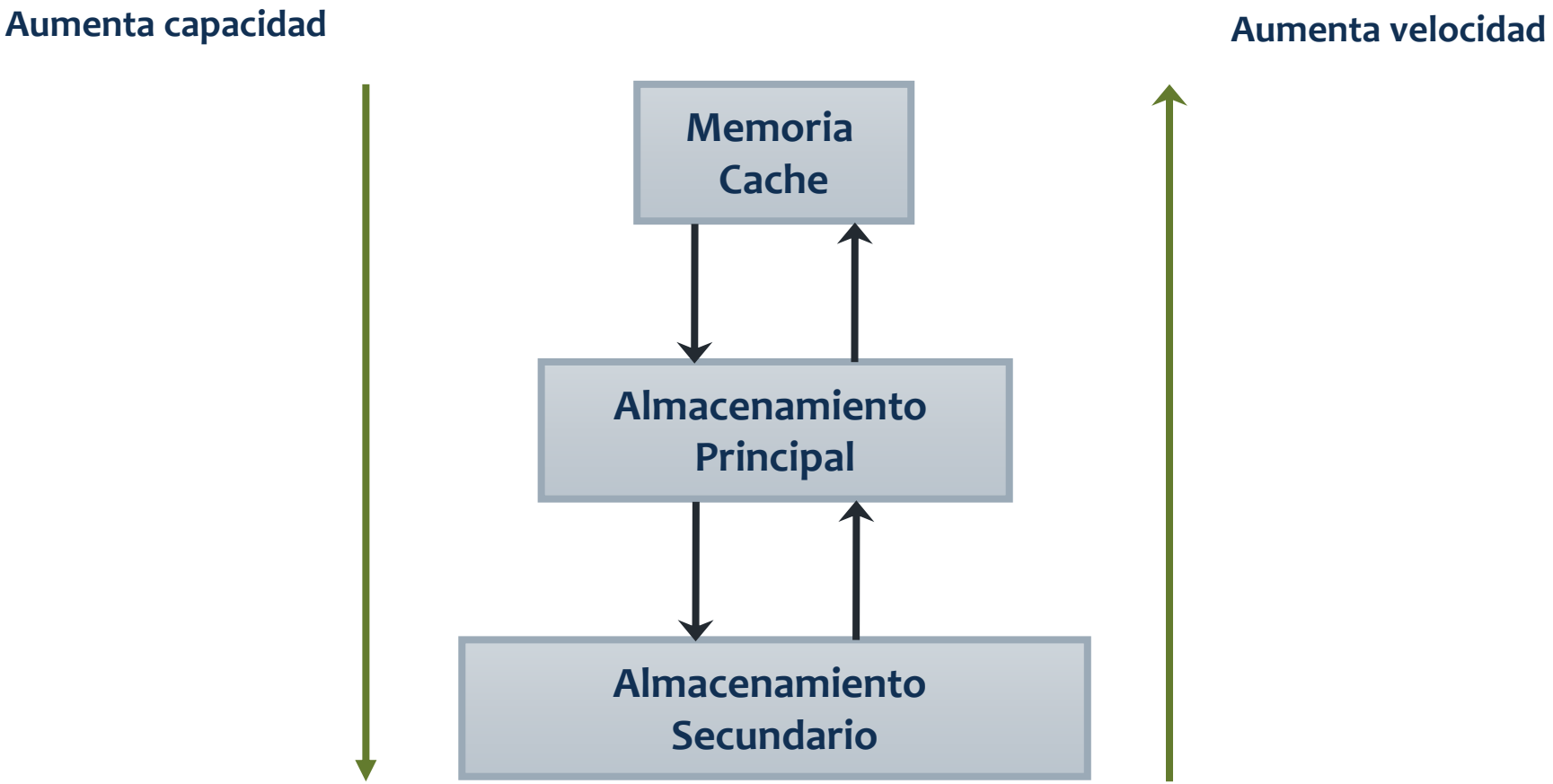
Tema III

Administración de Memoria

Jerarquía de las unidades de memoria



Asociación de direcciones

La asignación de la ubicación de un programa en memoria principal puede ser realizada en varios tiempos:

- **Tiempo compilación** (***compile time***): El programa será asignado a un lugar específico y conocido de la memoria física. La direcciones de memoria son referenciadas en forma absoluta (***static relocation***).
- **Tiempo de carga** (***load time***): La asignación del lugar de memoria donde será cargado el programa es hecho al momento de la carga. Las direcciones de memoria deben ser referenciadas en forma relativa (***dynamic relocation***).
- **Tiempo de ejecución** (***execution time***): Un programa puede variar su ubicación en memoria física en el transcurso de la ejecución.

Tipos de direccionamiento

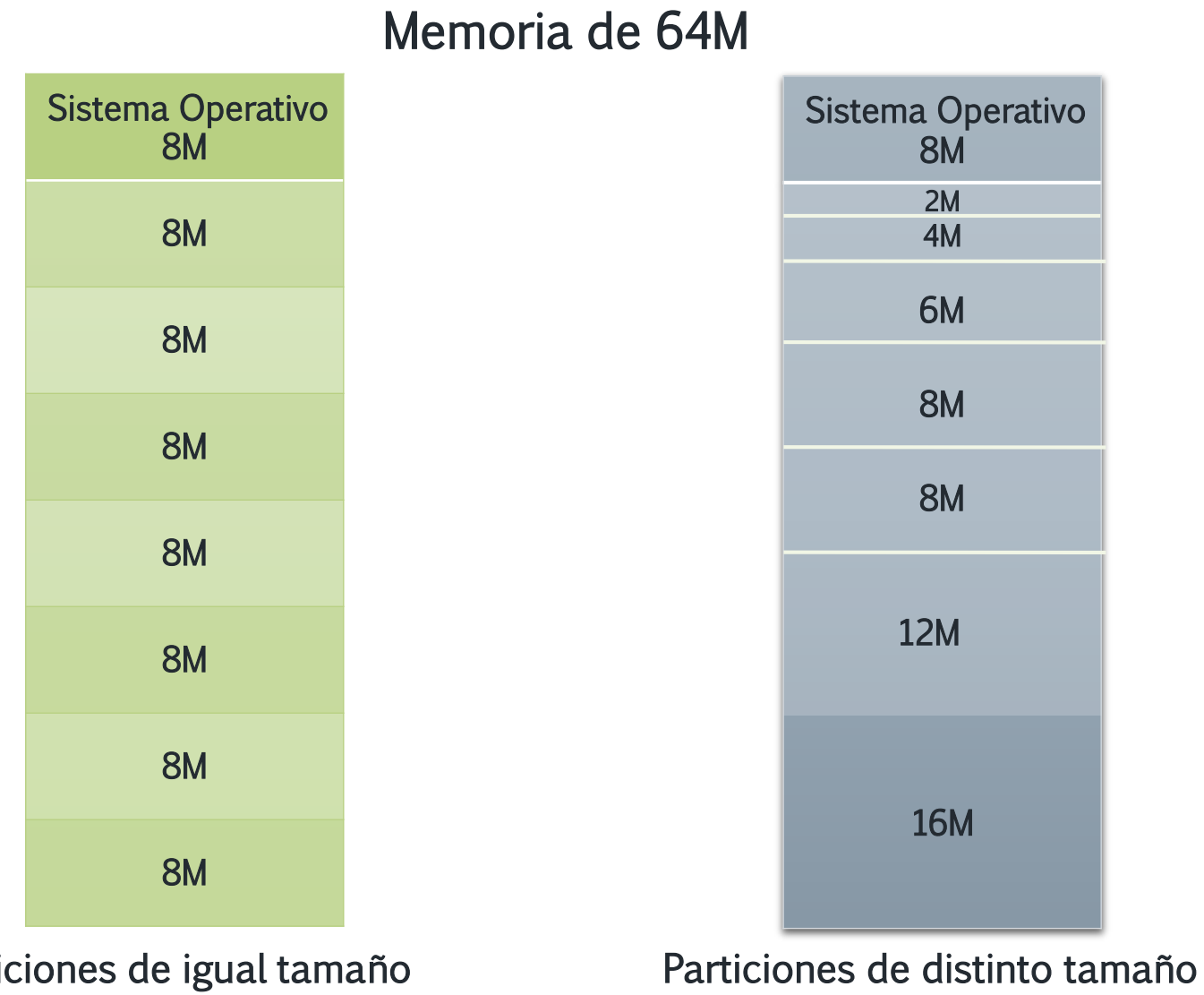
- **Direccionamiento físico** (***physical address***): La unidad de memoria manipula direcciones físicas.
- **Direccionamiento virtual** (***virtual address***): Son las direcciones lógicas que se generan cuando existe asociación de direccionamiento en tiempo de ejecución.

Para la asociación de direccionamiento en tiempo de compilación o carga, las direcciones lógicas o físicas coinciden. No es así para la asociación en tiempo de ejecución.

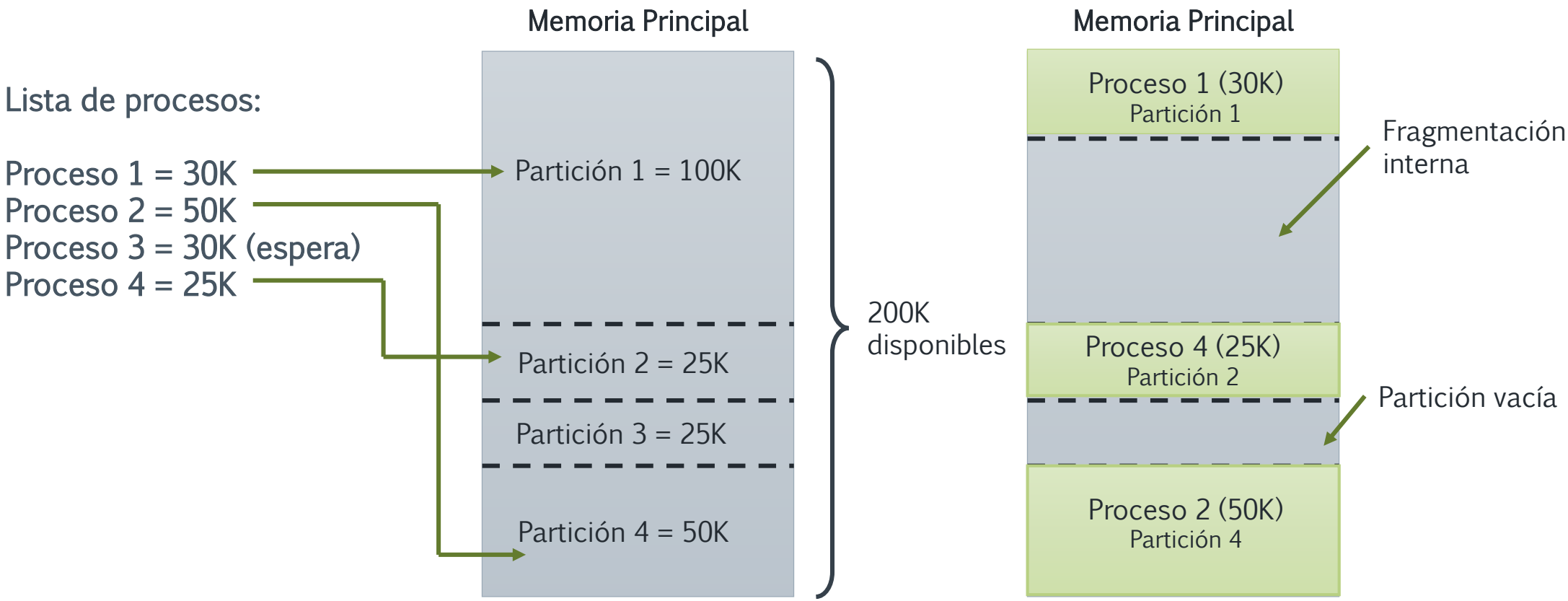
Asignación de Memoria Contigua Partición Estática

Descripción	Ventajas	Desventajas
La memoria principal se divide en un conjunto de particiones fijas diseñadas por el operador del sistema. Un proceso se puede cargar en una partición de menor o igual tamaño. Cada participación podía usarse sólo por un proceso.	Sencilla de implementar; poca sobrecarga del SO.	Empleo ineficiente de la memoria debido a la fragmentación interna; se requería un conocimiento preciso de antemano de todas las tareas a ejecutar en el sistema.

Asignación de Memoria Contigua
Ejemplo Partición Estática



Asignación de Memoria Contigua Ejemplo Partición Estática



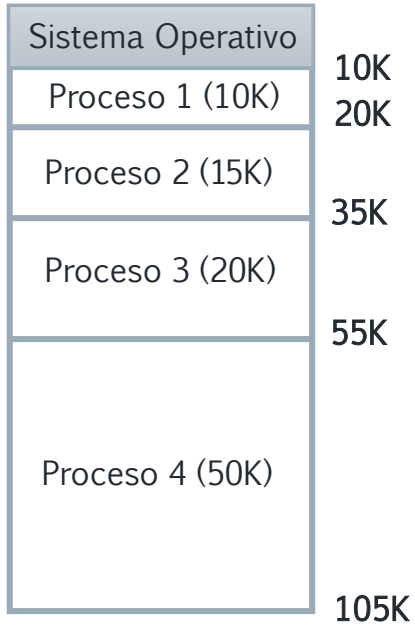
Asignación de Memoria Contigua Partición Dinámica

Descripción	Ventajas	Desventajas
Las particiones se crean dinámicamente, de forma que cada proceso se carga en una partición de exactamente el mismo tamaño que el proceso.	No hay fragmentación interna; uso más eficiente de la memoria principal.	Uso ineficiente del procesador debido a la necesidad de compactación para contrarrestar la fragmentación externa.

Asignación de Memoria Contigua Ejemplo Partición Dinámica

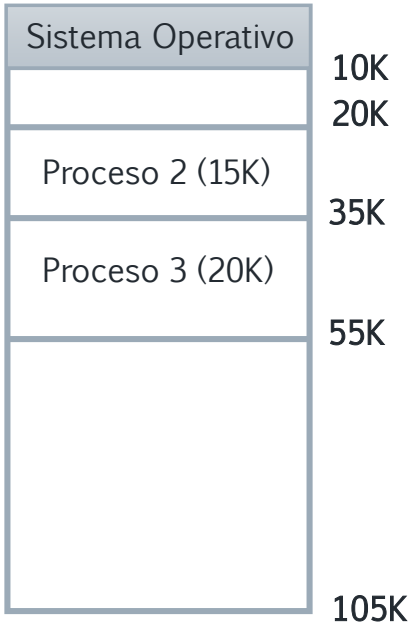
Lista de procesos:

Proceso 1 = 10K
Proceso 2 = 15K
Proceso 3 = 20K
Proceso 4 = 50K



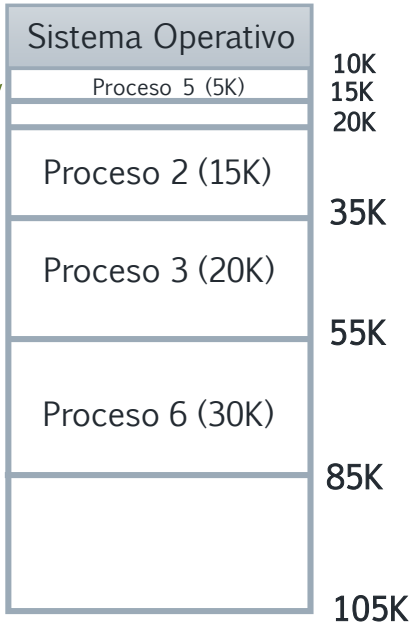
Asignación de memoria para el proceso que llega

Termina el Proceso 1
Termina el Proceso 4



Después de que terminan los Procesos 1 y 4

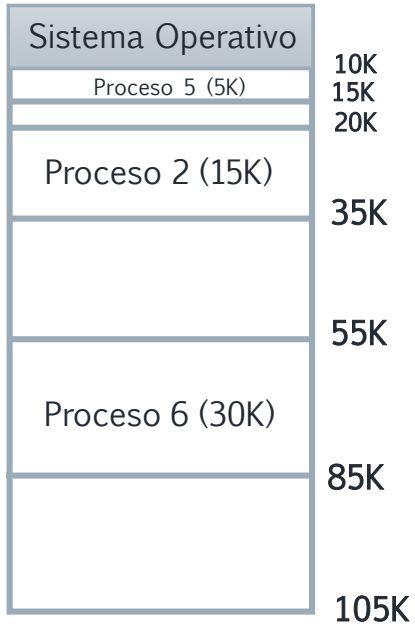
Llega el Proceso 5 (5K)
Llega el Proceso 6 (30K)



Después que han entrado los Procesos 5 y 6

Asignación de Memoria Contigua Ejemplo Partición Dinámica

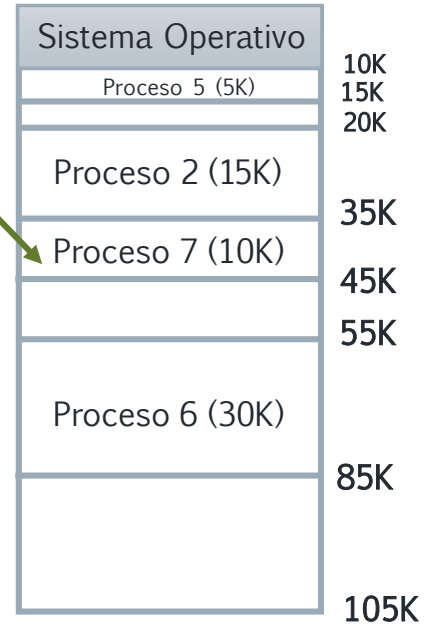
Termina el Proceso 3



Después que termina el Proceso 3

Llega el Proceso 7 (10K)

Llega el Proceso 8 (30K)



El Proceso 8 debe esperar

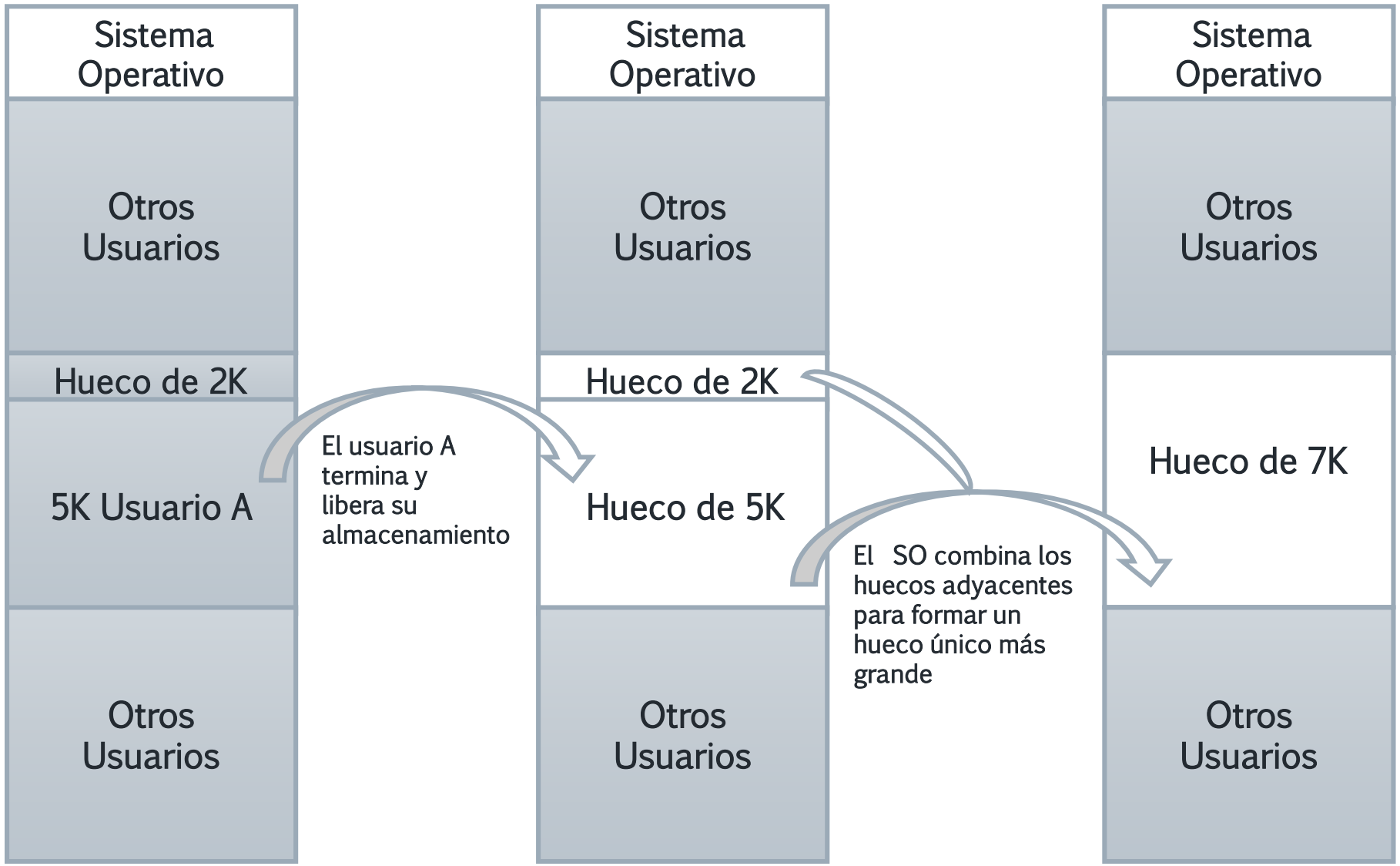
Después que ha entrado el Proceso 7

Condensación

El proceso de fusionar huecos adyacentes para formar un solo hueco más grande se denomina **condensación**.

Mediante la **condensación** de huecos se pueden recuperar los bloques contiguos de almacenamiento más grandes que sea posible.

Condensación



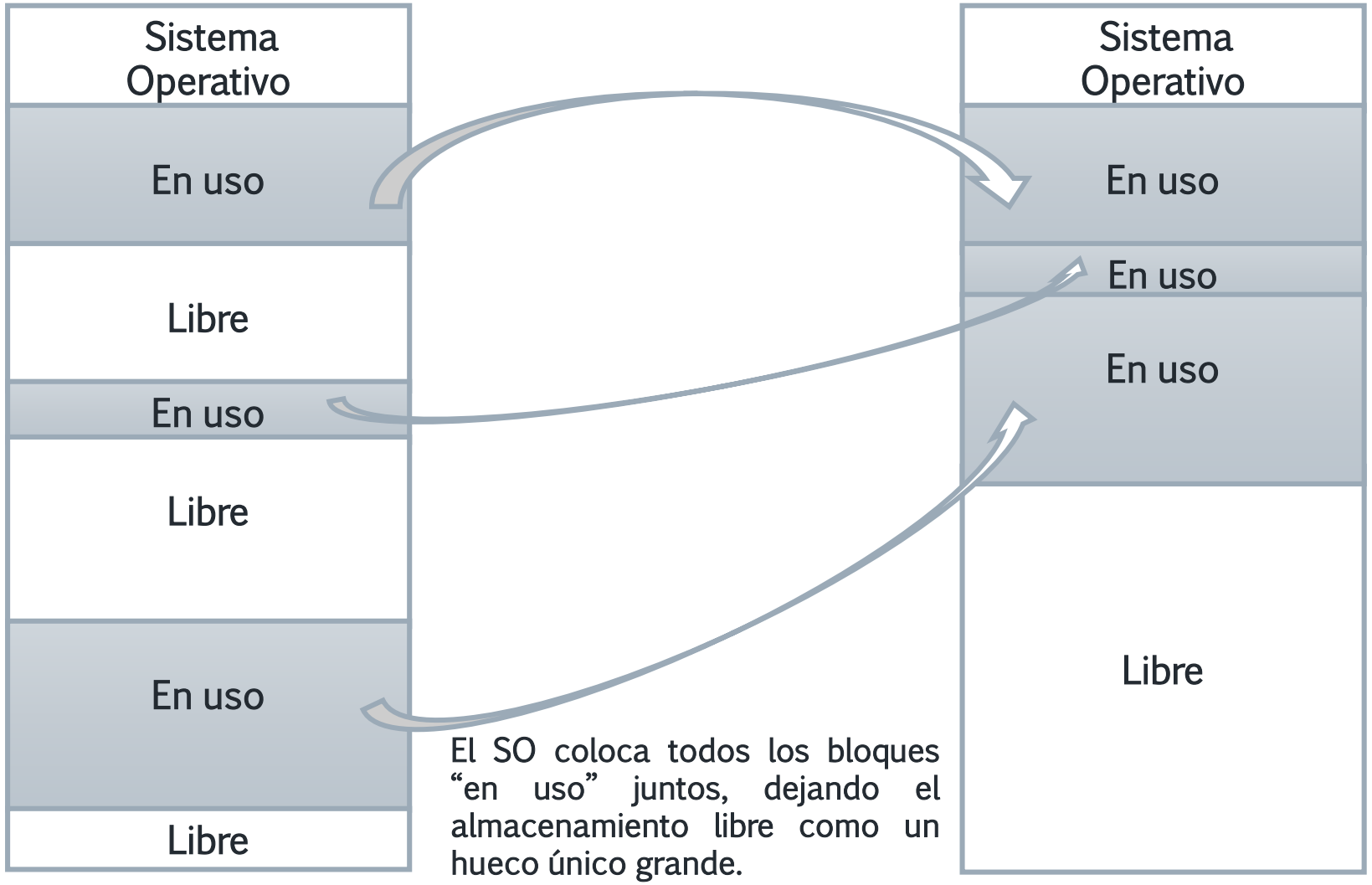
Compactación

La técnica de **compactación** del almacenamiento implica trasladar todas las áreas ocupadas del almacenamiento a algún extremo de la memoria principal.

Esto deja un gran hueco único de almacenamiento libre, en lugar de varios huecos pequeños característicos en la multiprogramación con partición dinámica.

Todo el almacenamiento libre está contiguo, así que un trabajo en espera puede ejecutarse si sus necesidades de memoria son satisfechas por el único hueco resultante de la compactación.

Compactación



El SO coloca todos los bloques "en uso" juntos, dejando el almacenamiento libre como un hueco único grande.

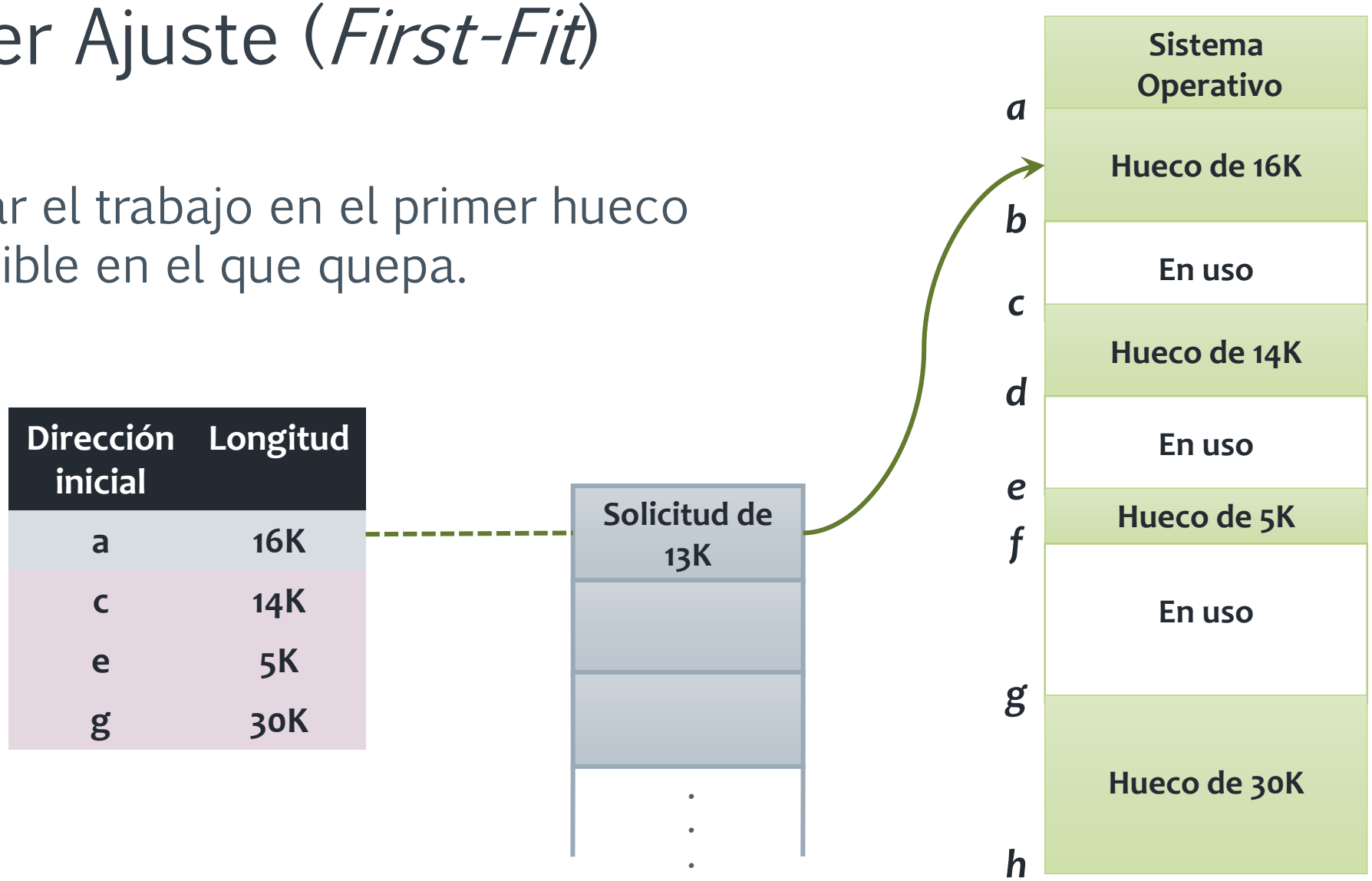
Inconvenientes de la Compactación

- Consume recursos del sistema que podrían utilizarse en forma productiva.
- El sistema debe detener todas sus actividades mientras realiza la compactación.
- La compactación implica reubicar los trabajos que estén en el almacenamiento.

Algoritmos de ubicación Primer Ajuste (*First-Fit*)

Colocar el trabajo en el primer hueco disponible en el que quepa.

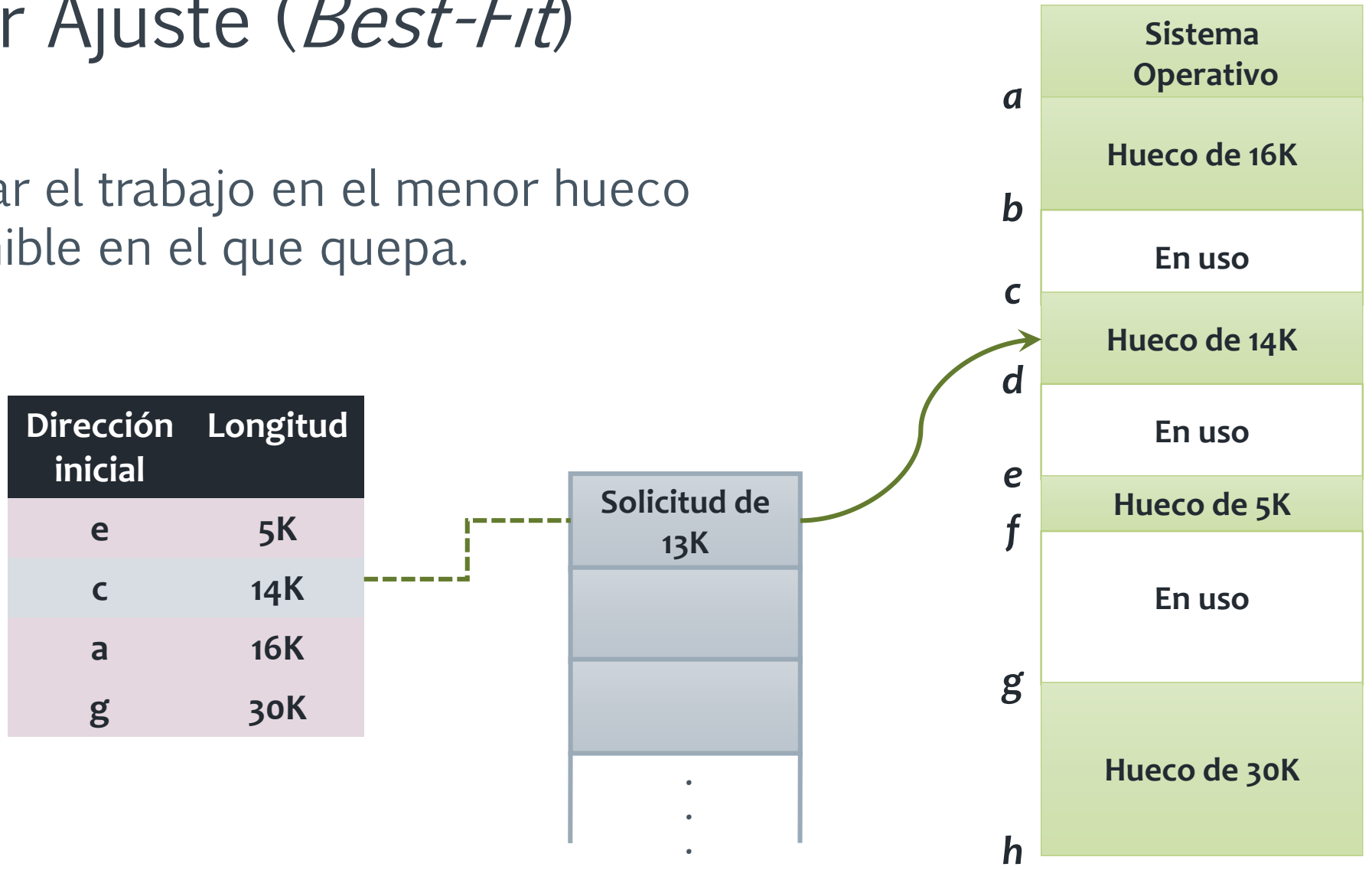
Dirección inicial	Longitud
a	16K
c	14K
e	5K
g	30K



Algoritmos de ubicación
Mejor Ajuste (*Best-Fit*)

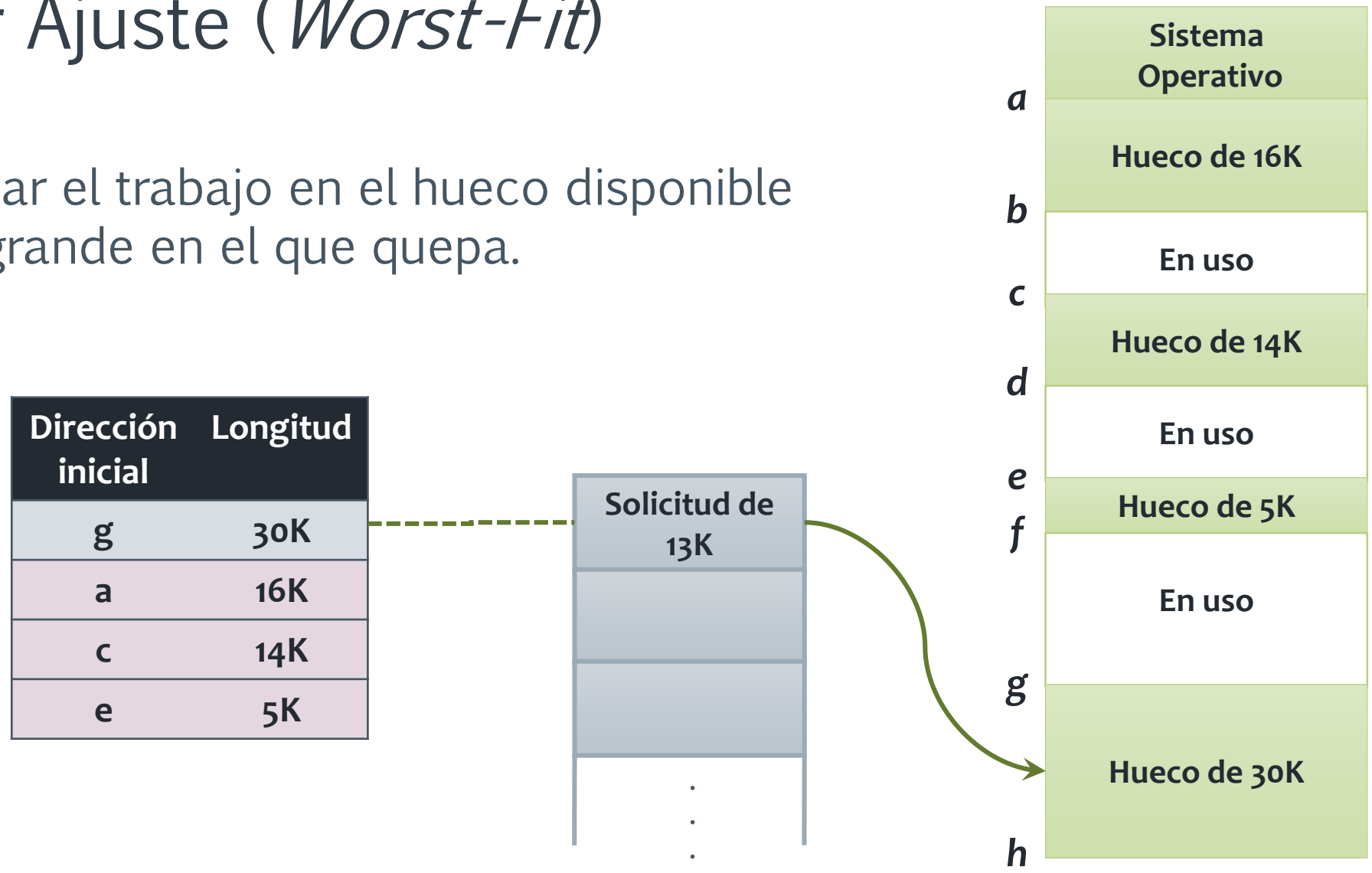
Colocar el trabajo en el menor hueco disponible en el que quepa.

Dirección inicial	Longitud
e	5K
c	14K
a	16K
g	30K



Algoritmos de ubicación
Peor Ajuste (*Worst-Fit*)

Colocar el trabajo en el hueco disponible más grande en el que quepa.



Siguiente Ajuste (*Next-Fit*)

Una variante del algoritmo del **Primer Ajuste** es la del **Siguiente Ajuste**, el cual comienza la búsqueda de un hueco disponible en el lugar donde se terminó la búsqueda anterior.

Ejemplo. Algoritmos de colocación en Partición Estática

Dadas las siguientes particiones de memoria de 100k, 500k, 200k, 300k y 600k (en ese orden). ¿Cómo se colocarían los siguientes procesos: 212k, 417k, 112k y 426k (en ese orden) en la memoria empleando los algoritmos de colocación: primer, mejor, peor y siguiente ajuste?

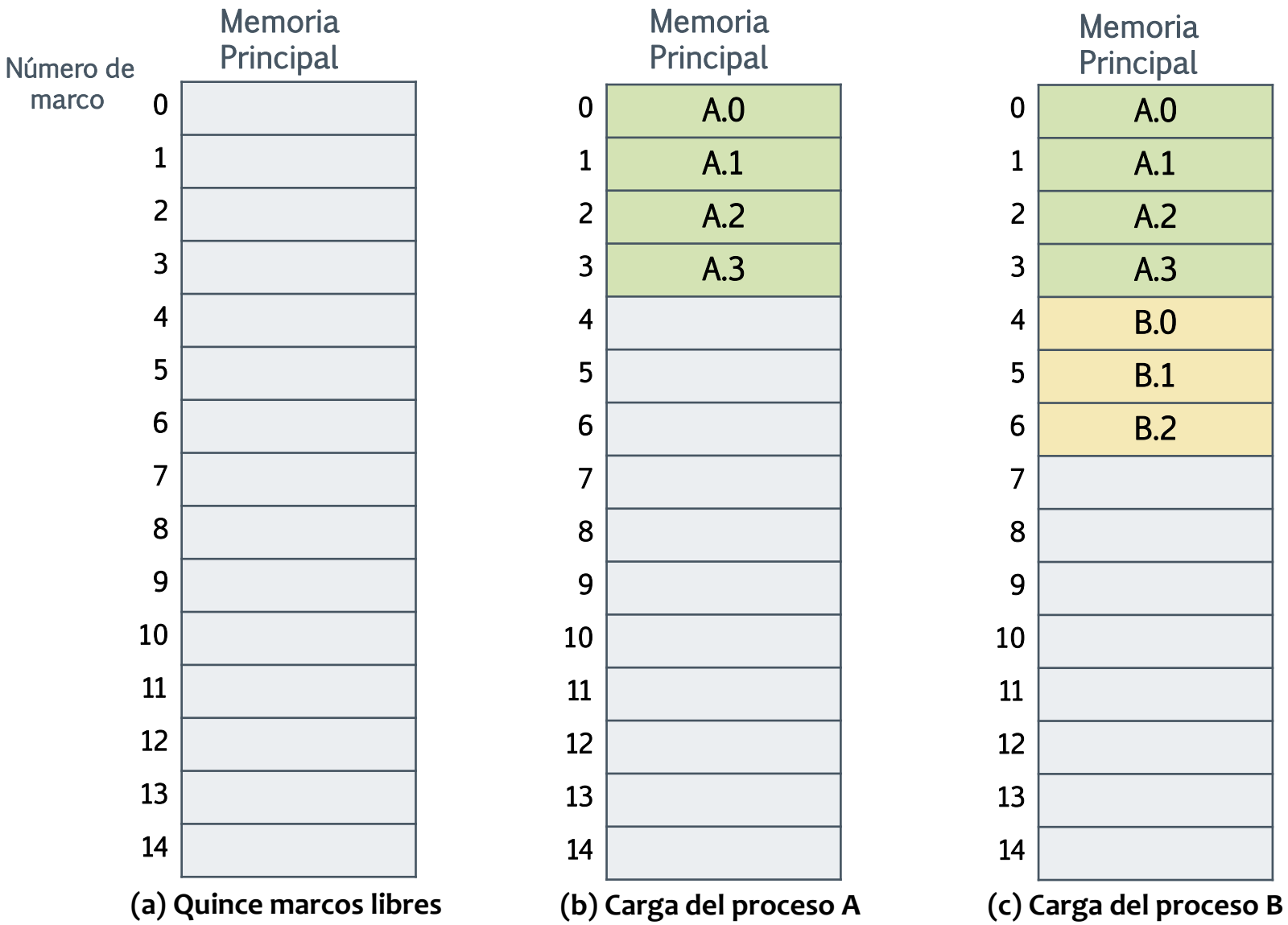
Páginación

- La paginación es una forma de reubicación dinámica.
- Si se usa un esquema de paginación, **no hay fragmentación externa.**
- Cualquier marco libre se puede asignar a un proceso que lo necesite.
- Puede **haber cierta fragmentación interna**.

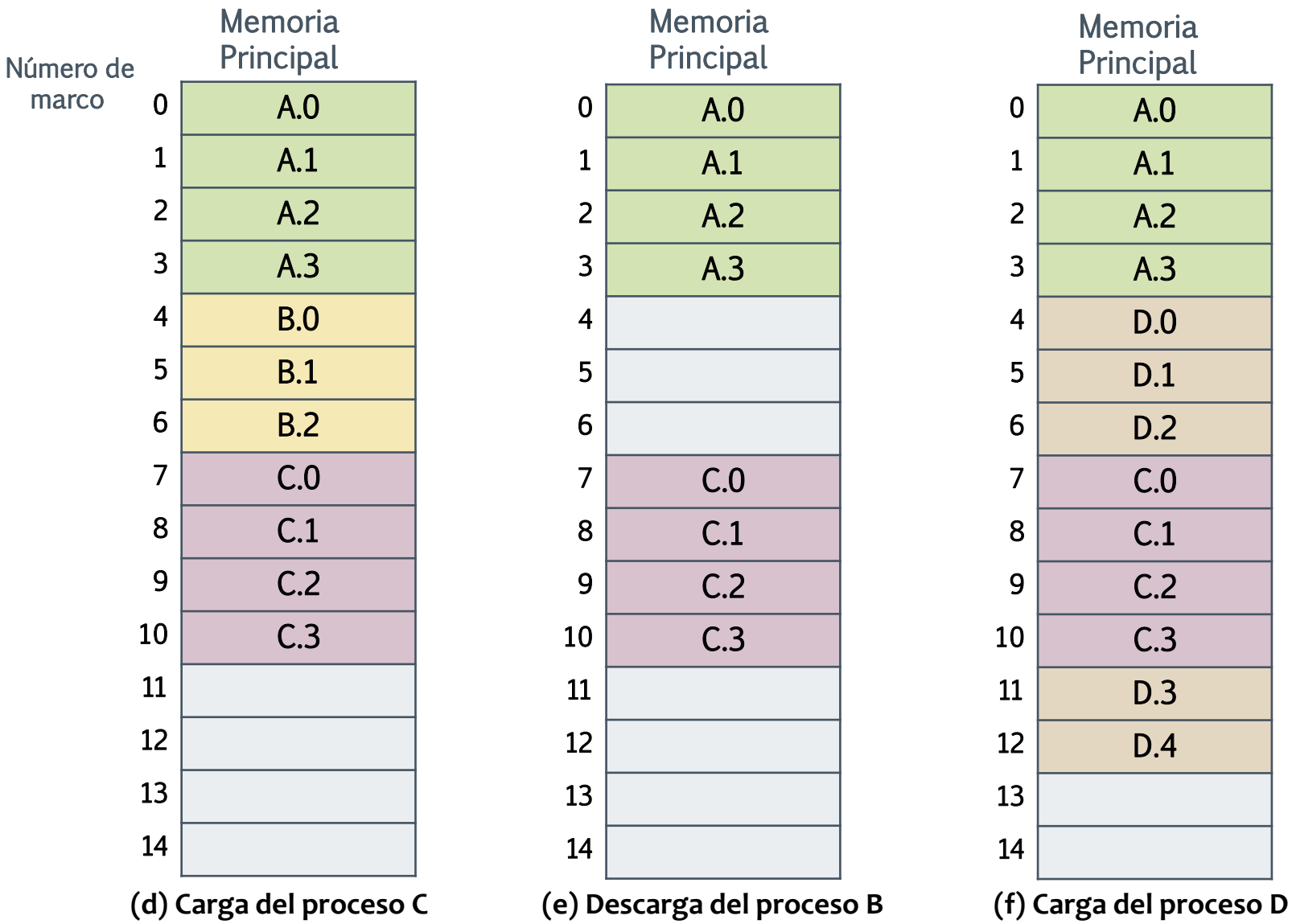
Paginación

- La memoria física se divide en bloques de tamaño físico llamados marcos (***frames***).
- La memoria lógica se divide en bloques del mismo tamaño llamados **páginas**.
- Los **marcos** tienen el mismo tamaño que las **páginas**.
- Cuando se va a ejecutar un proceso, sus páginas se cargan desde el almacenamiento secundario en cualquier marco de memoria que esté disponible.

Ejemplo. Asignación de páginas de procesos a marcos libres



Ejemplo (continuación)



(d) Carga del proceso C (e) Descarga del proceso B (f) Carga del proceso D

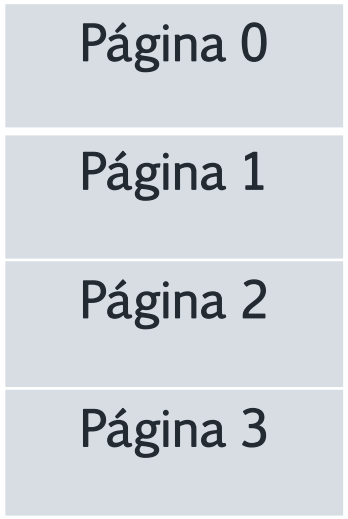
Tabla de Páginas

La dirección lógica tiene la estructura:

- **Número de página (p).**
- **Desplazamiento en la página (d).**

- El número de página se utiliza como índice de la **Tabla de Páginas**.
- La Tabla de Páginas contiene la **dirección base** de cada página en la memoria física.
- Esta dirección se combina con el **desplazamiento** en la página para definir la dirección de memoria física que se envía a la unidad de memoria.

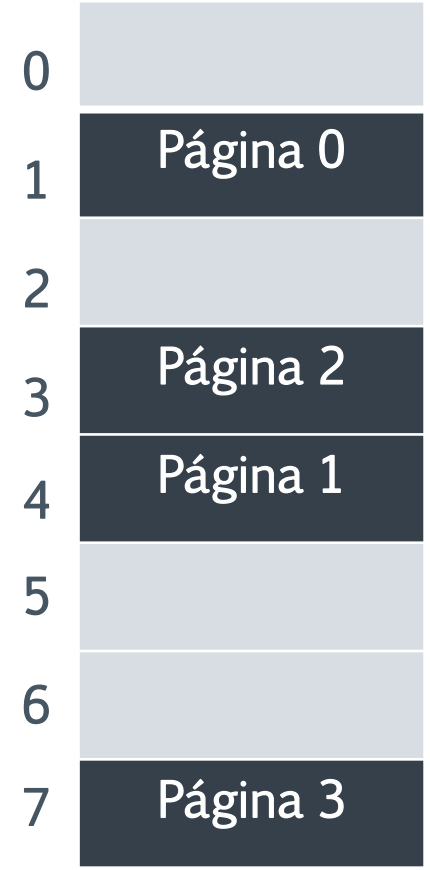
Modelo de Paginación



Memoria Lógica

Número de Página	Número de Marco
0	1
1	4
2	3
3	7

Tabla de Páginas



Memoria Física

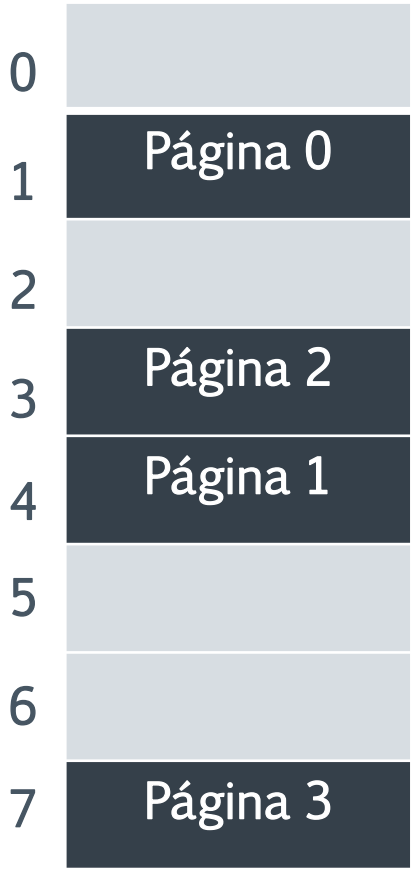
Ejemplo1 . Traducir dirección lógica a dirección física

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

Memoria Lógica

Número de Página	Número de Marco
0	1
1	4
2	3
3	7

Tabla de Páginas



Memoria Física

Tamaño de página = 4 bytes
Memoria física = 32 bytes
(8 páginas)

Dirección lógica 0:
página = 0
desplazamiento = 0
página 0 → marco 1

Dirección física 20:

marco * tamaño_página + desplazamiento

1 * 4 + 0 = 4

Traducción de dirección lógica a dirección física

Tenemos una dirección de **n + m bits**, en la que los **n bits más significativos** son el **número de página** y los **m bits menos significativos** son el **desplazamiento**.

Para traducir la dirección, tenemos que:

- Extraer el número de página de los n bits más significativos de la dirección lógica.
- Emplear el número de página como índice en la tabla de páginas del proceso para encontrar el número de marco **k**.
- El comienzo de la dirección física del marco es k x 2^m y la dirección física del byte referenciado es este número más el desplazamiento.
- La dirección física se construye concatenando el número de marco con el desplazamiento.

Ejemplo 2 . Traducir dirección lógica a dirección física

Se emplean direcciones de 16 bits y el tamaño de la página es de 1K=1024 bytes. Tenemos que:

- Con un tamaño de página de 1K se necesitan 10 bits para el campo de desplazamiento, dejando 6 bits para el número de página.
- De acuerdo con lo anterior, un programa puede estar formado por un máximo de $2^6 = 64$ páginas de 1Kb cada una.

Ejemplo 2 (continuación)

Si tenemos la **dirección virtual 1502** su representación en binario es **0000010111011110**.

De acuerdo con lo anterior tenemos que:

Desplazamiento → 10 bits

Número de página → 6 bits

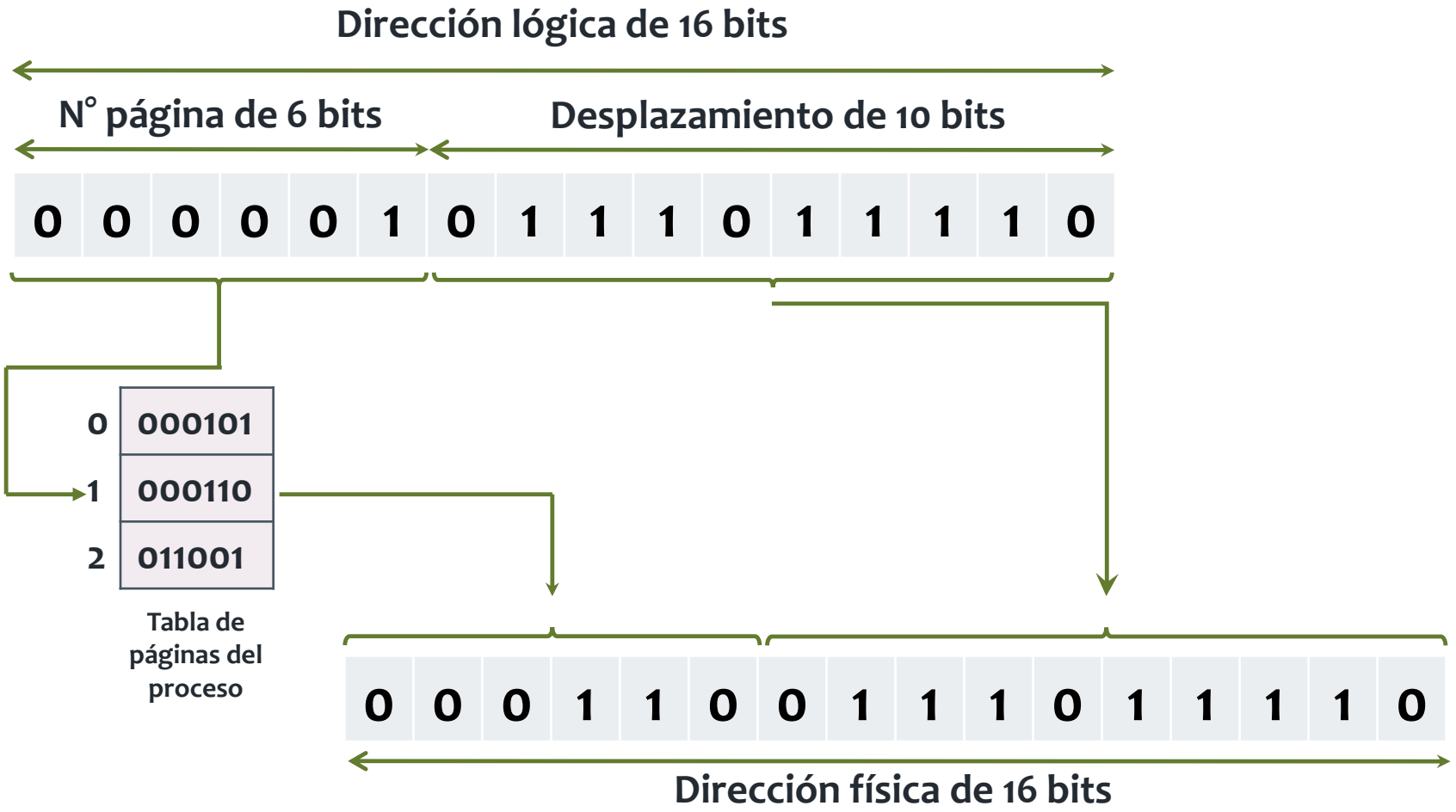
Por lo tanto, la **dirección virtual 1502** tiene:

Número de página : 1 (000001)

Desplazamiento : 478 (0111011110)

Ejemplo 2 (continuación)

Suponiendo que la página reside en el **marco** de memoria principal **6** cuya representación en binario es **000110**. Su dirección física es **6622**.



Segmentación

- La **segmentación** es otro modo de subdividir el programa.
- El programa y los datos asociados se dividen en un conjunto de segmentos.
- No es necesario que todos los segmentos de todos los programas tengan la misma longitud, aunque existe una longitud máxima de segmento.
- Un programa puede ocupar más de una partición y éstas no tienen por qué estar contiguas.

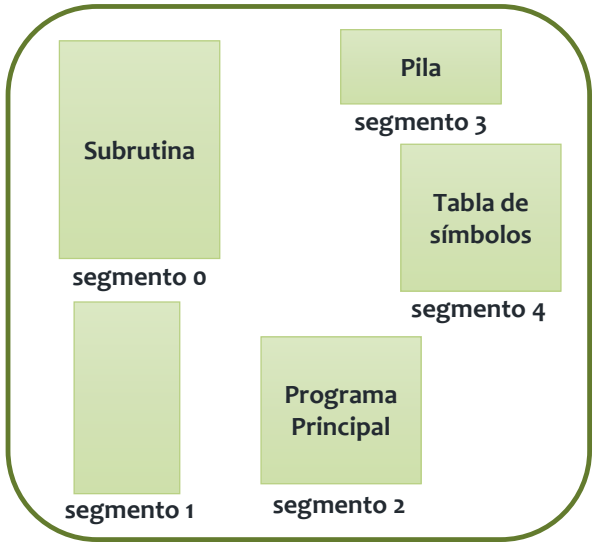
Segmentación

- La segmentación **elimina** la **fragmentación interna**, pero **sufre** de **fragmentación externa**. Sin embargo, será menor debido a que los procesos se dividen en un conjunto de partes más pequeñas.
- A diferencia de la **paginación**, la **segmentación** es **visible para el programador** y se ofrece como una ventaja para la organización de programas y datos.
- El principal inconveniente es que el programador debe ser consciente de la limitación del tamaño máximo de los segmentos.

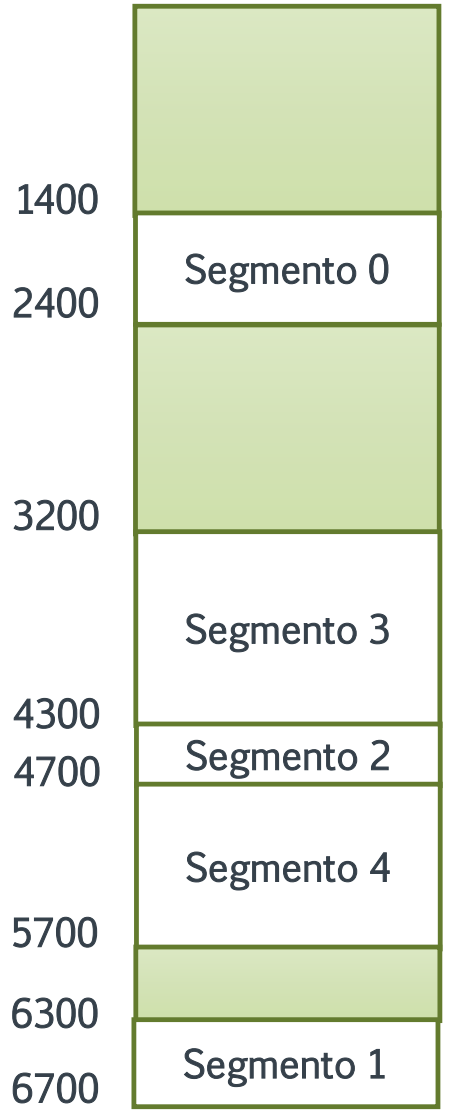
Tabla de Segmentos

- Cada entrada de la tabla tiene una **base de segmento** y un **límite de segmento**.
- La **base** contiene la **dirección física inicial** donde el segmento reside en la **memoria física**.
- El **límite** especifica la **longitud del segmento**.
- Una dirección lógica consta de:
  - **Número de segmento**, **s**.
  - **Desplazamiento del segmento**, **d**.
- El desplazamiento **d** de la **dirección lógica** debe estar **entre 0** y el **límite del segmento**.
- Si el desplazamiento es válido, se suma a la base del segmento para producir la dirección del byte deseado de la memoria física.

Ejemplo1 . Traducir dirección lógica a dirección física



	Límite	Base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700



Segmento 2 tiene 400 bytes de longitud e inicia en la posición 4300.

- Una referencia al **byte 53** del **segmento 2** se transforma en la posición 4300 + 53 = 4353
- Una referencia al **byte 852** del **segmento 3** se transforma en la posición 3200 + 852 = 4052
- Una referencia al **byte 1222** del **segmento 0** causaría una transferencia por trampa al SO, debido a que el segmento 0 sólo tiene 1000 bytes de longitud.

Traducción de dirección lógica a dirección física

Tenemos una dirección de **n + m bits**, en la que los **n bits más significativos** son el **número de segmento** y los **m bits menos significativos** son el **desplazamiento**.

Para traducir la dirección, tenemos que:

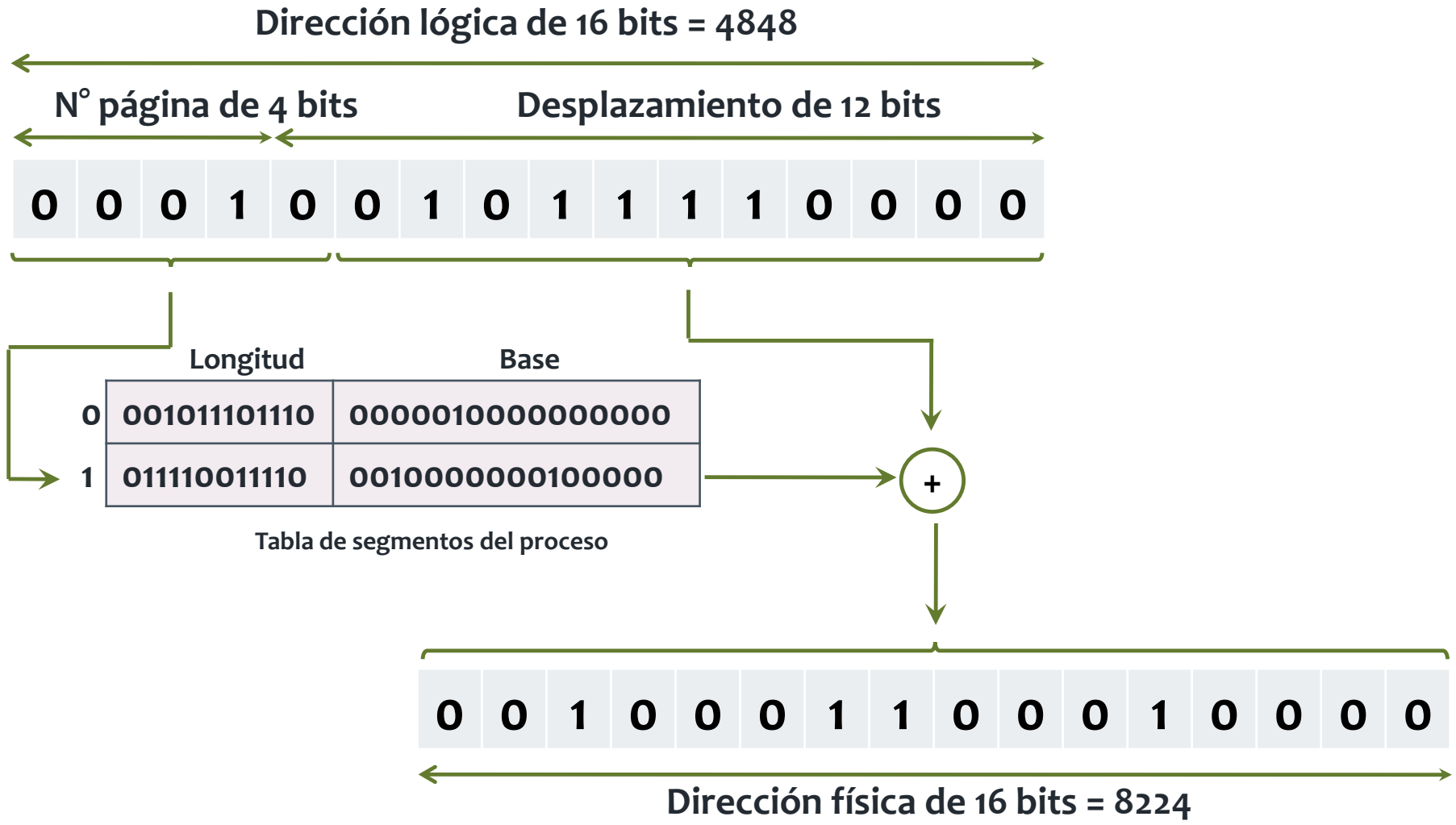
- Extraer el número de segmento de los n bits más significativos de la dirección lógica.
- Emplear el número de segmento como índice en la tabla de segmentos del proceso para encontrar la dirección física de comienzo del segmento.
- Comparar el desplazamiento, expresado por los m bits menos significativos, con la longitud del segmento. Si el desplazamiento es mayor que la longitud, la dirección no es válida.
- La dirección física buscada es la suma de la dirección física de comienzo del segmento más el desplazamiento.

Ejemplo 3. Traducir dirección lógica a dirección física

Si tenemos que n = 4 y m = 12, el máximo tamaño del segmento es $2^{12} = 4096$.

Se desea obtener la dirección física de la dirección lógica 4848.

Ejemplo 3 (continuación)



Memoria Virtual

- Ofrecer a los procesos mayor espacio en memoria del que existe físicamente, el sistema emplea espacio en **almacenamiento secundario** (típicamente, disco duro), a través de un esquema de **intercambio** (***swap***) guardando y trayendo páginas enteras.
- La memoria virtual es gestionada de **forma automática** y **transparente** por el sistema operativo.
- Con la memoria virtual, la cantidad de memoria disponible para procesamiento de procesos puede ser mucho mayor que la memoria física disponible.

Paginación por demanda

- La **paginación sobre demanda** significa que, para comenzar a ejecutar un proceso, el SO carga **solamente las páginas que se requieren** para comenzar la ejecución. Esto demanda acceso de alta velocidad a las páginas.
- A lo largo de la ejecución, sólo carga a memoria las páginas cuando van a ser utilizadas e incluso puede ocurrir que las páginas que no sean requeridas, nunca serán siquiera, cargadas a memoria.

Paginación por demanda en Memoria Virtual

Los elementos que debe contener la **Tabla de Correspondencia de Páginas** (**TCP**) son:

Bit de residencia de página	Dirección en memoria secundaria	Número de marco de página
r	s	p’

r=0 si la página no está en memoria principal
r=1 si la página está en memoria principal

Paginación por demanda en Memoria Virtual Traducción por Correspondencia Directa

Éste método permite encontrar la dirección real (dirección física) a partir de la dirección virtual, utilizando la estructura de la **TCP**.

La dirección virtual tiene la estructura:

V = (p, d)

Donde

p = número de página

d = desplazamiento

Paginación por demanda en Memoria Virtual Errores

- **Error de página:** Cuando una página no está en memoria principal: **r = 0**.
- **Error por desbordamiento:** Cuando el **desplazamiento** es **mayor** o **igual** que el **tamaño de la página**.

Ejemplo. Paginación por demanda en Memoria Virtual

Tenemos dos procesos A y B, cada uno con su Tabla de Correspondencia de Páginas.

TCP Proceso A

r	s	p'
0	520	5
1	410	20
1	235	11
0	450	8
1	210	5
0	135	14
0	280	10

TCP Proceso B

r	s	p'
1	320	8
1	100	4
0	70	12
0	300	11
0	720	10

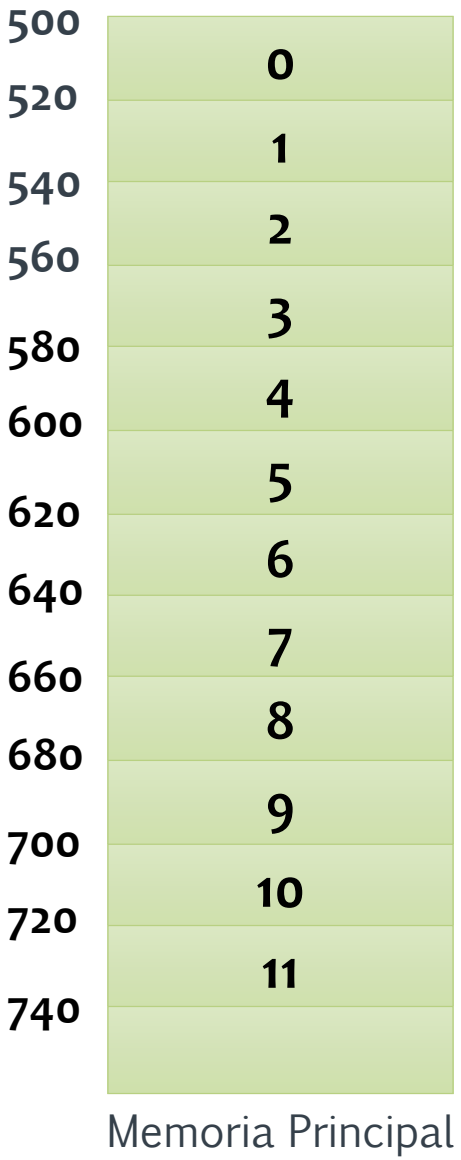
La **memoria física inicia** a partir de la **dirección 500** y el **tamaño de la página** es de **20**. Obtener las direcciones físicas de las siguientes direcciones virtuales, a partir de las TCP's de cada proceso:

a) **P(A)v = (2, 18)**

b) **P(B)v = (1, 7)**

c) **P(A)v = (3, 5)**

Ejemplo (continuación)



Memoria Principal

Para encontrar la dirección física, ubicamos la dirección en memoria principal en donde está cargado el marco de memoria, donde se aloja la página (se observa en la TCP en el campo p') y le sumamos el desplazamiento.

a) P(A)v = (2,18)

Observando la TCP del Proceso A, la página 2 está cargada en el marco 11 y éste comienza en la dirección 720. Por lo tanto:

Dirección física = 720 + 18 = 738

b) P(B)v = (1,7)

Observando la TCP del Proceso B, la página 1 está cargada en el marco 4 y éste comienza en la dirección 580. Por lo tanto:

Dirección física = 580 + 7 = 587

c) P(A)v = (3,5)

Observando la TCP del Proceso A, la página 3 no está en memoria principal, ya que el valor del campo *r* es cero. Por lo tanto, hay un **error de página**.

Segmentación por demanda en Memoria Virtual

Los elementos que debe contener la **Tabla de Correspondencia de Segmentos** (**TCS**) son:

Bit de residencia del segmento	Dirección en memoria secundaria	Longitud del segmento	Bits de protección	Dirección base del segmento
r	s	L	r | w | x | a	s'

r=**0** si el segmento no está en memoria principal
r=**1** si el segmento está en memoria principal

Bits de protección (1 = si lo posee; 0 = no lo posee)

r acceso de lectura
w acceso de escritura
x acceso de ejecución
a acceso de adición

Segmentación por demanda en Memoria Virtual Traducción por Correspondencia Directa

Éste método permite encontrar la dirección real (dirección física) a partir de la dirección virtual, utilizando la estructura de la **TCS**.

La dirección virtual tiene la estructura:

V = (s, d)

Donde

s = número de segmento

d = desplazamiento

Segmentación por demanda en Memoria Virtual Errores

- **Error de segmento:** Cuando un segmento no está en memoria principal: **r = 0**.
- **Error por desbordamiento:** Cuando el **desplazamiento** es **mayor** o **igual** que la **longitud del segmento**.
- **Error de protección de segmento**: Cuando se solicita una operación que no esta permitida. Éste error se detecta comprobando los bits de protección, y si ocurre se cancela la ejecución del proceso.

Segmentación por demanda en Memoria Virtual Ejemplo

Tenemos la siguiente TCS para los **Procesos A, B y C**.

El **Proceso A** inicia a partir de la **dirección física 100**.

El **Proceso B** inicia a partir de la **dirección física 112**.

El **Proceso C** inicia a partir de la **dirección física 105**.

Dirección física	r	s	L	r	w	x	a	s'
100	0	1024	10	1	0	0	1	532
101	1	1035	45	1	1	1	0	1150
102	1	532	28	0	1	1	0	1200
103	0	1150	37	0	1	0	0	45
104	1	1200	36	0	1	0	1	118
105	1	45	20	0	0	1	0	2005
106	0	118	70	1	1	0	1	2310
107	1	2005	55	0	1	0	1	324
108	0	2310	100	1	1	1	1	1501
109	0	324	35	0	0	0	0	1708
110	1	1501	62	0	1	0	0	1810
111	1	1708	48	1	1	0	1	1340
112	1	1810	30	1	0	0	0	1492
113	0	1340	80	1	1	1	1	1232
114	1	1492	27	1	0	0	0	750
115	1	1237	75	0	1	0	1	824
116	0	750	62	0	1	1	0	1024
117	1	824	46	1	1	0	1	1035

Paginación y Segmentación Combinadas

- El espacio virtual de direcciones de un proceso se divide en **segmentos** que se administran a nivel de **páginas**. Por otra parte, la **memoria principal** se divide en **marcos**.
- Un **segmento** estará compuesto de un número entero de **páginas**. Los segmentos deben ser de un tamaño tal que sea múltiplo del tamaño de la página, el caso más crítico es que el segmento sea del tamaño de la página.
- El intercambio de información se hace a nivel de páginas, por lo tanto no es necesario que todas las páginas de un segmento estén al mismo tiempo en memoria principal.
- Las páginas que son contiguas en memoria virtual no tienen porque ser contiguas en memoria principal.

Paginación y Segmentación Combinadas

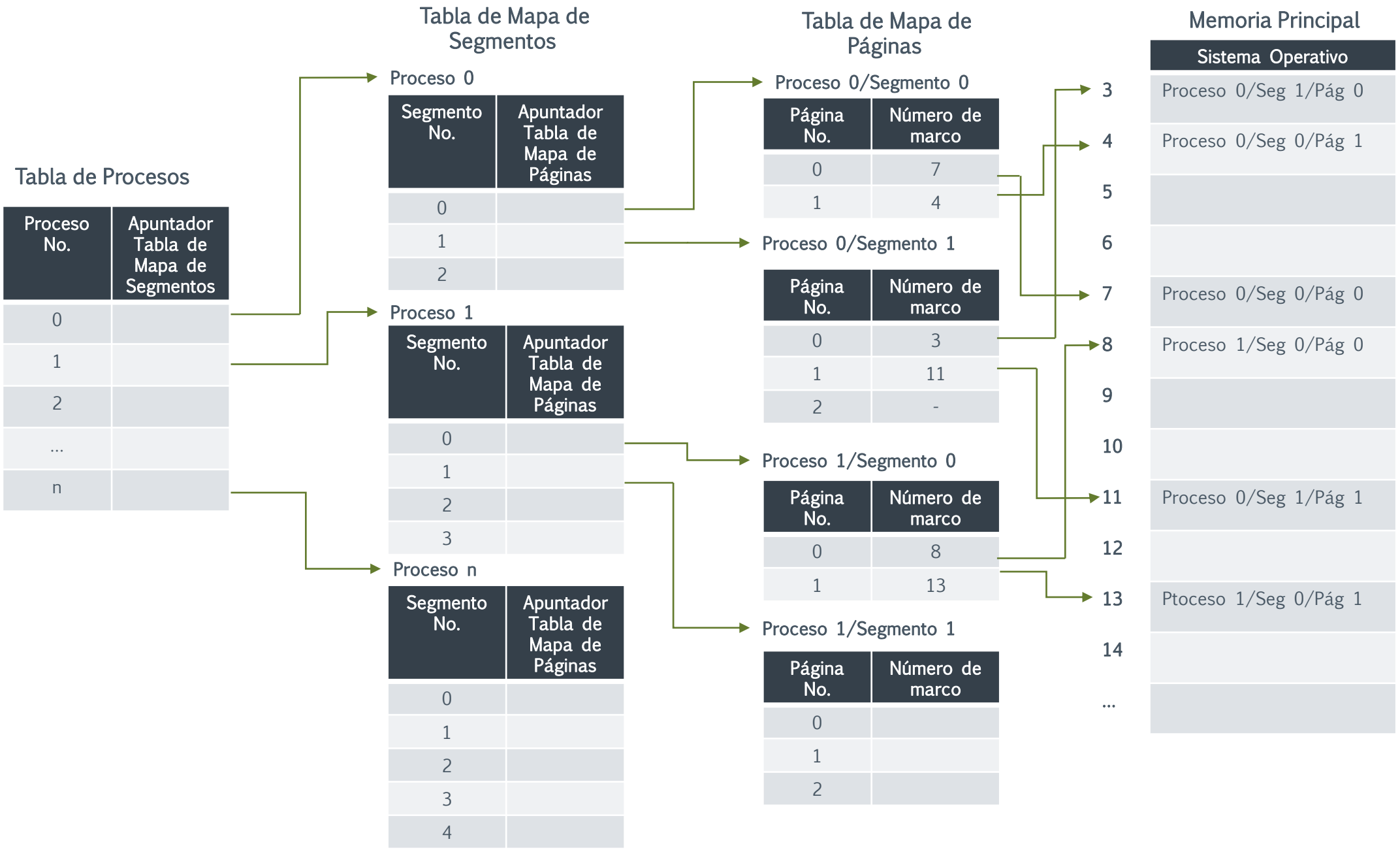
La dirección virtual estará dada por:

V = (**s**, **p**, **d**)

donde:

s	Número de segmento
p	Número de página
d	Desplazamiento

Paginación y Segmentación Combinadas: Tablas



Paginación y Segmentación Combinadas Ejemplo

Tabla de Segmentos del Proceso A

Segmento 0	r	m_1	s'	l	r	w	x	a
Segmento 1	r	m_1	s'	l	r	w	x	a
Segmento 2	**r**	**m_1**	**s'**	**l**	**r**	**w**	**x**	**a**

Tabla de Páginas del Segmento 2 del Proceso A

Página 0	r	m2	p'
Página 1	r	m2	p'
Página 2	r	m2	p'

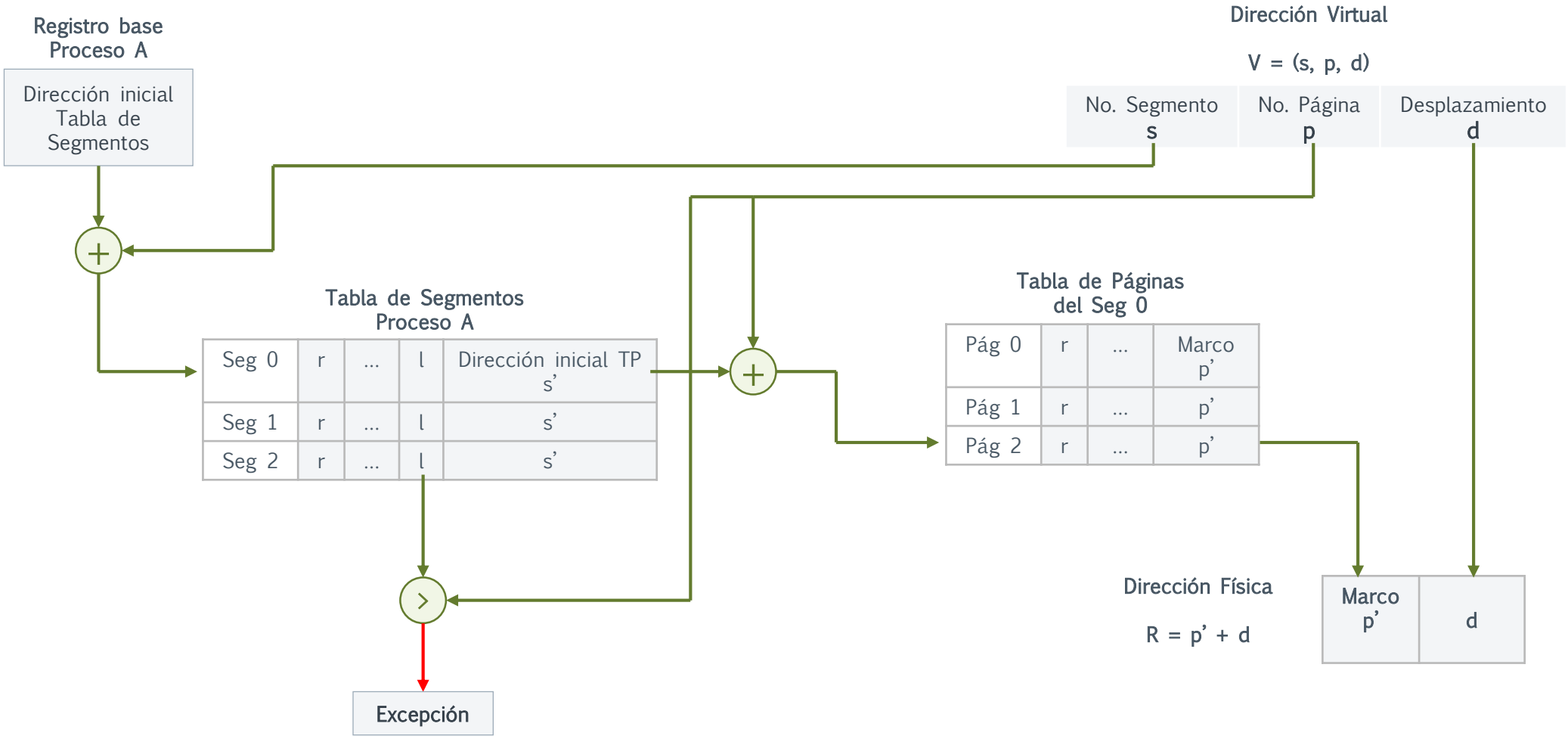
Formato de un elemento de la Tabla de Segmentos:

r	bit de residencia del segmento en memoria principal.
m_1	dirección de almacenamiento secundario.
s'	dirección real del comienzo de su tabla de páginas.
l	longitud del segmento (número de páginas).
r, w, x, a	bits de protección (1 si lo posee, 0 no lo posee). **r** acceso de lectura, **w** acceso de escritura, **x** acceso de ejecución, **a** acceso de adición.

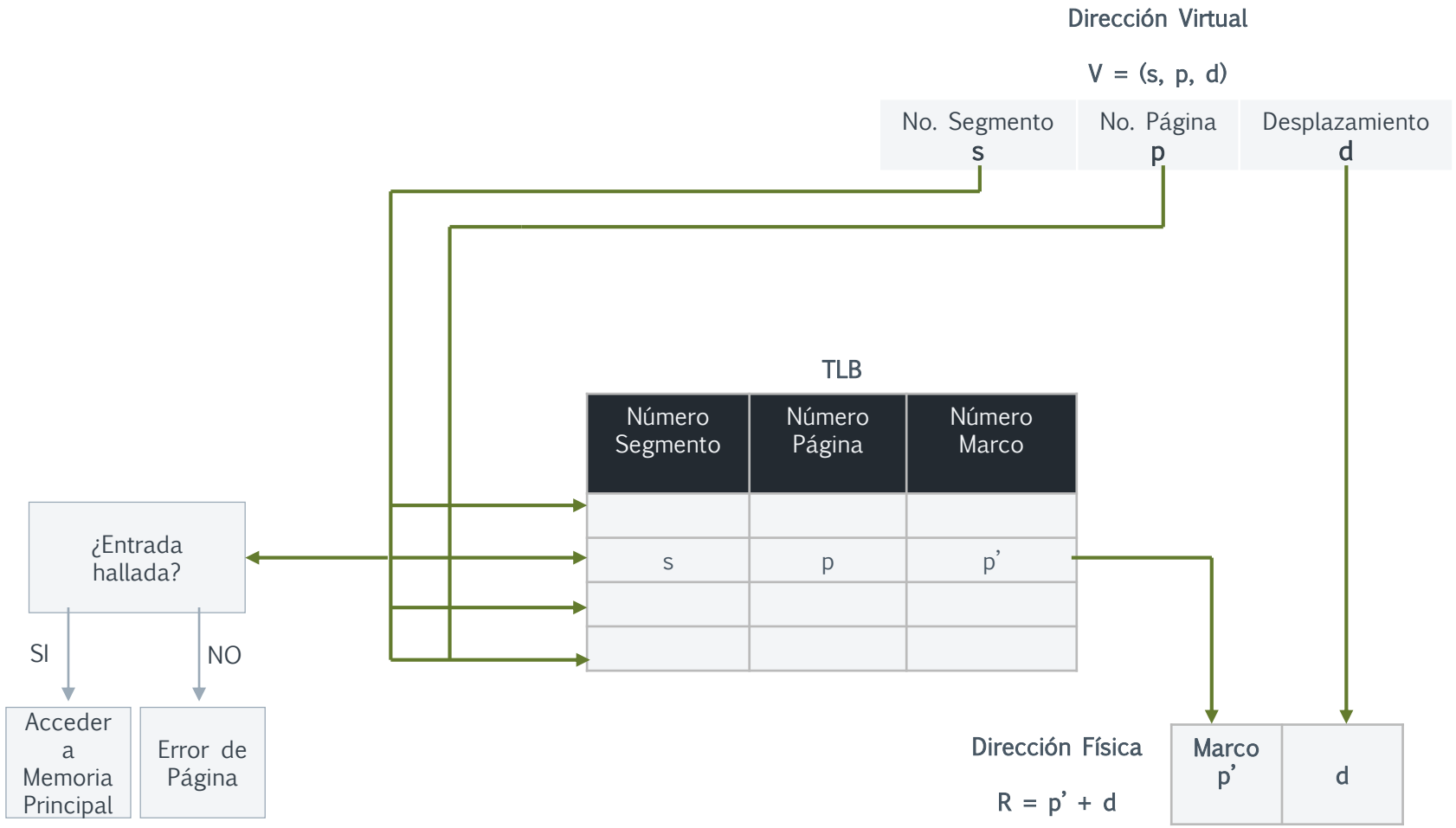
Formato de un elemento de la Tabla de Páginas:

r	bit de residencia de página en memoria principal.
m_2	dirección de almacenamiento secundario.
p'	dirección real del comienzo del marco de página.

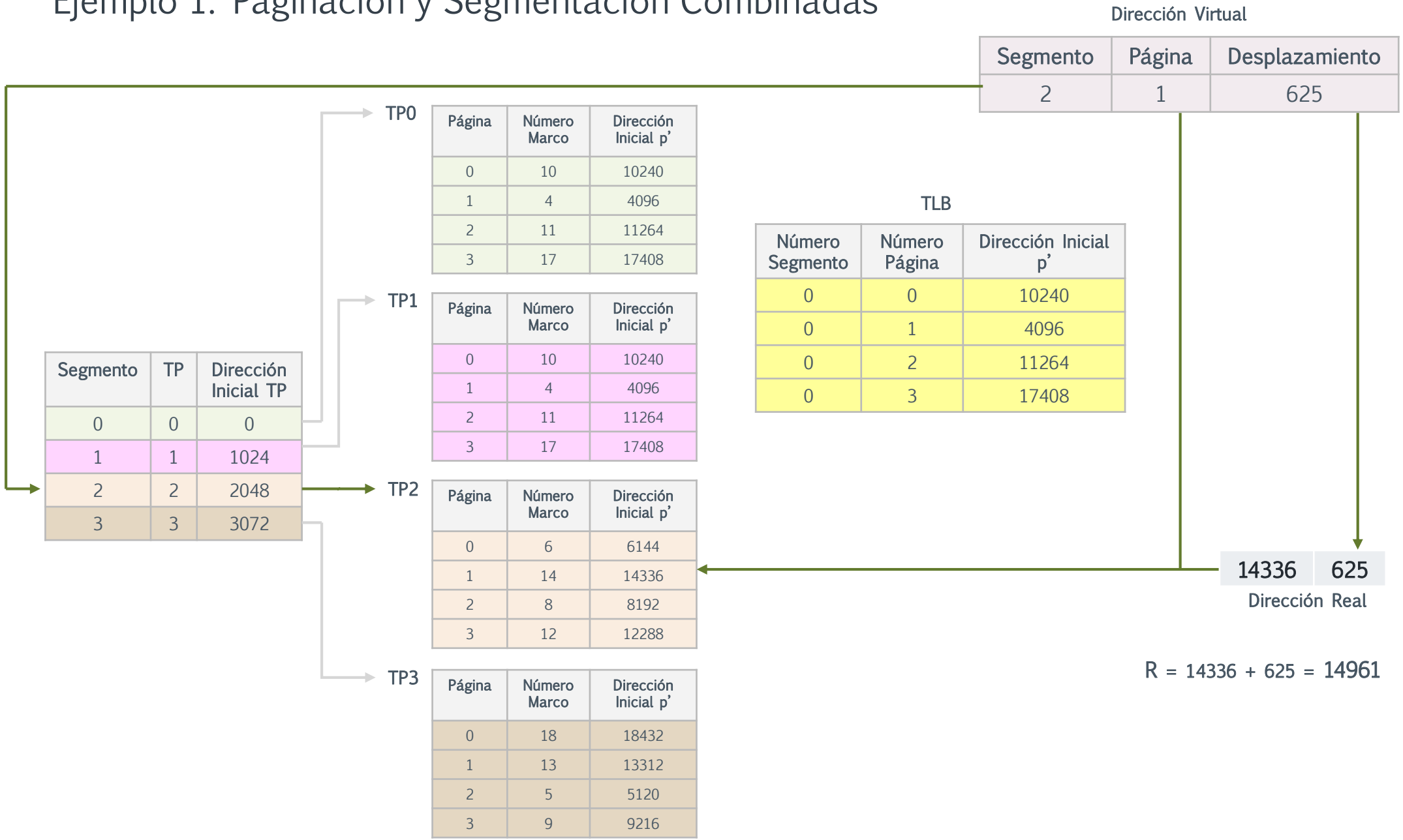
Paginación y Segmentación Combinadas
Traducción directa



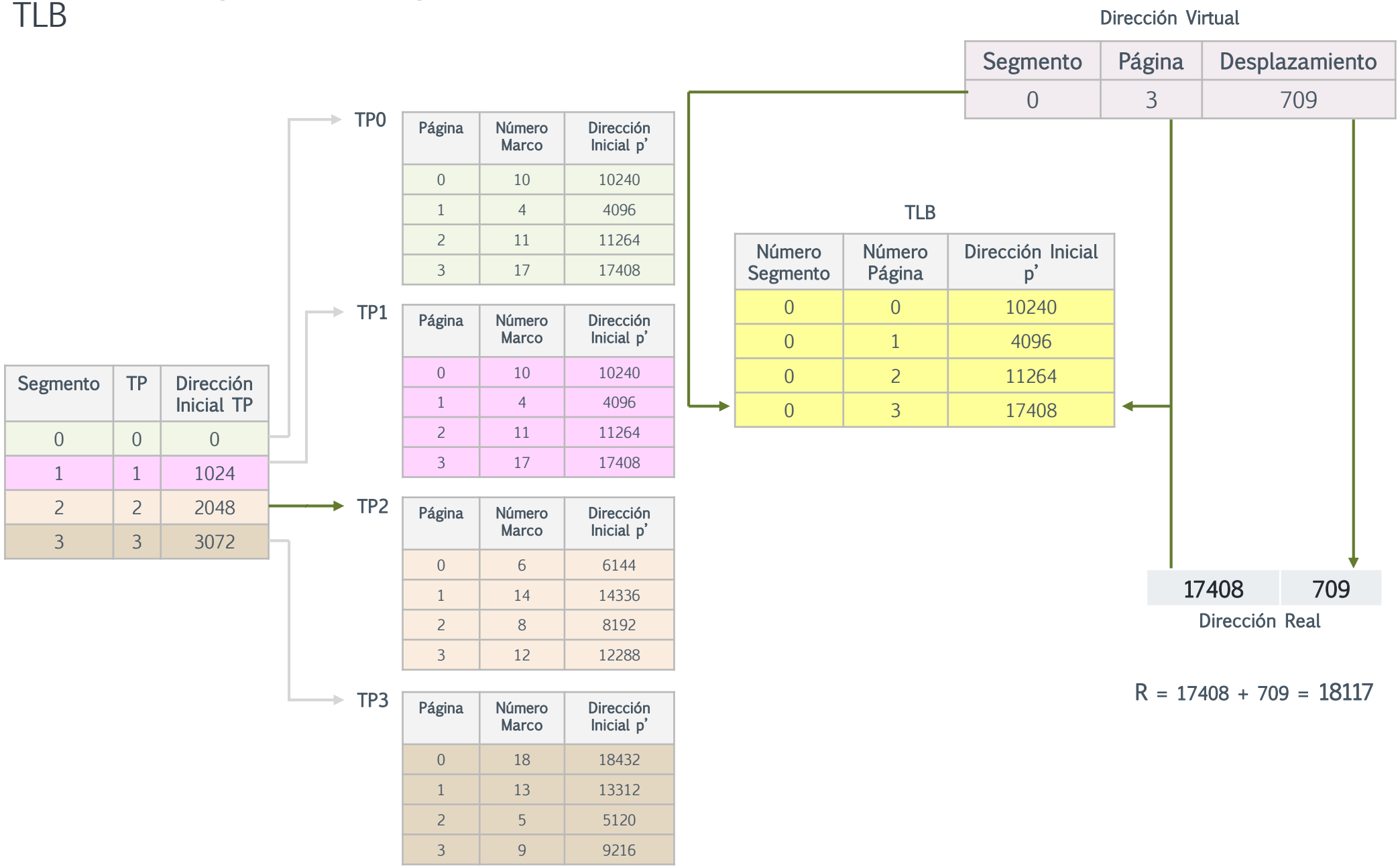
Paginación y Segmentación Combinadas
Traducción asociativa (TLB, *Translation Lookaside Buffer)*



Ejemplo 1. Paginación y Segmentación Combinadas



Ejemplo 2. Paginación y Segmentación Combinadas TLB



Paginación y Segmentación Combinadas Errores

- **Error de pérdida de segmento**

  Ninguna de las páginas del segmento esta en memoria principal. Este error se detecta comprobando el bit de residencia **r** de la tabla de segmentos y se soluciona creando la tabla de páginas para dicho segmento y cargando la página referenciada en memoria principal.

- **Error de pérdida de página**

  Alguna de las páginas del segmento al que pertenece la página referenciada está en memoria principal, pero la página en cuestión no. Esto se detecta comprobando el bit de residencia **r** de la tabla de páginas para el citado segmento y se soluciona cargando dicha página desde el almacenamiento secundario.

Paginación y Segmentación Combinadas Errores

- **Error de desbordamiento de segmento**

  Se hace referencia a una página que no posee el segmento (se verifica si **p ≤ l**) y por lo tanto se aborta la ejecución del proceso.

- **Error de pérdida de página**

  Cuando los bits de protección indican que la operación solicitada no está permitida, se cancela la ejecución del proceso.

Reemplazo de páginas

- Cuando todos los marcos de memoria principal estan ocupados y es necesario traer una nueva página ocurre un error de página (*fault page*).
- Si hay un error de página se debe:
  - Encontrar la página demandada en memoria secundaria.
  - Encontrar un marco libre o liberarlo usando un Algoritmo de Reemplazo de Páginas.
  - Cargar la página en memoria principal (*page in*).
  - Transferir el control al proceso de usuario.
- La política de reemplazo se encarga de seleccionar la página a reemplazar de entre las que están actualmente en memoria.
- Se pretende utilizar el algoritmo que seleccione páginas que causen la frecuencia de errores más baja.

Reemplazo de páginas
Cadena de referencias

Se tiene la siguiente cadena de referencias, donde el tamaño de la página es de 100 bytes:

0100, 0432, 0101, 0612, 0102, 0103, 0104, 0101, 0611, 0102, 0103, 0104, 0101, 0610, 0102, 0103, 0104, 0101, 0609, 0102, 0105

Cadena de referencias:

- Sólo nos interesa el número de página.
- Si se referencia una página **p,** las referencias inmediatamente sucesivas a esa página nunca causarán error de página.
- Cadena de referencia simplificada:

  1, 4, 1, 6, 1, 6, 1, 6, 1, 6, 1

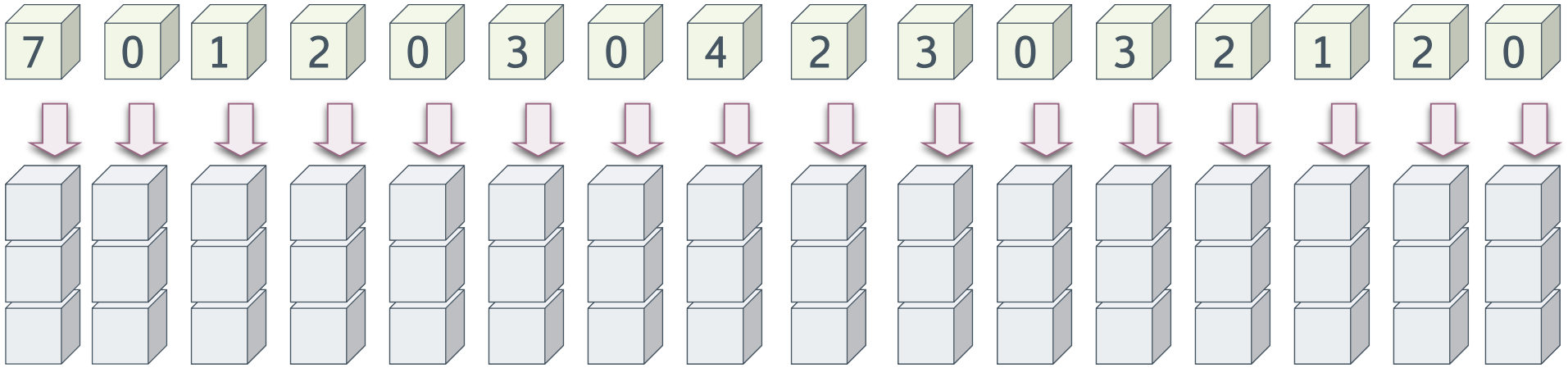
  - Con **tres marcos** habrá **3 errores** y con **un marco** habrá **11 errores**.

- Si el número de marcos aumenta, en general, el número de errores de página disminuye.

Algoritmos de Reemplazo de páginas
Óptimo

- Se reemplaza la página que va a tardar más tiempo en ser usada.
- Genera la tasa de errores más baja.
- Algoritmo imposible de implementar, puesto que requiere que el SO tenga un conocimiento exacto de los sucesos futuros.
- Sirve como un estándar con el que comparar los otros algoritmos.

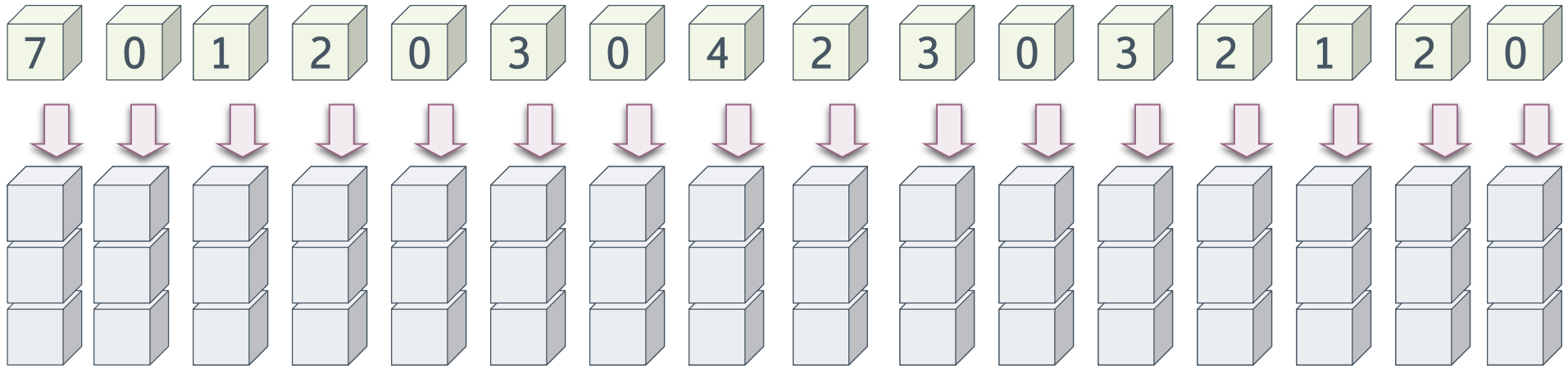
Algoritmos de Reemplazo de páginas Óptimo



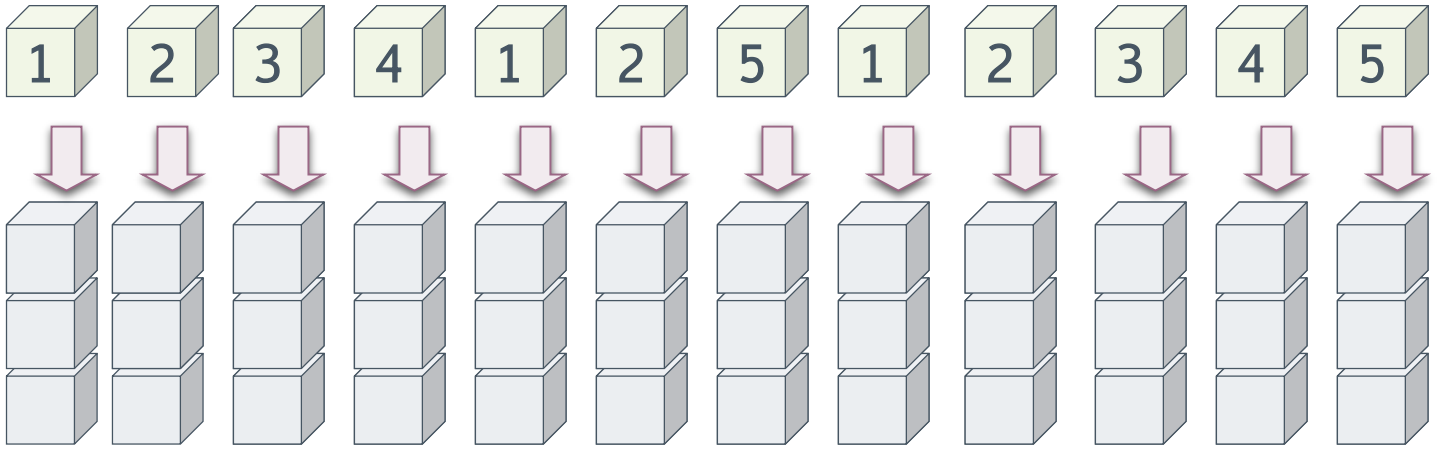
Algoritmos de Reemplazo de páginas FIFO (first-in, first-out)

- Es el más sencillo de entender e implementar.
- Sustituye la página que lleva más tiempo en memoria.
- Inconvenientes:
  - **Rendimiento pobre**. Páginas frecuentemente usadas pueden ser sustituidas.
  - Se puede producir la **Anomalía de Belady**: Aumenta el número de errores de página al aumentar el número de marcos.

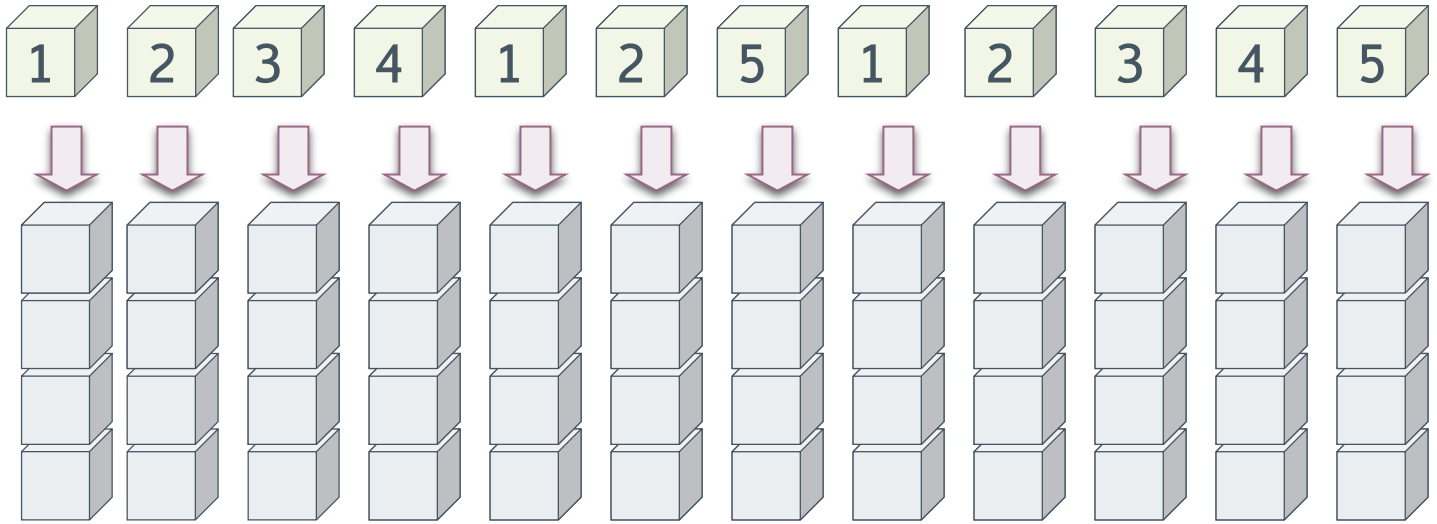
Algoritmos de Reemplazo de páginas FIFO



Algoritmos de Reemplazo de páginas FIFO



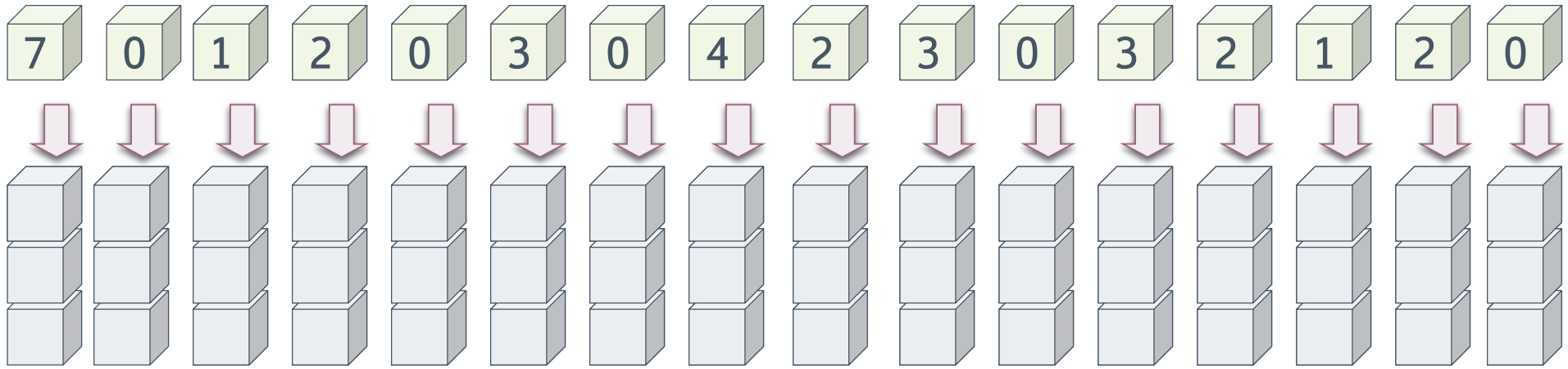
Algoritmos de Reemplazo de páginas FIFO Anomalía de Belady



Algoritmos de Reemplazo de páginas LRU

- Algoritmo de aproximación al reemplazo óptimo.
- Sustituye la página menos usada en el pasado inmediato, es decir, debería de ser la página con menor probabilidad de ser referenciada en un futuro cercano.
- Carece de la anomalía de Belady.
- La implementación requiere de hardware adicional:
  - LRU con reloj contador.
  - LRU con pila.

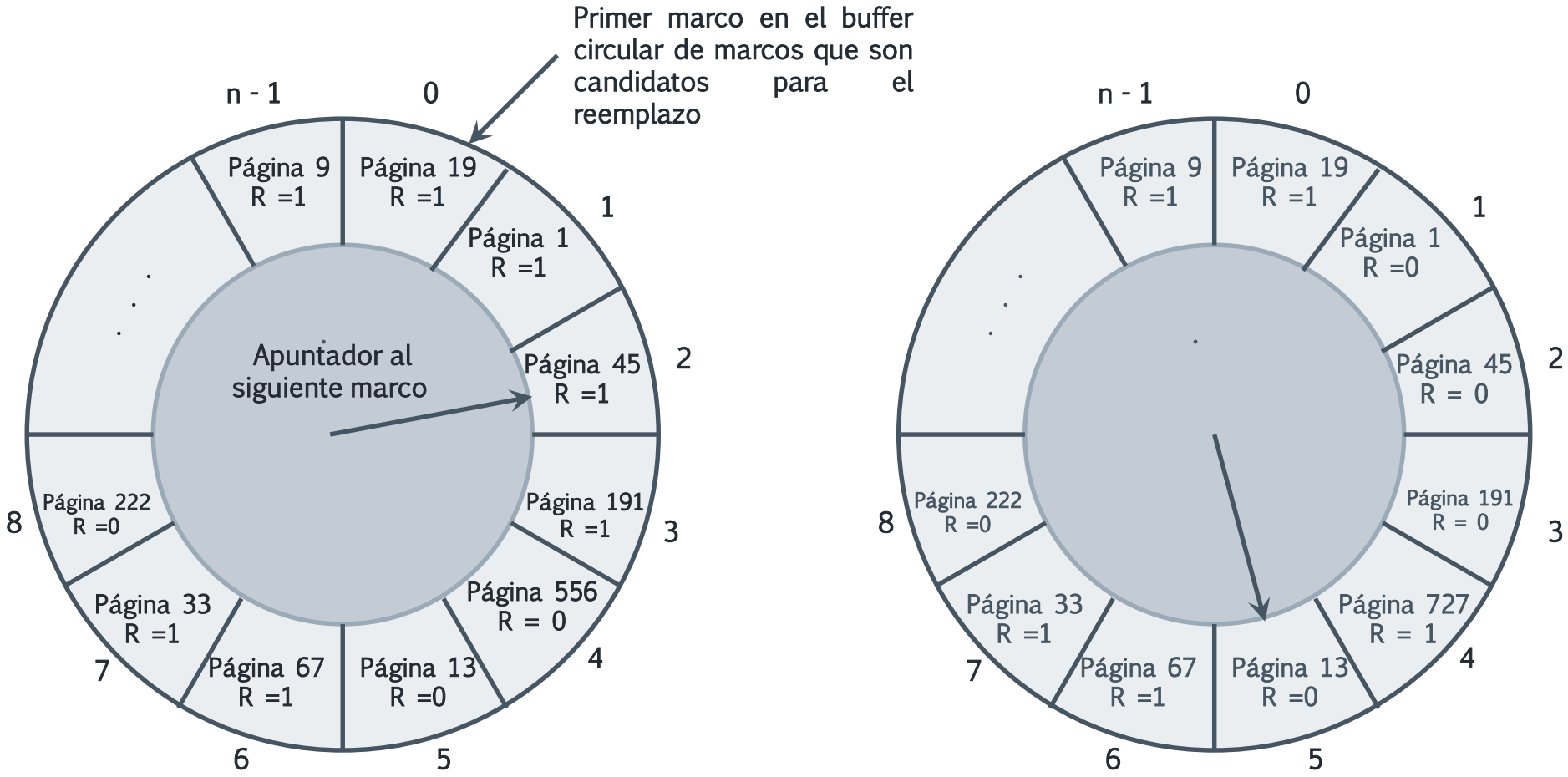
Algoritmos de Reemplazo de páginas LRU



Algoritmos de Reemplazo de páginas
Reloj Global

- Algoritmo de aproximación al LRU.
- Se requiere:
  - 1 bit de referencia (R, inicialmente a 0).
  - Marcos candidatos en lista circular.
  - Un apuntador (manecilla del reloj).
- A partir de la posición actual de la manecilla, la acción que se realiza depende del bit R:
  - Si la página tiene R=0, retira la página de la memoria.
  - Si la página tiene R=1, se pone a R=0 (se le da una segunda oportunidad) y avanza la manecilla.
  - Se sustituye la primera página que encontramos con el R=0. Se avanza la manecilla.

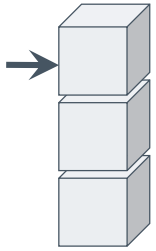
Algoritmos de Reemplazo de páginas Reloj Global



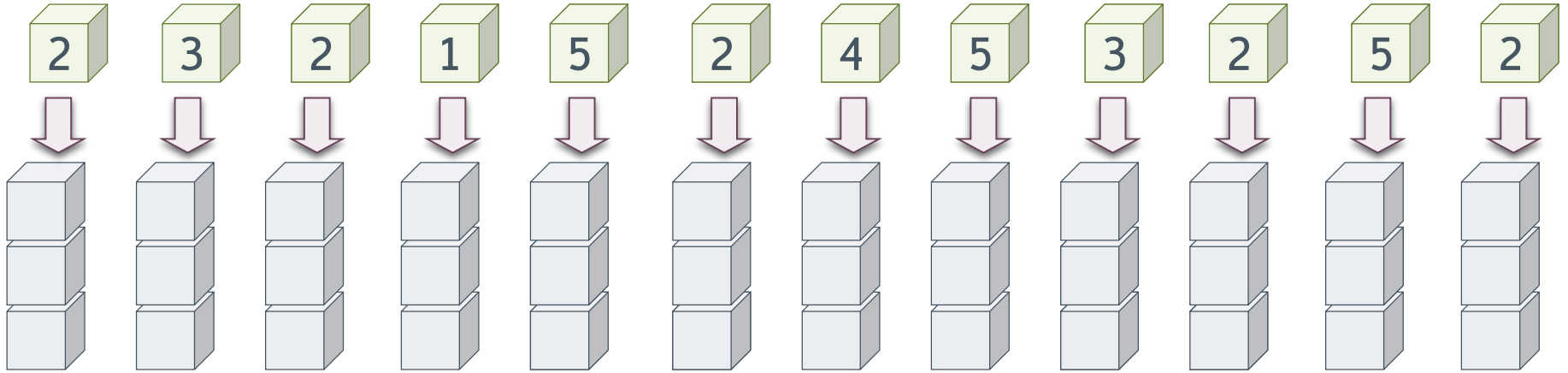
(a) Estado del buffer justo antes del reemplazo de página

(b) Estado del buffer justo después del siguiente reemplazo de página

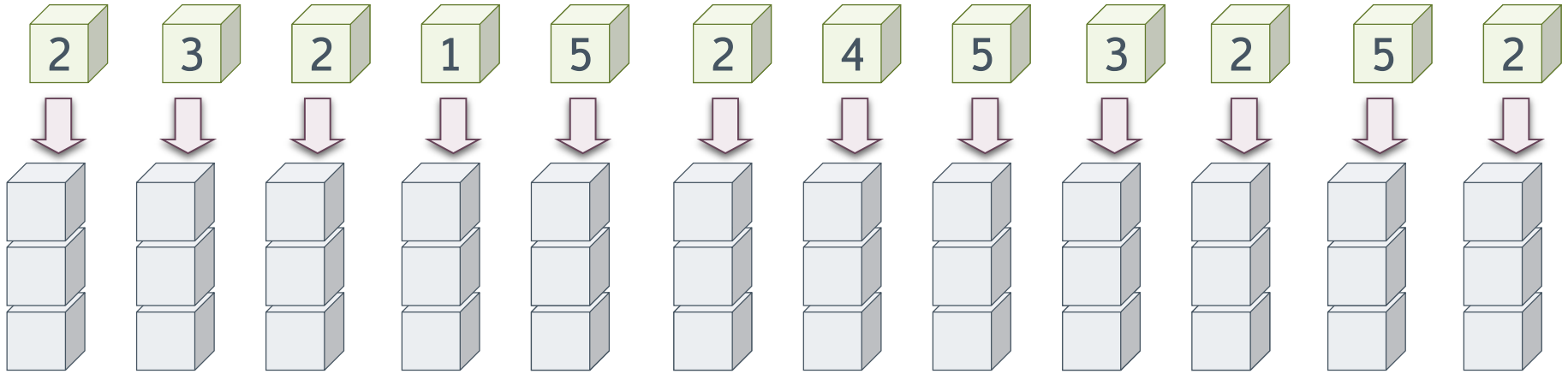
Algoritmos de Reemplazo de páginas Reloj Global



- La presencia del asterisco indica que el bit de uso correspondiente es igual a 1.
- La flecha indica la posición actual del apuntador.



Algoritmos de Reemplazo de páginas Reloj Global



Ing. Yesenia Carrera Fournier

sofiunam at gmail dot com