SISTEMAS OPERATIVOS

INGENIERÍA CIVIL INFORMÁTICA

GONZALO CARREÑO

GONZALOCARRENOB@GMAIL.COM

Gestión de memoria

La gestión de memoria física conlleva la realización de tres políticas:

- Política de búsqueda
- Política de ubicación
- Política de reemplazo

Política de búsqueda

Establece cuando se debe cargar una pagina en memoria principal.

Hay dos alternativas:

- Paginación por demanda: se trae a la memoria principal la pagina a la que se hace referencia.
- Paginación previa: Cuando además de la pagina que se referencia se traen una o dos paginas mas distintas a la demandada.

Política de ubicación

En paginación pura o combinada la política de ubicación es trivial, la pagina demandada se ubicara en una zona de memoria principal que este libre.

Política de reemplazo

Cuando no hay mas espacio en memoria física y hay que traer la pagina que ha sido demandada entonces, previamente a traerla, hay que desalojar una pagina de la memoria principal o física y de este procedimiento se encarga la política de reemplazo.

Si elegimos la pagina solamente de las paginas del proceso (monoprogramado) o de todas las paginas del sistema (Multiprogramado).

¿Qué pagina elegimos?. La idea es elegir la pagina que con menor probabilidad va a ser referenciada en el futuro.

Sistema operativo

El responsable de gestionar las tres políticas es el sistema operativo.

Mientras que el responsable de traducir las direcciones virtuales a direcciones reales o físicas es el hardware del procesador, concretamente la unidad de manejo de memoria (MMU).

Cuando se produce un fallo de pagina, se aborda la ejecución del programa y se pasa el control al sistema operativo.

Memoria virtual

Introducción

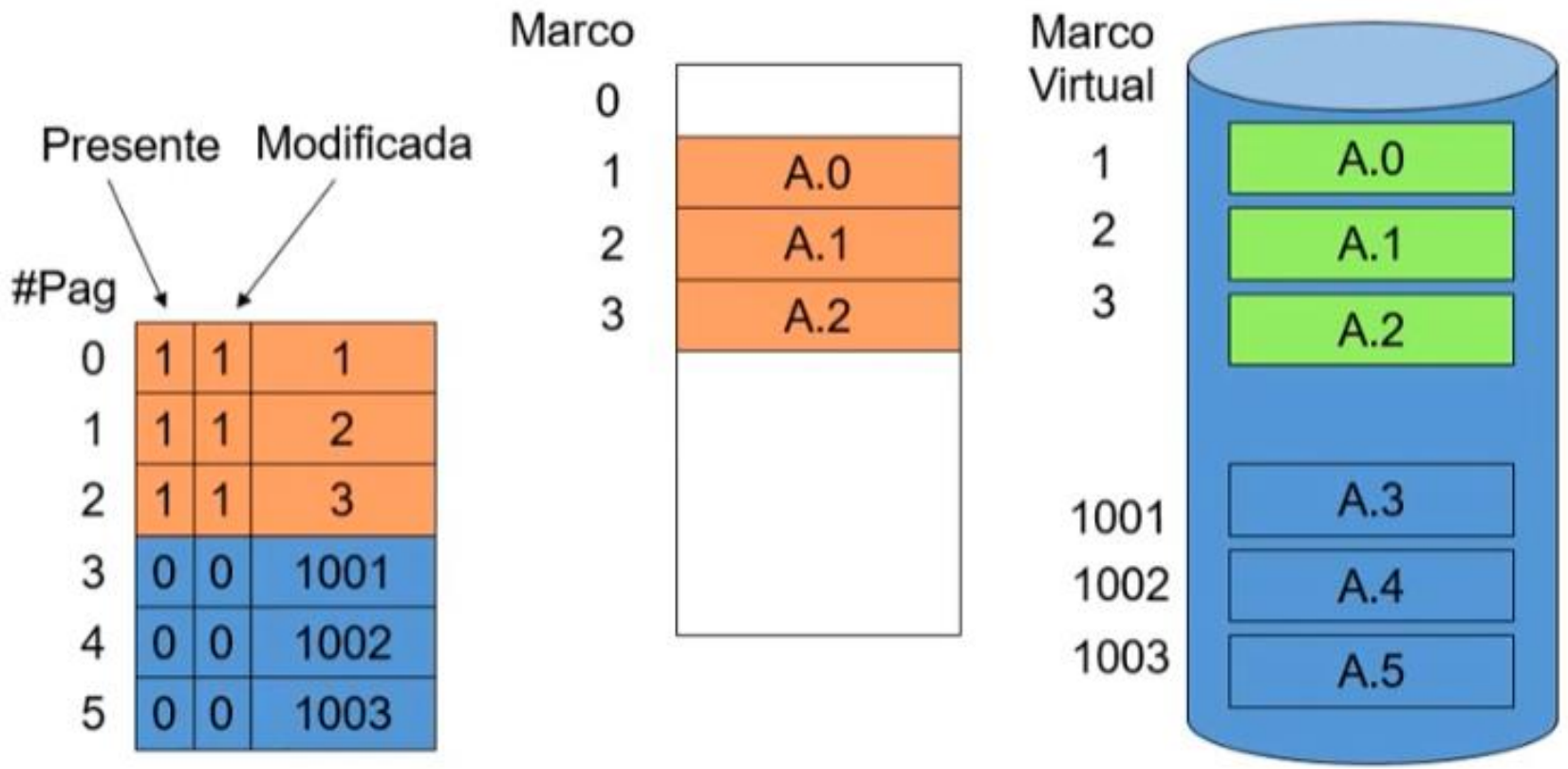
- Memoria virtual – separación de la memoria lógica de usuario de la memoria física.
- Solo una parte del programa se necesita que esté en memoria para su ejecución.
- El espacio de direcciones lógicos puede ser más grande que el espacio de direcciones físico.
- Se necesita permitir que las paginas salgan y entren. Memoria a disco – disco a memoria.

Introducción

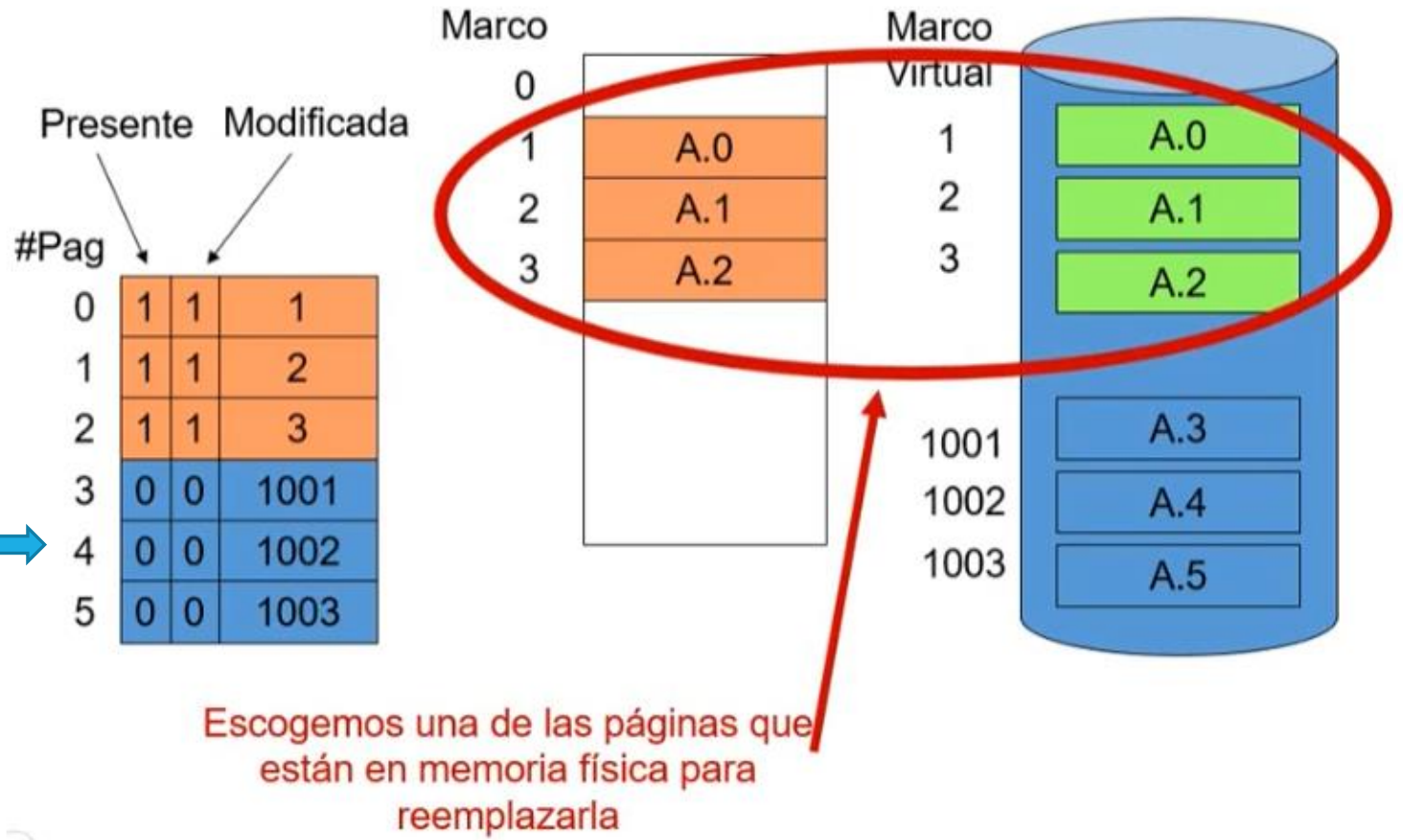
La memoria virtual puede implementarse vía:

- Paginación por demanda
- Segmentación por demanda

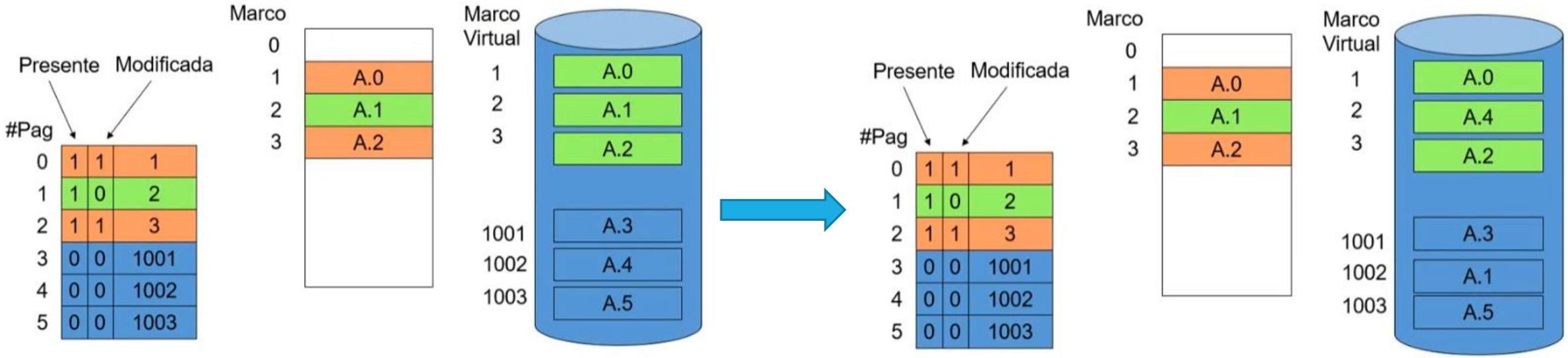
Memoria virtual

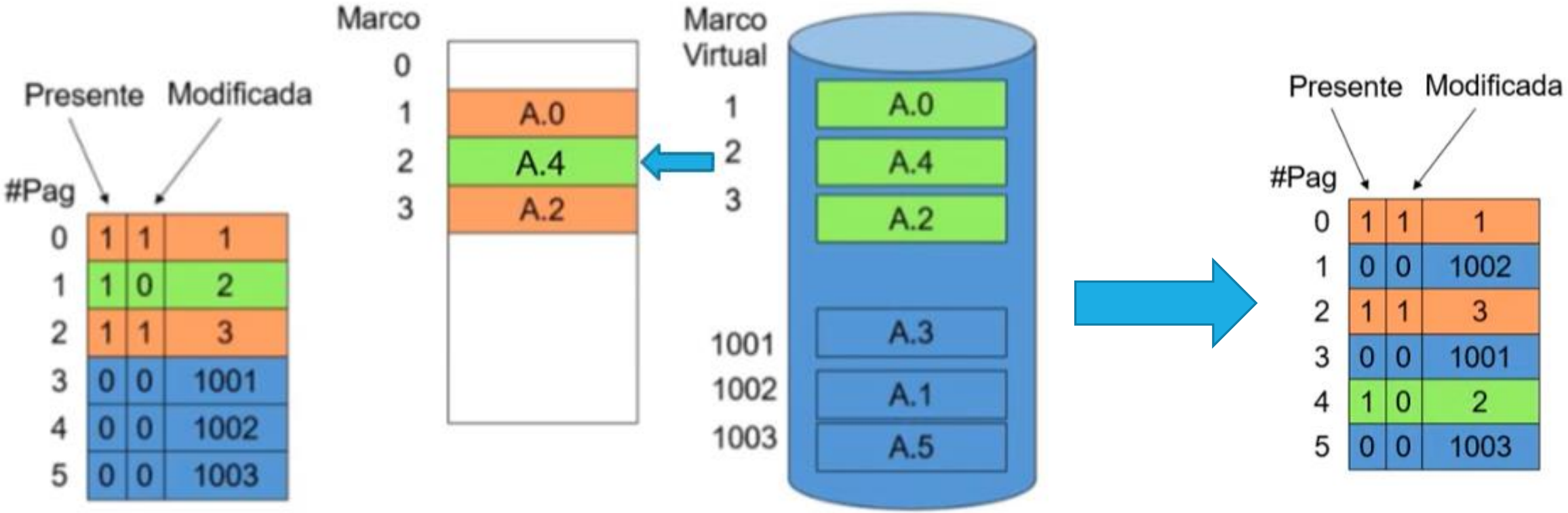


Se referencía una dirección que corresponde a la página 4



Memoria virtual




Presente
Modificada
#Pag
0 1 1 1
1 1 0 2
2 1 1 3
3 0 0 1001
4 0 0 1002
5 0 0 1003
Marco
0
1
2
3
A.0
A.4
A.2
Marco Virtual
1
2
3
1001
1002
1003
A.0
A.4
A.2
A.3
A.1
A.5
Presente
Modificada
#Pag
0 1 1 1
1 0 0 1002
2 1 1 3
3 0 0 1001
4 1 0 2
5 0 0 1003

Paginación por demanda

Traer una pagina a memoria solo cuando se necesita.

Como consecuencia de lo anterior se va a requerir:

- Menos E/S necesaria
- Menos memoria requerida
- Respuesta más rápida
- Más usuarios

Se requiere una pagina cuando se referencia.

- Abortar si hay una referencia invalida.
- Si la pagina no esta en memoria vamos a traerla a memoria

Bit de presente

En cada entrada de la tabla de páginas hay un bit de presente

- 1=en memoria
- 0=no en memoria

Inicialmente se establece en 0 en todas las entradas

Durante la traducción de direcciones, si el bit de presente es 0, entonces hay un **fallo de página**

Fallo de página

La primera referencia a una pagina siempre ocasionará una excepción llamada fallo de pagina.

El SO busca en otra tabla para decidir:

- Si es una referencia invalida, abortar.
- Solo es que no esta en memoria

Obtener un marco vacío

Guardar la pagina en un marco

Restaurar tablas, bit de presente =1

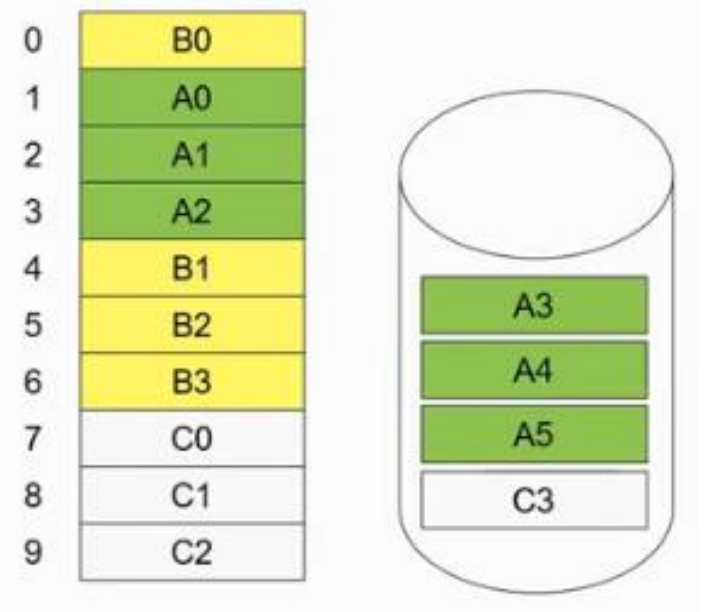
Instrucción de reinicio: LRU (Least Recently Used)

¿Qué sucede si no hay marcos disponibles?

Reemplazo de páginas

Encontrar una pagina en memoria que no se este utilizando para sacarla

Página	Marco	Presente
0	1	1
1	2	1
2	3	1
3	Marco virtual	0
4	Marco virtual	0
5	Marco virtual	0

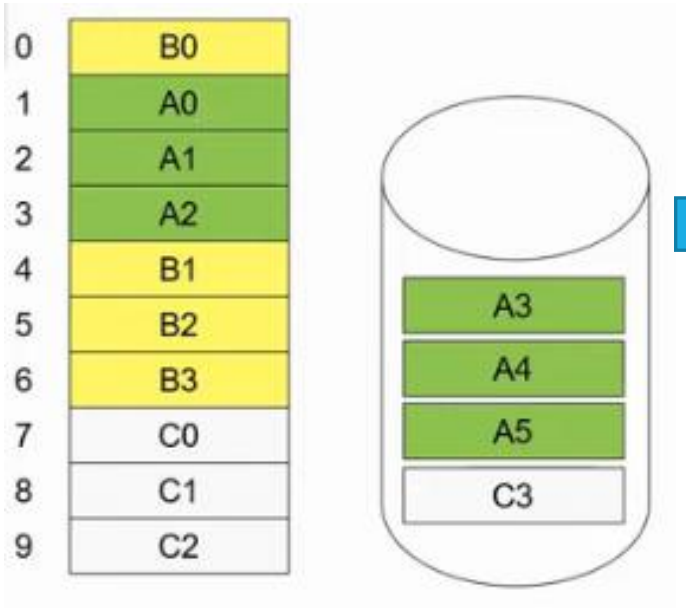


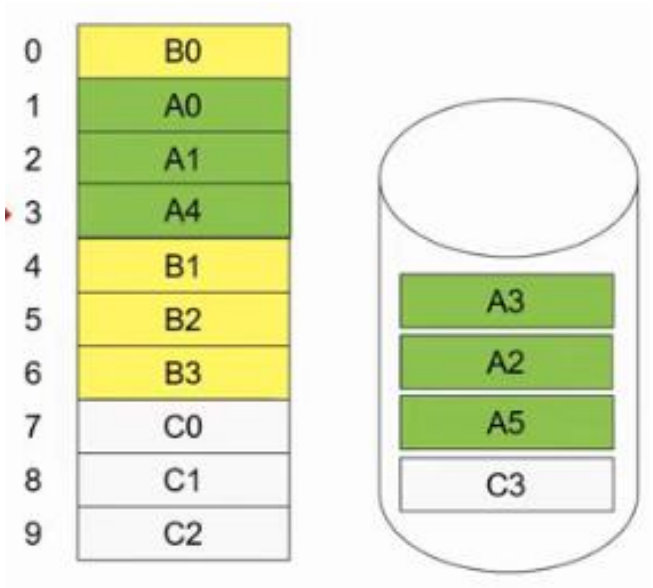
¿Qué sucede si no hay marcos disponibles?

Algoritmos de reemplazo

Rendimiento

Se necesita un algoritmo que nos de como resultado un mínimo de fallos de página, la misa página podría ser traía muchas veces a memoria.





Página	Marco	Presente
0	1	1
1	2	1
2	Marco virtual	0
3	Marco virtual	0
4	3	1
5	Marco virtual	0

Rendimiento en paginación por demanda

- Tasa de fallos de página $0 \leq p \leq 1.0$
  - if $p = 0$ no fallos de página
  - if $p = 1$, cada referencia es un fallo

- Tiempo de Acceso Efectivo (EAT)
  - EAT = $(1 - p)$ x tiempo_accesos_a_memoria +
    p (sobrecarga del fallo de página +
    [escribir página a disco] +
    cargar página del disco +
    sobrecarga por reinicio)

Ejemplo

Tiempo de acceso a memoria = 10 nanosegundos

50% de las veces la pagina que se esta reemplazando fue modificada y por lo tanto necesita escribirse en disco.

Tiempo de intercambio de paginas = 10 milisegundos

- EAT = (1-p)x10+p(15000000)
- EAT = 10-10p+15000000p

Reemplazo de páginas

Prevenir la sobreasignación de memoria modificando la rutina de fallos de pagina para incluir el reemplazo de páginas.

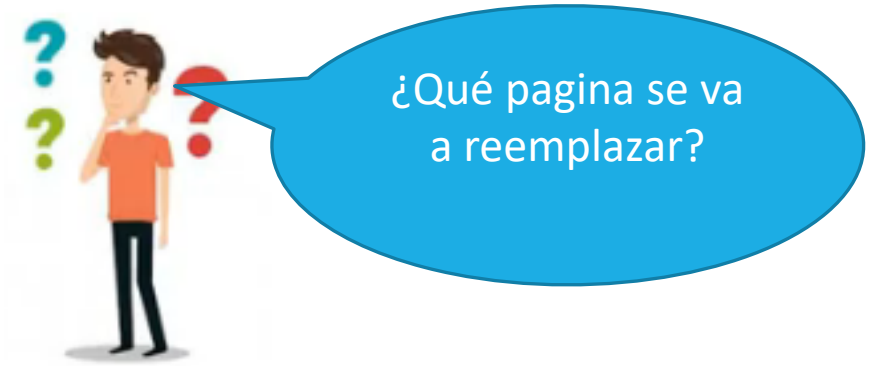
Bit de modificado

- Reducir la sobrecarga de transferencias de pagina.
- Solo las paginas modificadas se escriben en disco

El reemplazo de paginas completa la separación entre memoria lógica y memoria física.

- Memoria virtual mas grande que la memoria física

Políticas de reemplazo



Tiene que ser la que tenga una menor posibilidad de ser referenciada en un futuro cercano.

La mayoría de las políticas intentan predecir el comportamiento futuro en función del comportamiento pasado.

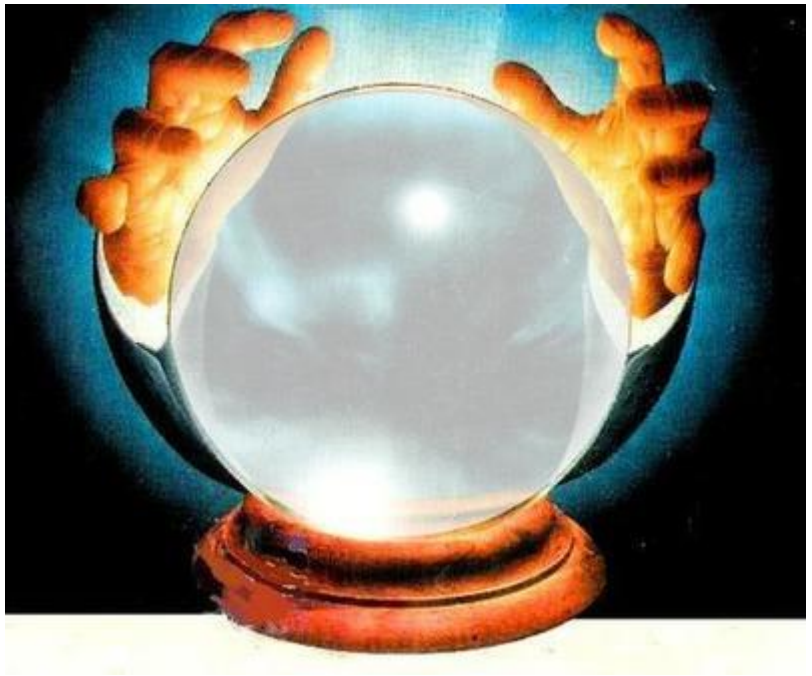
Políticas de reemplazo

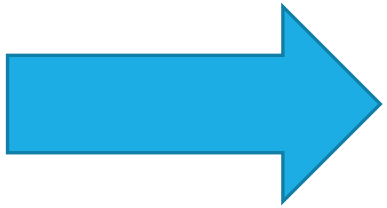
Bloqueo de marcos:

- Hay paginas que no deben salir.
- Paginas del núcleo del sistema operativo.
- Estructuras de control.
- Buffers de E/S.
- El bloqueo se consigue asociando un bit de bloqueo a cada marco.

Algoritmos básicos de reemplazo

- Política óptima:
  - Selecciona para reemplazar la página que esperará más tiempo hasta que se produzca la referencia siguiente.





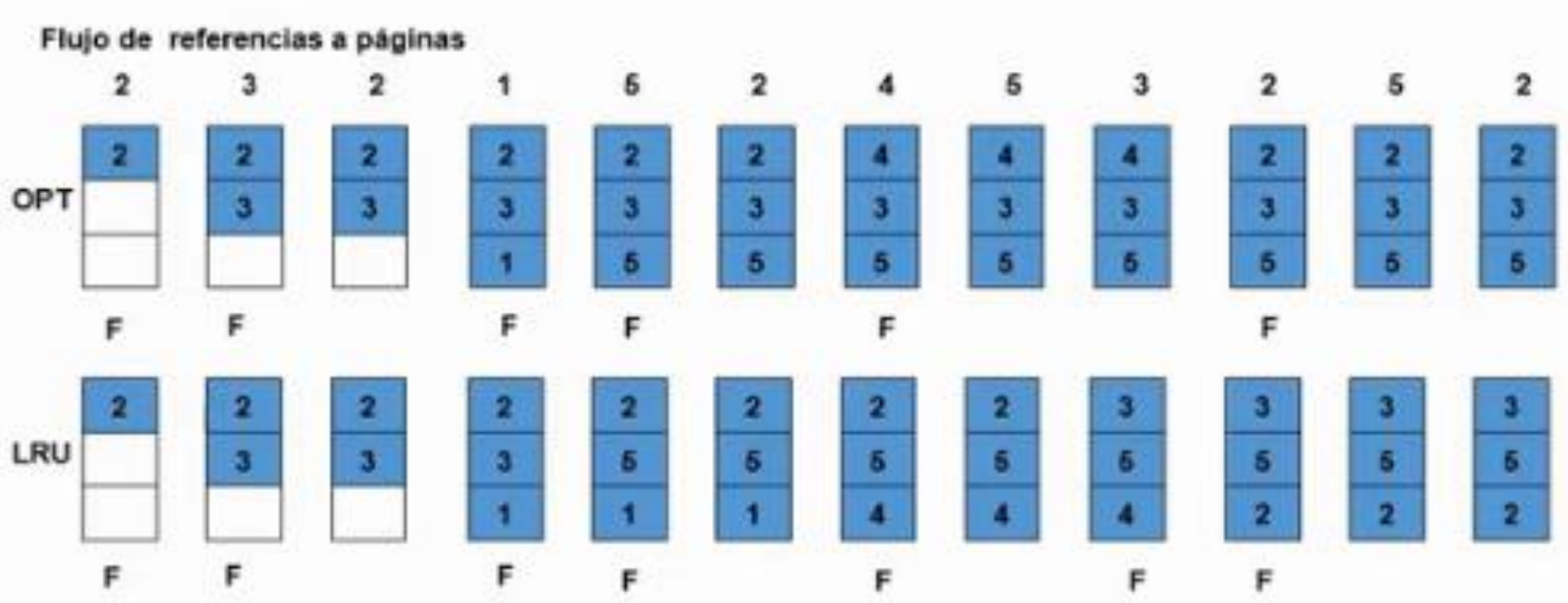
Algoritmos básicos de reemplazo

Política del uso menos reciente (LRU):

- Reemplaza la pagina de memoria que no ha sido referenciada desde hace mas tiempo.
- Esta seria la pagina con menor probabilidad de ser referenciada en un futuro cercano.
- Etiquetar cada pagina con el momento de su ultima referencia.

La política de la usada hace más tiempo

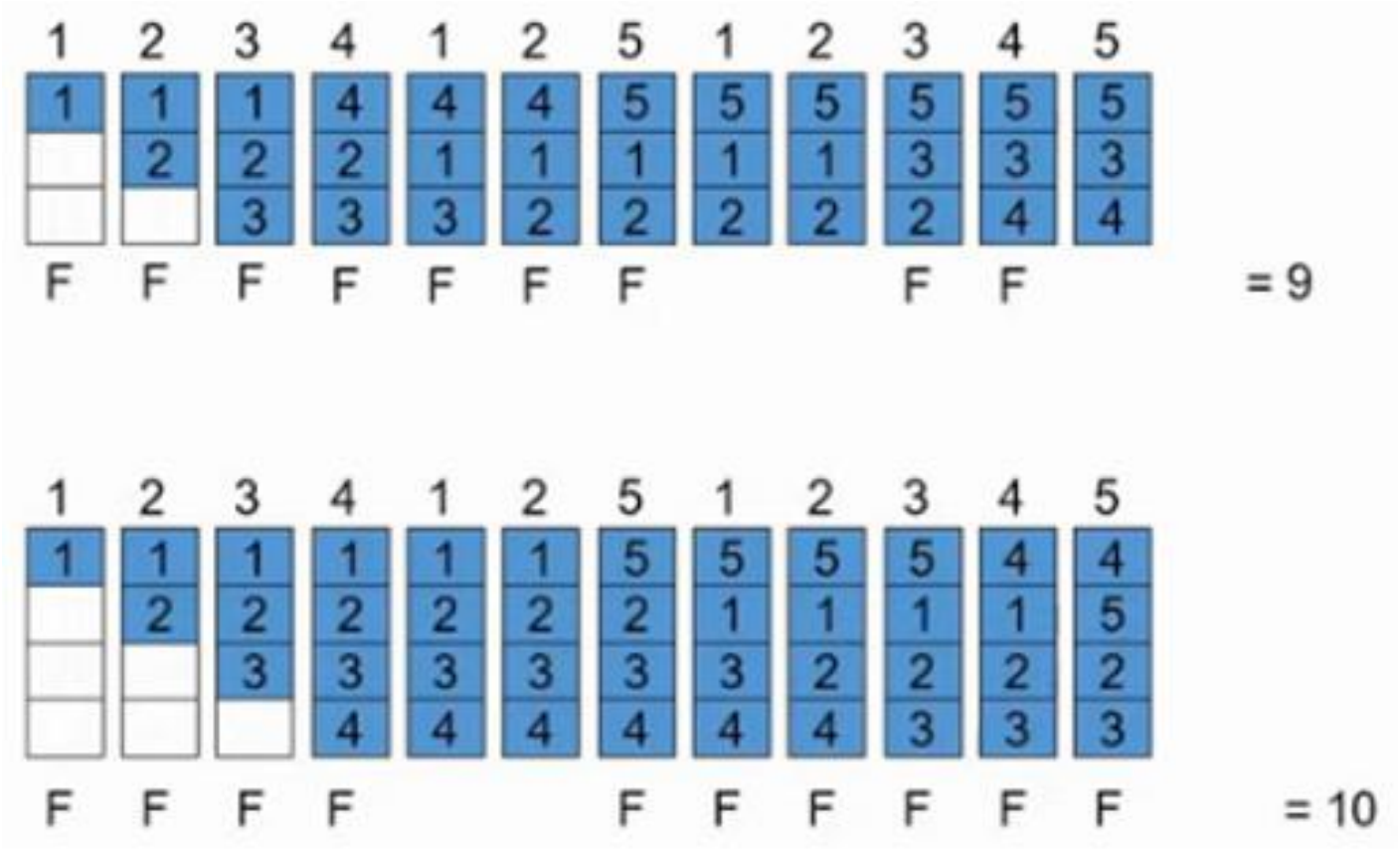
Ejemplo: un proceso que hace referencia a 5 paginas con una asignación constante de 3 marcos para el proceso.



Algoritmos básicos de reemplazo

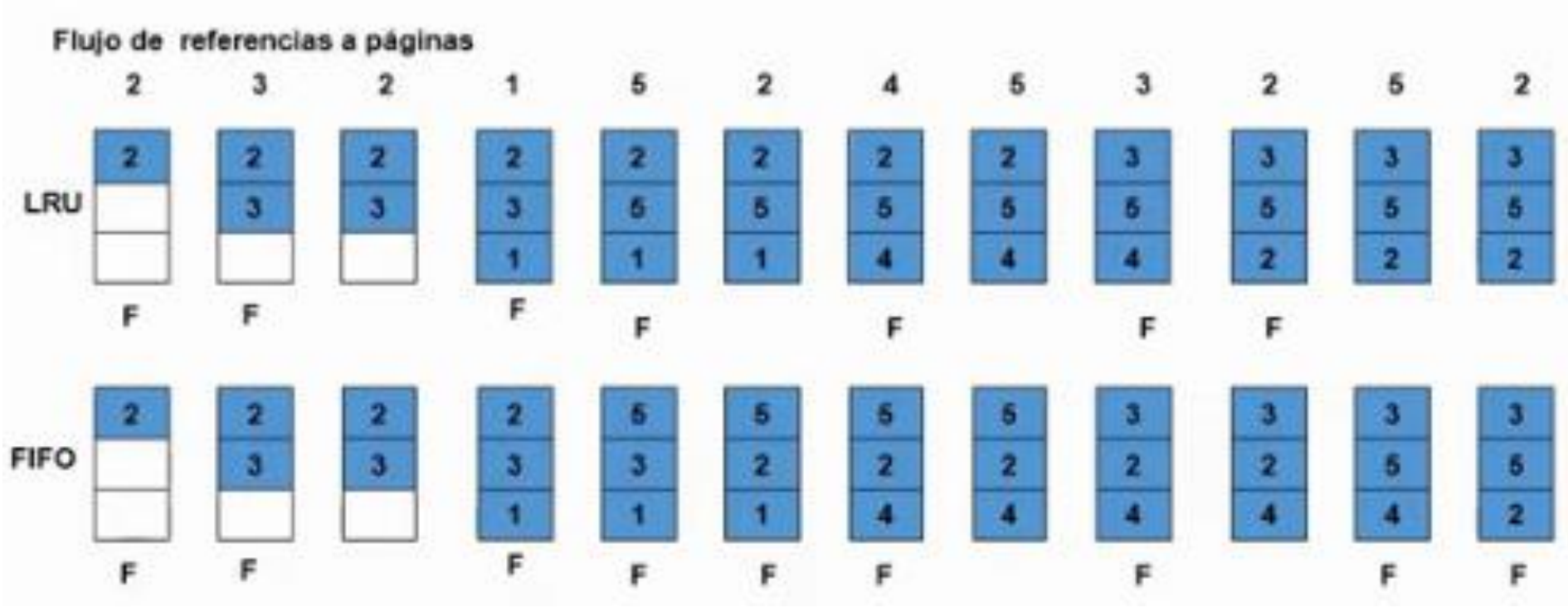
- Primera en salir (FIFO):
  - Los marcos de un proceso como un buffer circular.
  - Muy sencilla de implementar.
  - Se reemplaza la página que ha estado más tiempo en la memoria.
  - Estas páginas pueden necesitarse de nuevo y en un plazo de tiempo corto.

Anomalía de Belady (FIFO)

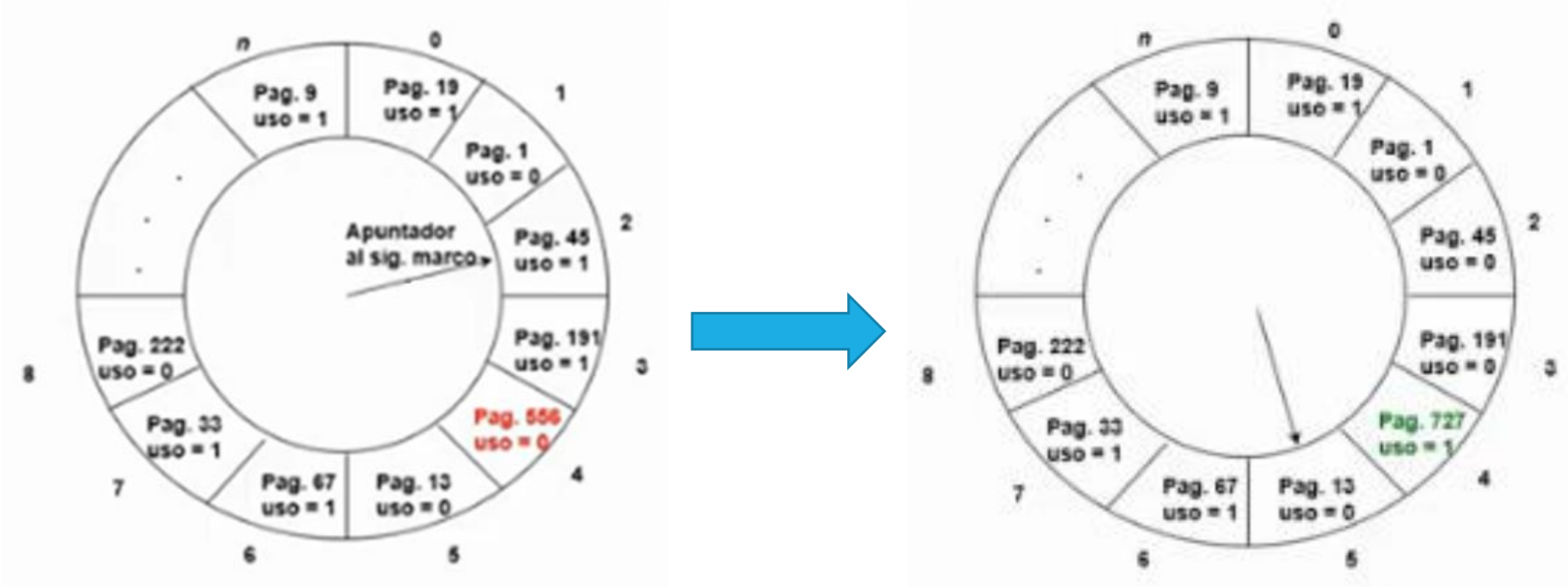


Comparación de FIFO con LRU

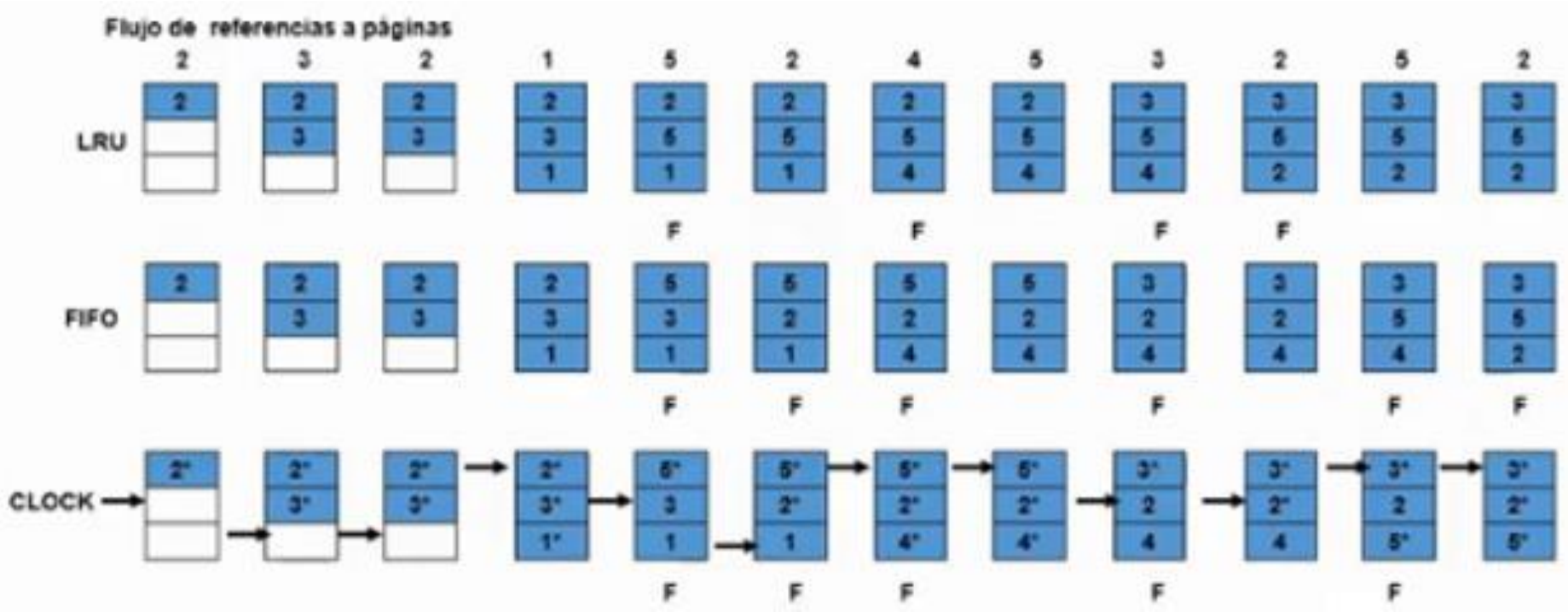
El rendimiento de FIFO es relativamente pobre.



La política de reloj: ejemplo

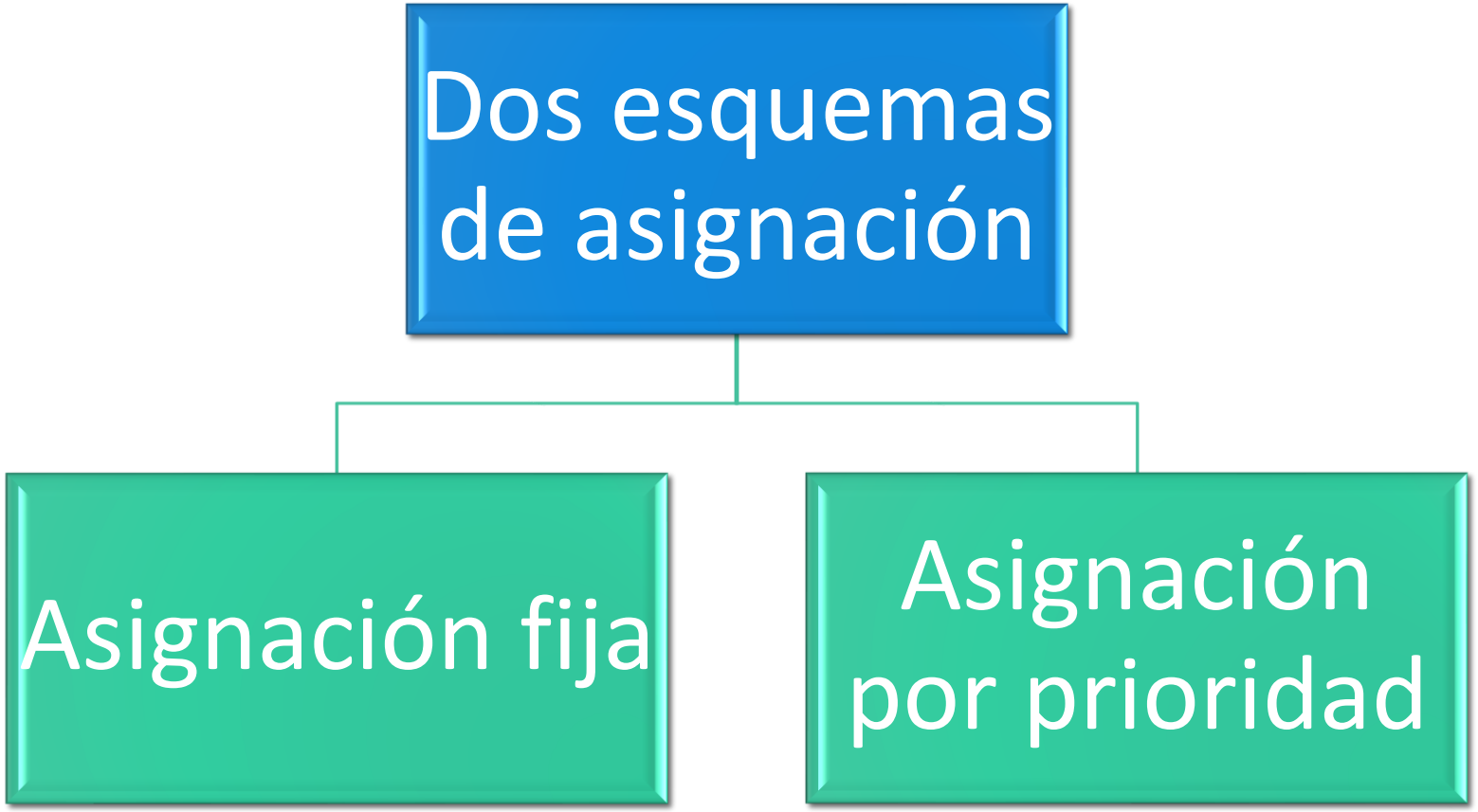


Comparación del reloj con FIFO y LRU



El asterisco indica que el bit de uso correspondiente está en 1
El reloj proteje las páginas frecuentemente referenciadas poniendo el bit de uso en 1 en cada referencia

Asignación de marcos



Asignación fija

Asignación equitativa:

- Ejemplo: Si son 100 marcos y 5 procesos, dale a cada uno 20 marcos.

Asignación proporcional

- Asignar de acuerdo al tamaño del proceso

s_i = tamaño del proceso p_i
$S = \Sigma s_i$
m = número total de marcos
a_i = asignación para $p_i = (s_i / S) \times m$

$$m = 64$$
$$s_1 = 10$$
$$s_2 = 127$$
$$a_1 = \frac{10}{137} \times 64 \approx 5$$
$$a_2 = \frac{127}{137} \times 64 \approx 59$$

Asignación por prioridad

Usar un esquema de asignación proporcional usando prioridades en vez de tamaño.

Si el proceso P_j ocasiona un fallo de página

- Selecciona uno de sus marcos para reemplazarlo
- Selecciona a un marco de un proceso con menor prioridad

Reemplazo global vs local

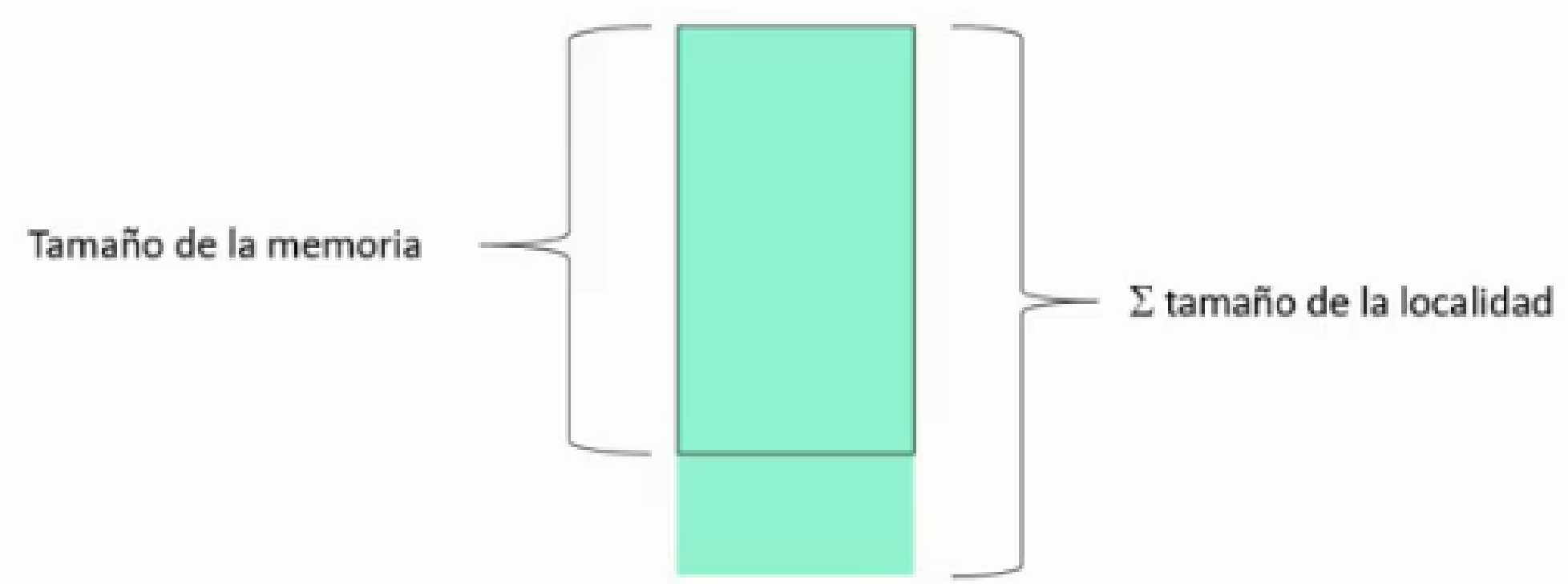
Trashing (Hiperpaginación)

Si un proceso no tiene suficientes marcos

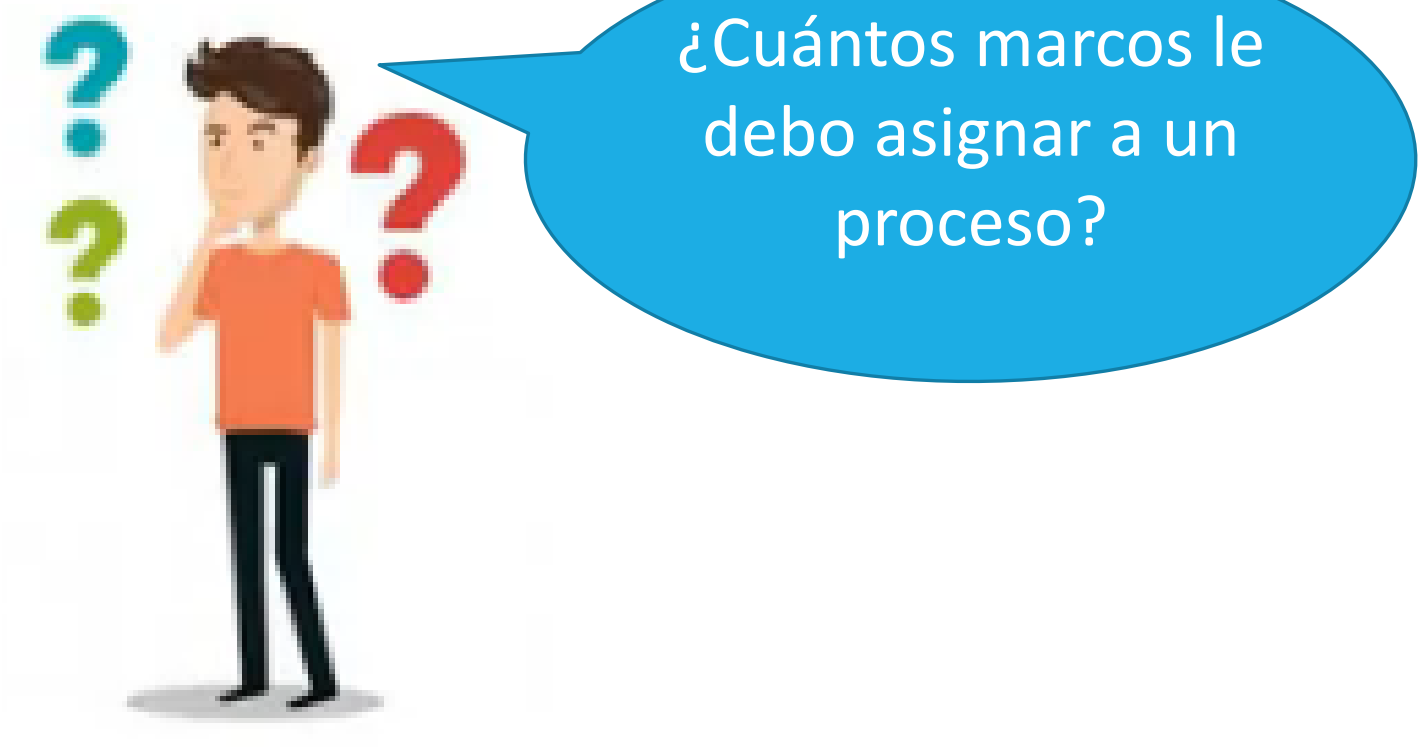
- La tasa de fallos de pagina es muy alta: mucha E/S
- Baja utilización del CPU
- El sistema Operativo cree que es necesario incrementar el grado de multiprogramación
- Se agrega otro proceso al sistema.

Diagrama de hiperpaginación

¿Por qué ocurre la hiperpaginación?



Modelo del conjunto residente



Modelo del conjunto residente

- $\Delta \equiv$ ventana del conjunto residente
  - Un número fijo de referencias de páginas
  - Ejemplo: 10,000 instrucciones

- WSS_i (conjunto residente del proceso P_i) =
  - Total de páginas referenciadas en la más reciente Δ
    - varía en el tiempo

si Δ es demasiado pequeña no abarcará la localidad entera.
si Δ es demasiado grande abarcará muchas localidades.
si $\Delta = \infty \Rightarrow$ abarcará todo el programa.

Modelo del conjunto residente

- $\boldsymbol{D} = \Sigma\ WSS_i$
  - Sumatoria del total de páginas referenciadas para todos los procesos
  - Demanda total de marcos

- si $\boldsymbol{D} > m \Rightarrow$ Hiperpaginación
  - Política: si D > m, entonces suspende uno de los procesos.

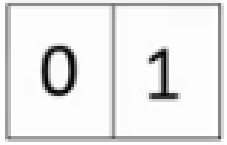
Monitoreo del junto residente

Aproximar con timer periódico + bit de referencia

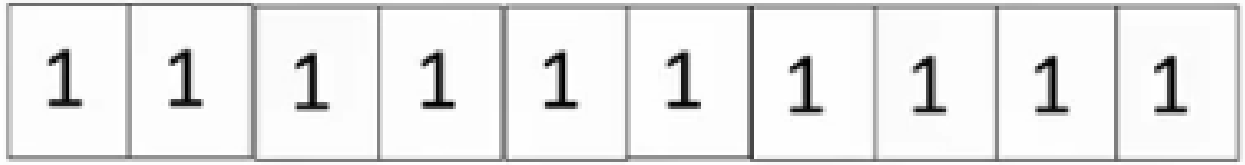
Ejemplo: Δ = 10,000

- Interrupción del timer cada 5000 unidades de tiempo
- Mantener en memoria 2 bits por cada pagina
- En la interrupción copiar bit de referencia y ponerlo en cero (shift)
- Si uno de los bit = 1 pagina pertenece a conjunto de trabajo.

¿Por qué no es preciso?

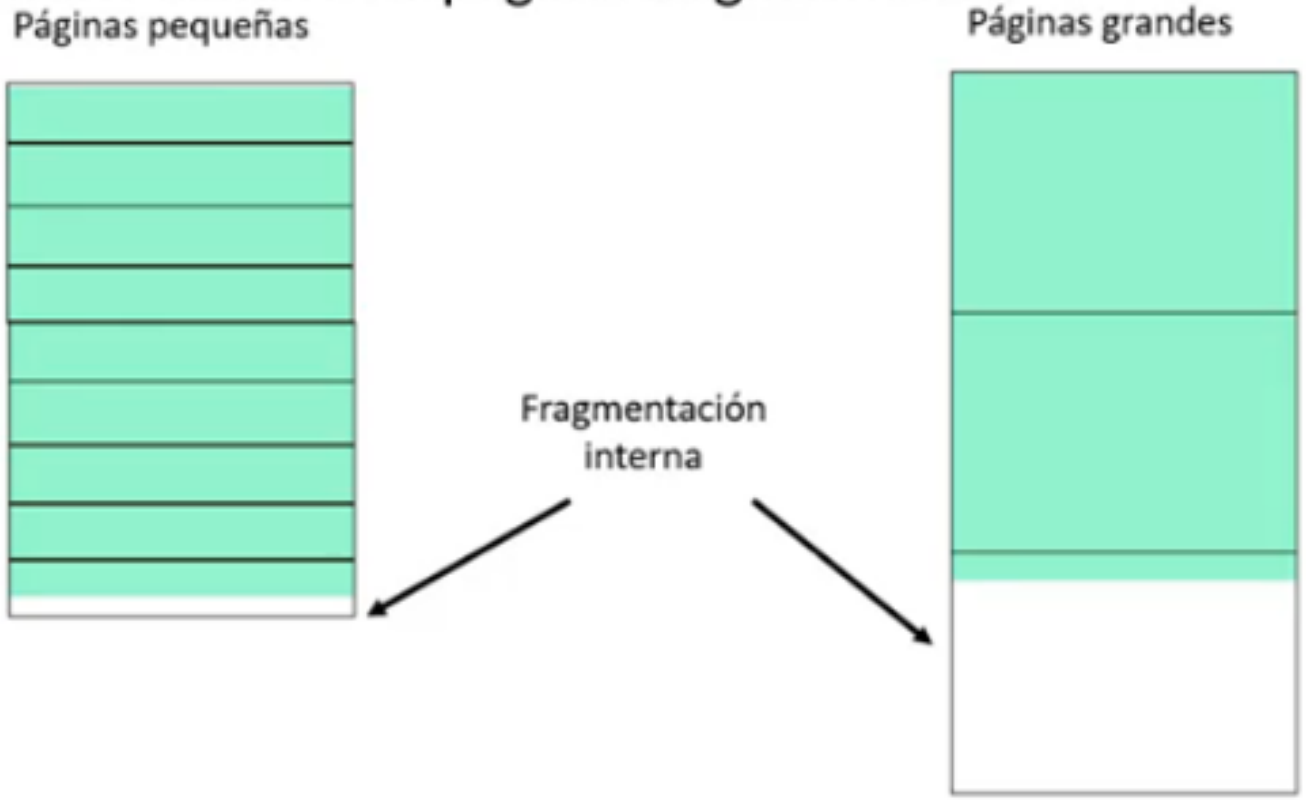


Mejora = 10 bits e interrupt cada 1000 unidades de tiempo



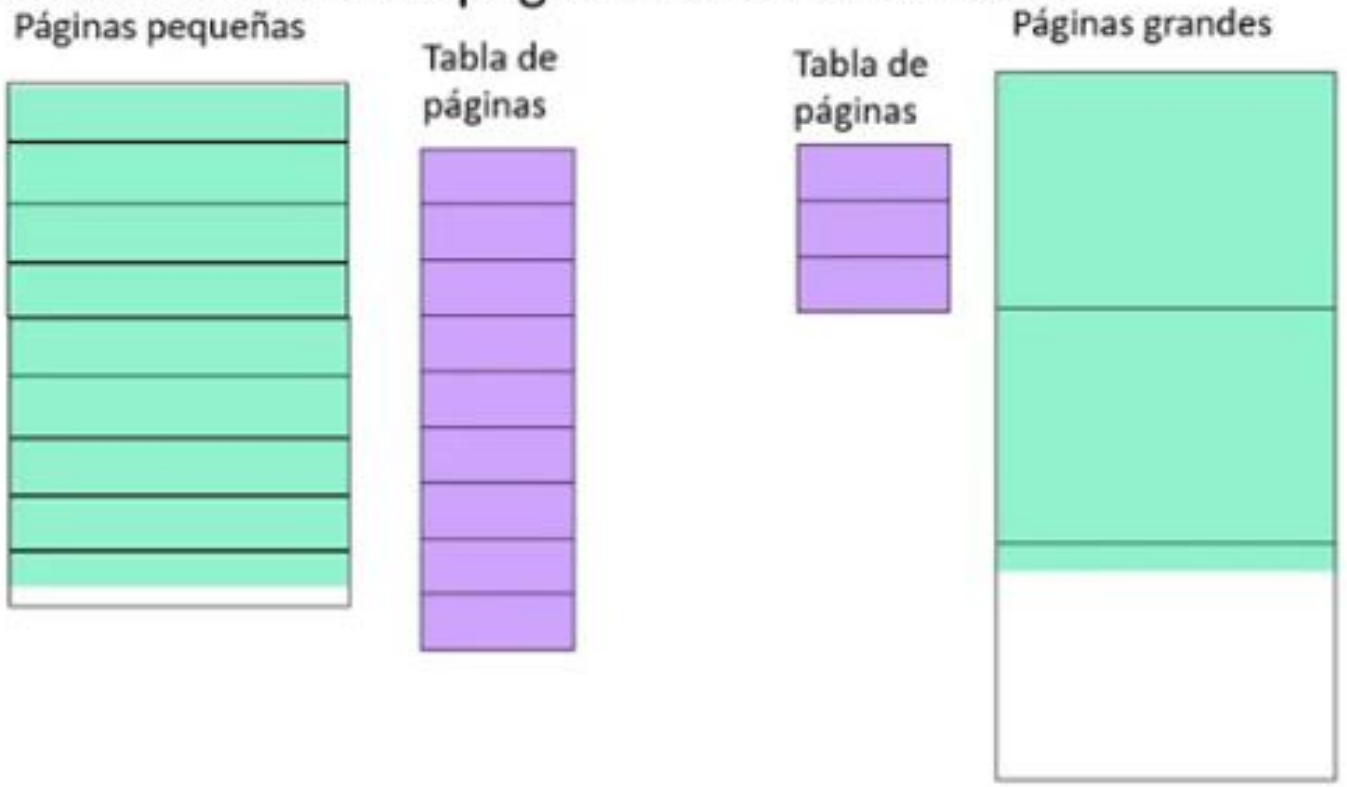
Otras consideraciones

- Selección del tamaño de página: fragmentación



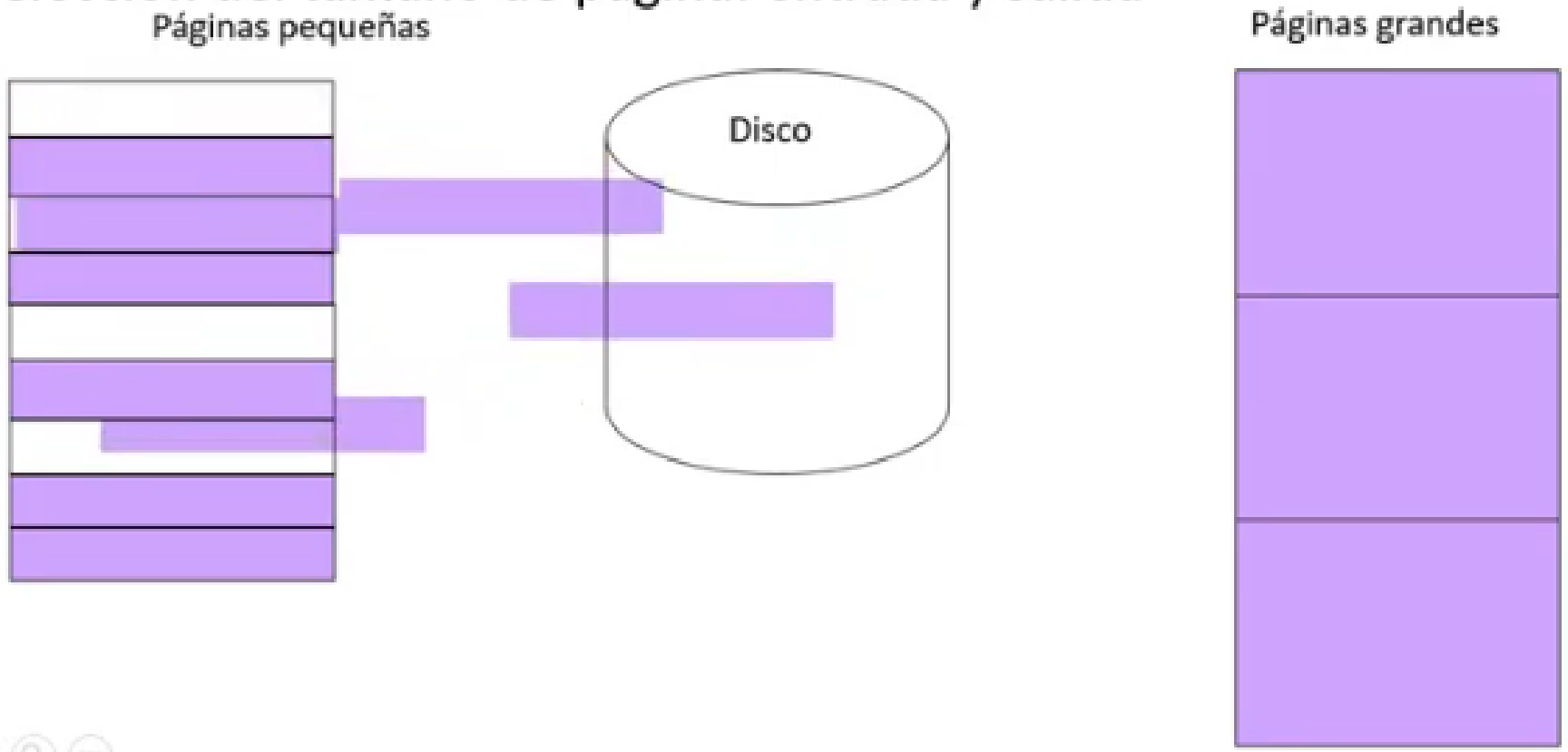
Otras consideraciones

- Selección del tamaño de página: tamaño de tabla

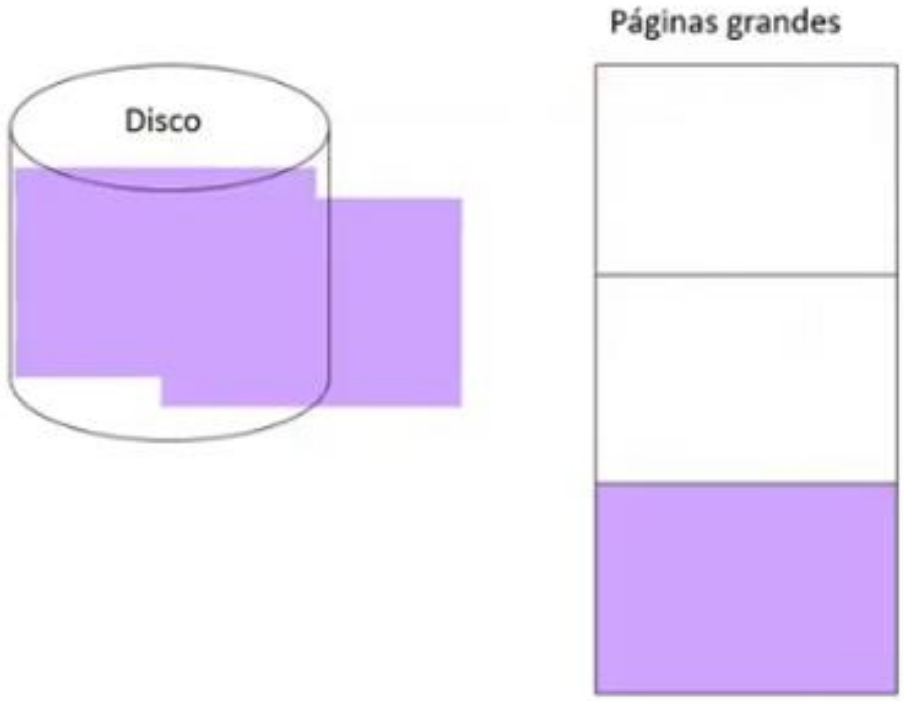


Otras consideraciones

- Selección del tamaño de página: entrada y salida

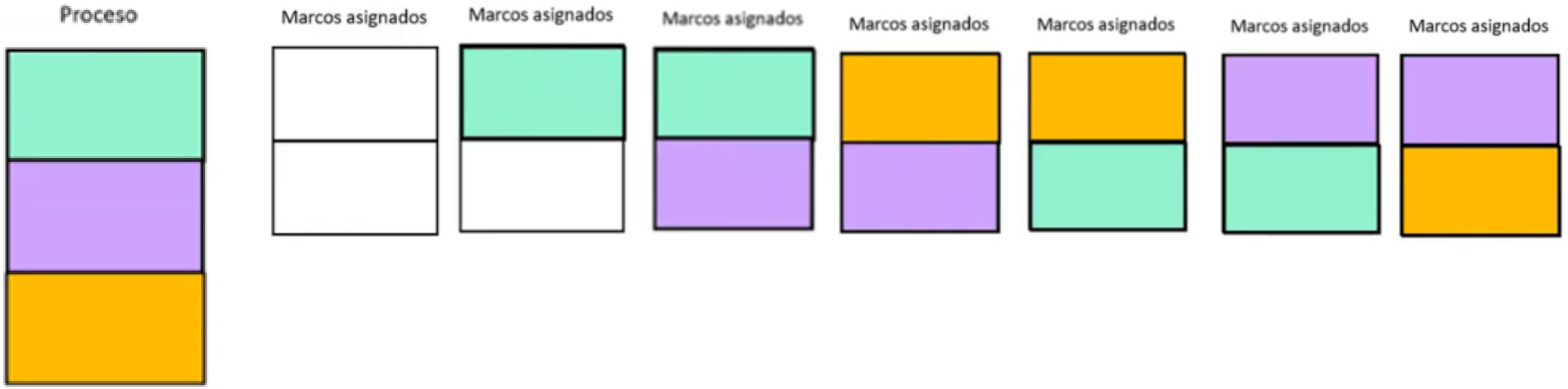


Otras consideraciones



Otras consideraciones

- Selección del tamaño de página: localidad



Otras consideraciones

- Selección del tamaño de página: localidad

Segmentación por demanda

Usada cuando no hay hardware suficiente para implementar paginación por demanda

OS/2 asigna memoria en segmentos, y cual lleva registro a través de los descriptores de segmento (tabla de segmentos)

El descriptor de segmentos contiene un bit presente para indicar cuando el segmento está en memoria

- Si el segmento está en memoria principal, el acceso continúa
- Si no está en memoria, hay un fallo de segmento.