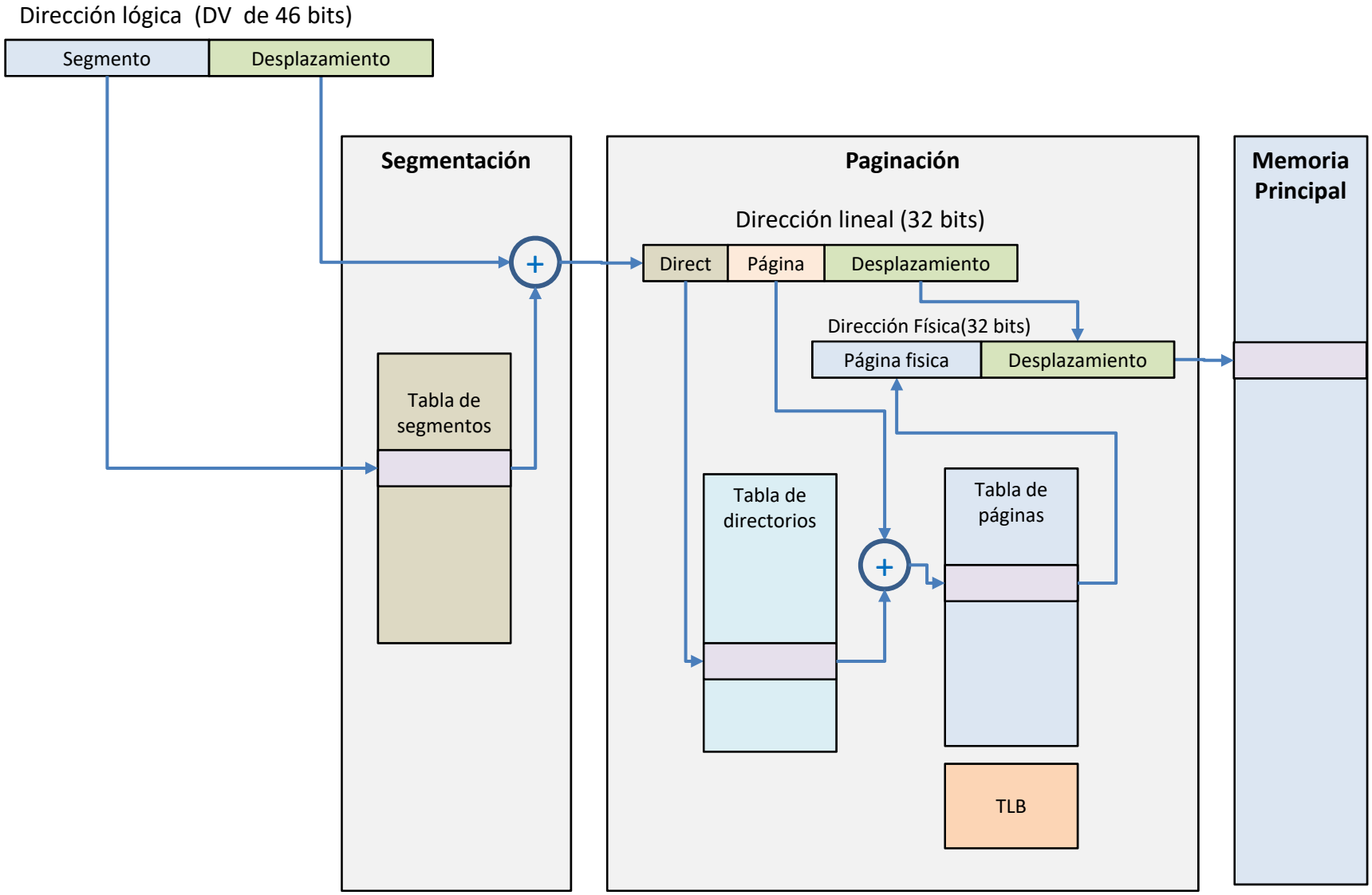
- **Unidad de segmentación (US)**
  - Utiliza DV de 46 bits (64 TB direccionables)
  - Genera una *Dirección Lineal (DL)* de 32 bits
  - Traducción de direcciones mediante TS
  - Tamaño del segmento: de 1 byte a 4 GB
- **Unidad de paginación (UP)**
  - Utiliza DV de 32 bits (4GB direccionables)
  - Genera una DF 32 bits
  - Traducción de direcciones a dos niveles mixta
    - Tabla de Directorios + TPs (una por directorio)
    - TLB de 32 entradas con las pág. más activas
  - Tamaño de la página: 4 KB o 4 MB

• **Activación**: los dos mecanismos se pueden activar o desactivar con independencia, dando pues lugar a cuatro formas de funcionamiento del sistema de memoria:

- Memoria no segmentada no paginada
- Memoria paginada no segmentada
- Memoria segmentada no paginada
- Memoria segmentada paginada

6. Memoria virtual del Pentium II

➢ Esquema funcional



6. Memoria virtual del Pentium II

➢ Modos de funcionamiento:

- **Memoria no segmentada no paginada**
  - La dirección virtual coincide con la dirección física.
  - Esta alternativa resulta útil cuando el procesador se utiliza como controlador de sistemas empotrados.

- **Memoria paginada no segmentada**
  - La memoria constituye un espacio lineal de direcciones paginado.
  - La protección y la gestión de memoria se realiza a través de la paginación.

- **Memoria segmentada no paginada**
  - La memoria constituye un conjunto de espacios de direcciones virtuales (lógicas)
  - La ventaja frente a la paginación es que proporciona protección a nivel de byte.

- **Memoria segmentada paginada:** se utilizan simultáneamente los dos mecanismos:
  - Segmentación para definir particiones lógicas de memoria en el control de acceso
  - Paginación para gestionar la asignación de memoria dentro de las particiones.