# Gestión de la Memoria

SISTEMAS OPERATIVOS

PROF. PHD MIRELLA HERRERA

# Políticas de Gestión del SO

- **Políticas de Lectura**
  - Paginación por Demanda
  - Paginación Previa

- **Políticas de Ubicación**
  - Para sistemas con segmentación pura
  - Para paginación y segmentación/paginada

# Políticas de Gestión del SO

- **Políticas de Reemplazo**
  - Alcance Global
  - Alcance Local
  - Mecanismos de Reemplazo: Algoritmos de Sustitución

- **Políticas de Asignación**
  - Asignación Equitativa
  - Asignación Proporcional

# Políticas de Gestión del SO

- **Gestión del Conjunto Residente**
  - Asignación Fija con Alcance Global
  - Asignación Fija con Alcance Local
  - Asignación Variable con Alcance Global
  - Asignación Variable con Alcance Local

- **Políticas de Vaciado**
  - Vaciado por Demanda
  - Vaciado Previo

# Políticas de Gestión del SO

- **Estrategia del Conjunto de Trabajo**
  - Working Set
- **Frecuencia de Fallo de Página**
- **Políticas de Control de Carga**
  - Suspensión de Procesos

# Políticas de Gestión del SO

## Políticas de Lectura

Determina cuando una página debe traerse a memoria principal

- **Por Demanda**
  - Se trae una página a memoria principal sólo cuando se hace referencia a una posición en dicha página
  - Produce muchos fallos de página cuando un proceso se ejecuta por primera vez; pero debe disminuir a medida que se traigan a memoria más páginas (principio de cercanía)

# Políticas de Gestión del SO

## Políticas de Lectura

- **Previa**
  - Se cargan otras páginas distintas a la demandadas, debido a un fallo de página
  - El principio de cercanía sugiere que sea más eficaz traer páginas que residen contiguamente en disco
  - Eficiencia no establecida definitivamente cuando las páginas extras traídas, no son “frecuentemente” referenciadas

# Políticas de Gestión del SO

## Políticas de Ubicación

Determina dónde va a residir una parte de un proceso en memoria real

- **Ubicación en sistemas con segmentación pura:**
  - Primer-ajuste, próximo-ajuste, mejor-ajuste (relevante)

- **Ubicación en sistemas con paginación o segmentación/paginada:**
  - El hardware decide dónde poner la página (no relevante)

# Políticas de Gestión del SO

## Políticas de Reemplazo

Trata de la selección de la página a reemplazar en memoria principal, al momento de cargar una nueva página. Esto ocurre siempre que la memoria principal esté llena (ningún marco libre disponible)

- Ocurre frecuentemente debido a que el SO intenta traer a memoria principal tantos procesos como pueda, para aumentar el nivel de multiprogramación
- Algunos marcos que están bloqueados no pueden paginarse (paged out):
  - Mucho del kernel es mantenido en marcos bloqueados; así como también, las estructuras clave de control y buffers de I/O

# Políticas de Gestión del SO

## Políticas de Reemplazo

- Se desea reemplazar la página que tenga menos posibilidad de ser referenciada en otro momento
- Se debe evitar la complejidad de la política de reemplazo para no producir overhead del SO
- El SO decide, entre el conjunto de páginas consideradas, si el reemplazo tiene un:
  - **Alcance Local**: Limitado al proceso que ha sufrido un fallo de página
  - **Alcance Global:** El conjunto de todas las páginas en marcos no bloqueados (unlocked)

# Políticas de Gestión del SO

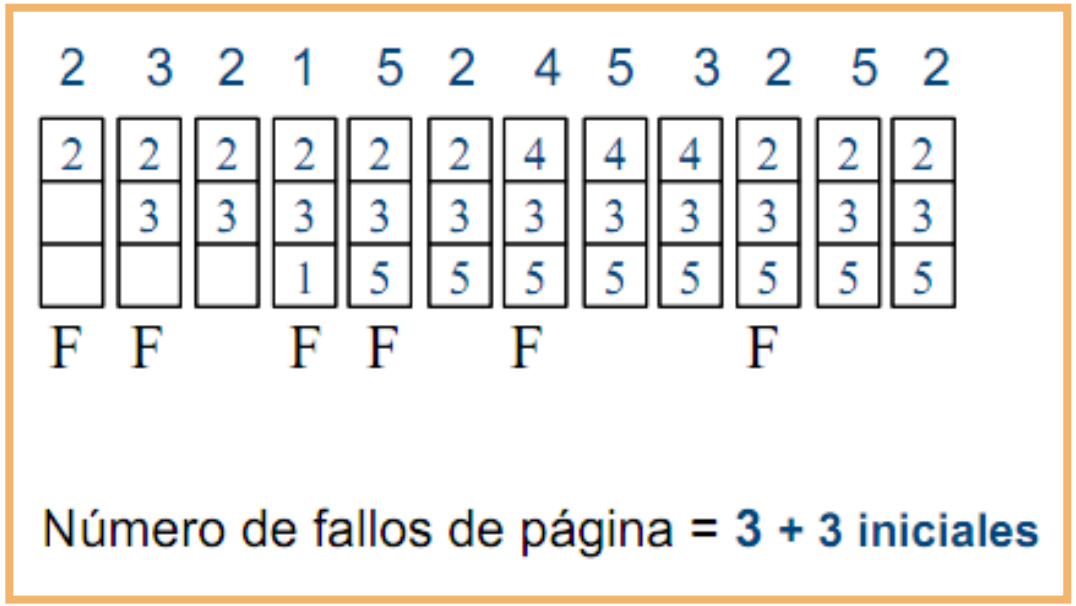
## Algunos algoritmos de reemplazo (en negrita, los de uso significativo)

- **Algoritmos canónicos:**
  - **FIFO**
  - OPT / MIN – algoritmo óptimo
  - LRU – least recently used
  - LFU – least frequently used
- **Aproximaciones realistas:**
  - **NRU – not recently used**
  - **segunda oportunidad o reloj**
  - **aging**
  - NFU – not frequently used
  - **WSClock**

# Políticas de Gestión del SO

## Algoritmo Reemplazo de Página: Óptimo

- Consiste en sustituir la página que vaya a ser referenciada más tarde, por ejemplo si hay una página A que será usada dentro de 10000 instrucciones, y una página B que será usada dentro de 2800 instrucciones, se debería eliminar de la memoria la página A
- El sistema operativo debería ver en cuánto tiempo será usada cada página en memoria y elegir la que está más distante
- Es un algoritmo teórico
- Se utiliza a los efectos comparativos con los algoritmos factibles de ser implementados para ver cuál se aproxima más a éste

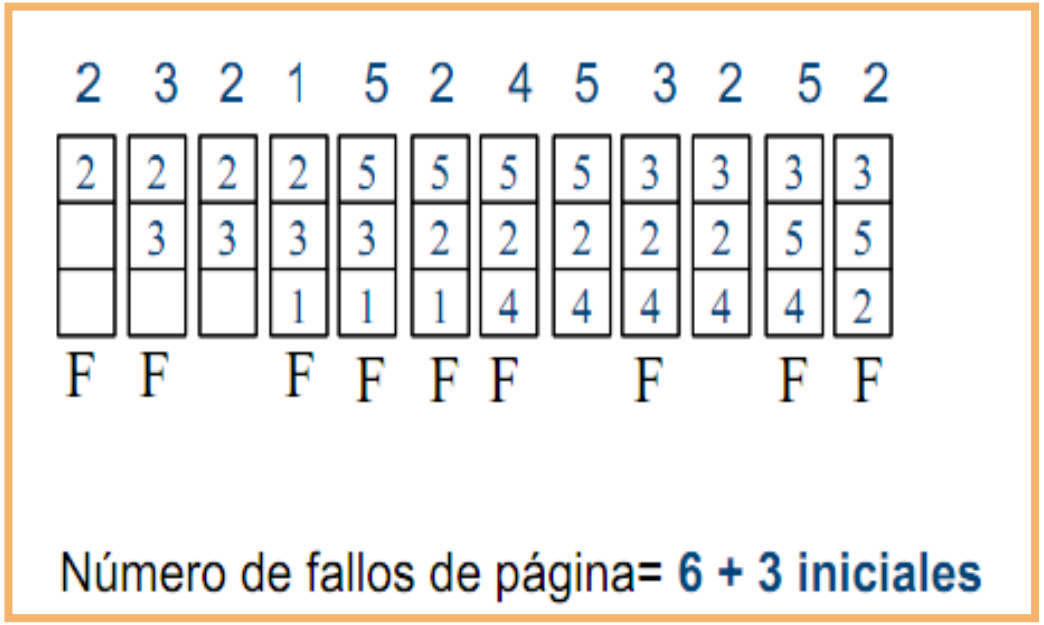



**Problema: Requiere conocimiento del futuro, por lo que es imposible su implementación**

# Políticas de Gestión del SO

## Algoritmo de Reemplazo Página: FIFO

El sistema operativo tiene una lista de todas las páginas que se encuentran en memoria, siendo la primera página la mas antigua y la última la mas reciente, en un fallo de página, se elimina la primera página y se añade la nueva al final de la lista




Número de fallos de página= **6 + 3 iniciales**

**Problema: El rendimiento no es siempre bueno, pueden sustituirse páginas muy usadas. No se usa FIFO, al menos en su forma pura**

# Políticas de Gestión del SO

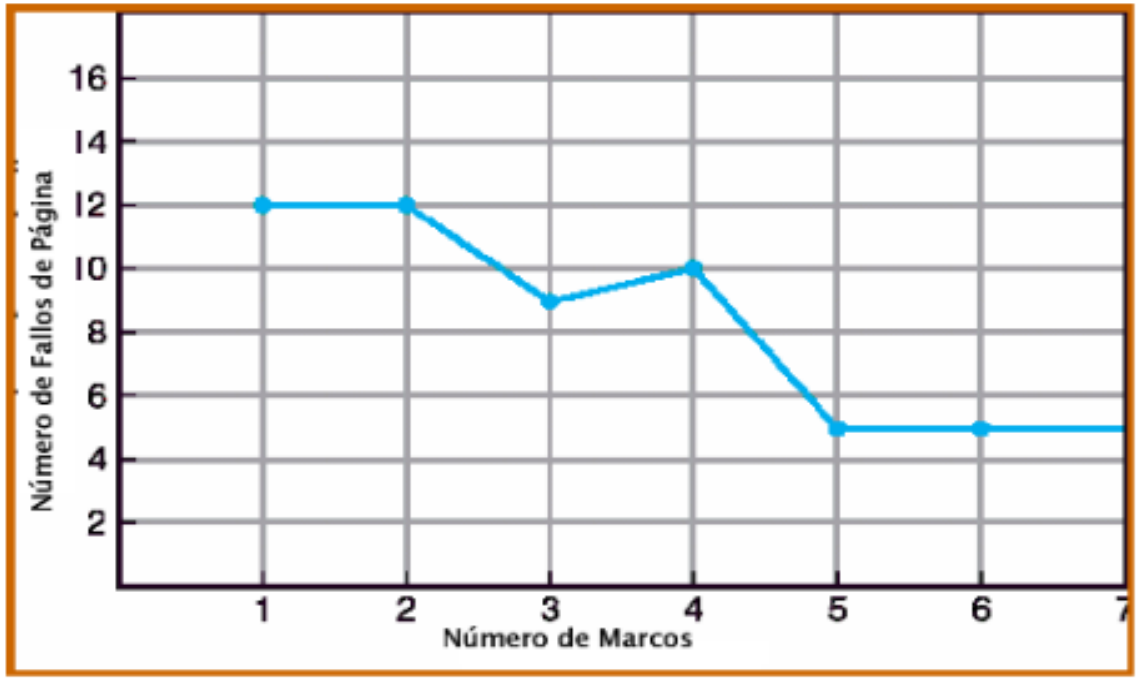
**Algoritmo de Reemplazo Página: FIFO**

**Anomalía de Bélády**

FIFO padece un fenómeno paradójico...

...si aumentamos el número de marcos físicos, **¡puede aumentar la cantidad de fallos de página!**

Descrito por László Bélády (IBM) en 1969

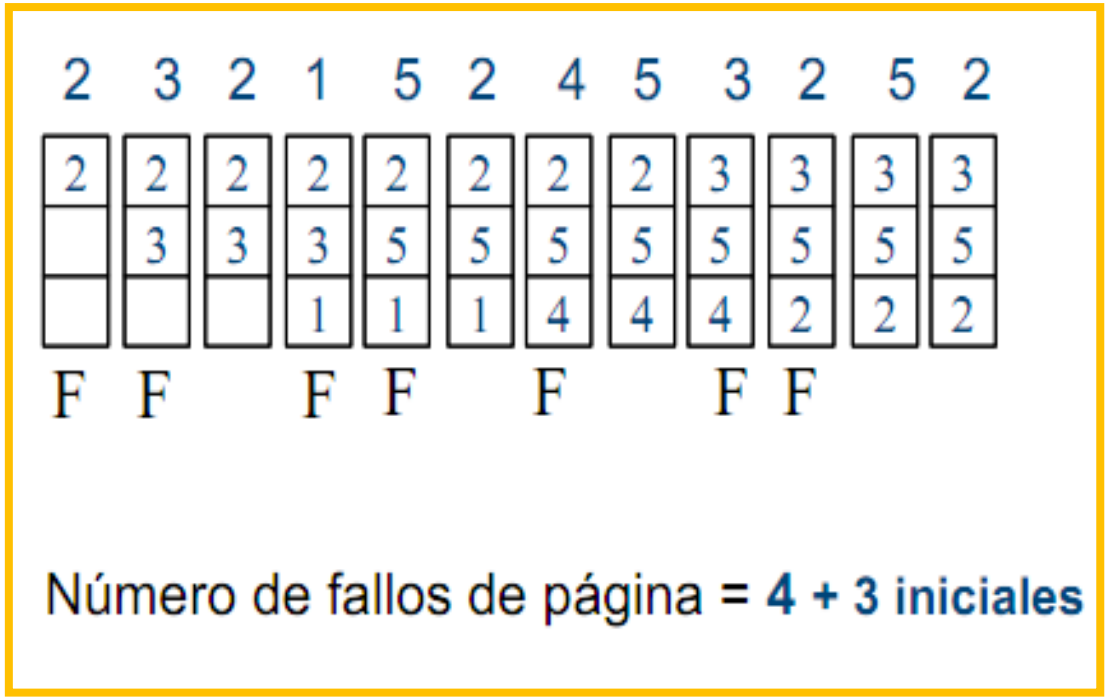



Cadena de referencias: 3 2 1 0 3 2 4 3 2 1 0 4

# Políticas de Gestión del SO

## Algoritmo de Reemplazo Página: Menos Recientemente Usada (Least Recently Used)

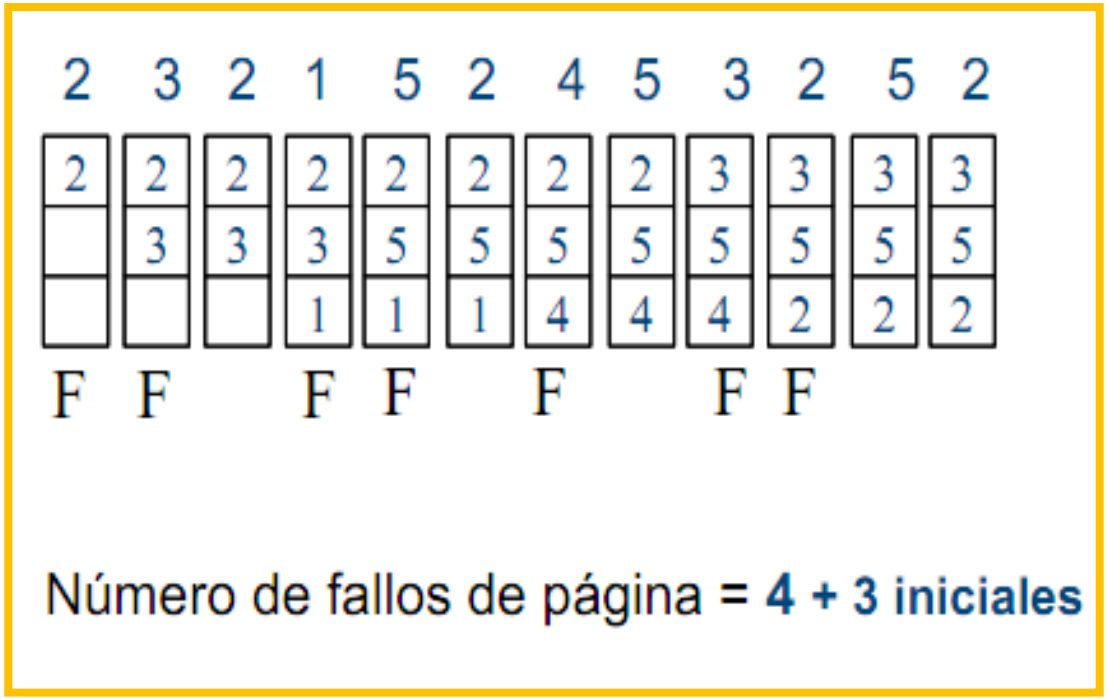
- El algoritmo Menos Usada Recientemente intenta proveer un comportamiento casi óptimo mediante la observación de las páginas que fueron menos usadas recientemente. Este tipo de páginas, estadísticamente son las que tienen menor probabilidad de ser usadas nuevamente



# Políticas de Gestión del SO

## Algoritmo de Reemplazo Página: Menos Recientemente Usada (Least Recently Used)

- Se desaloja la que tenga más tiempo sin usarse
- Requiere mantener una lista enlazada de todas las páginas:
  - Las usadas más recientemente al frente
  - Las menos usadas recientemente al final



# Políticas de Gestión del SO

## Algoritmo de Reemplazo Página: Menos Recientemente Usada (Least Recently Used)

### Implementación por hardware a partir de un Contador

- Cada vez que se accede memoria se incrementa su valor
- Se copia el valor del contador en la tabla de páginas asociado a la página a la que se accedió
- Se elimina la página que tiene el valor del contador más bajo

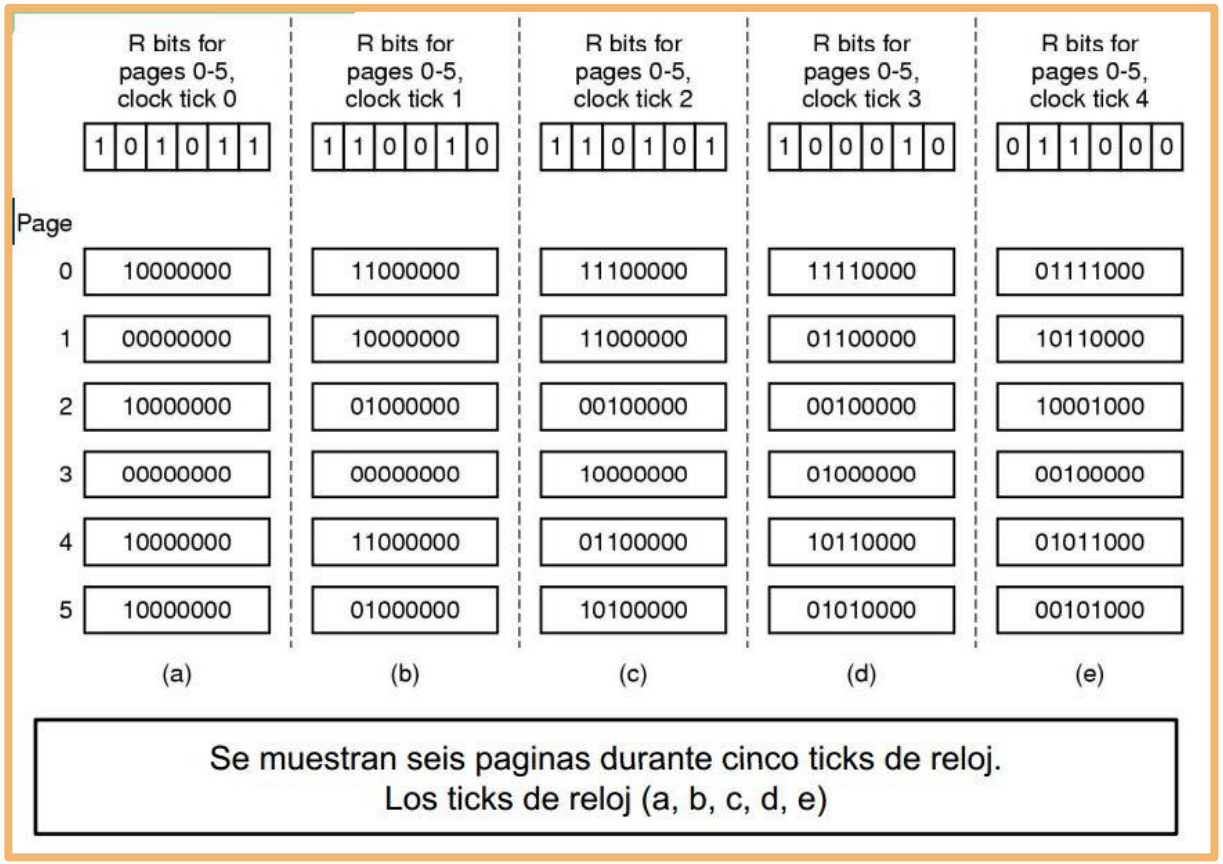
**Problema: El algoritmo nunca olvida**

# Políticas de Gestión del SO

## Algoritmo de Reemplazo Página: Menos Recientemente Usada (Least Recently Used)

### Contador Modificado

- Todos los contadores se desplazan a la derecha un bit antes de sumarle el bit R
- El bit R se suma al bit de la extrema izquierda, en lugar de la derecha
- Se desaloja la página cuyo contador es el mas bajo
- Es evidente que al no hacer referencia a una página, ésta tendrá n ceros a la izquierda(por ticks del reloj), por lo que será el valor mas pequeño
- Mismo contador: se toma al azar cualquiera

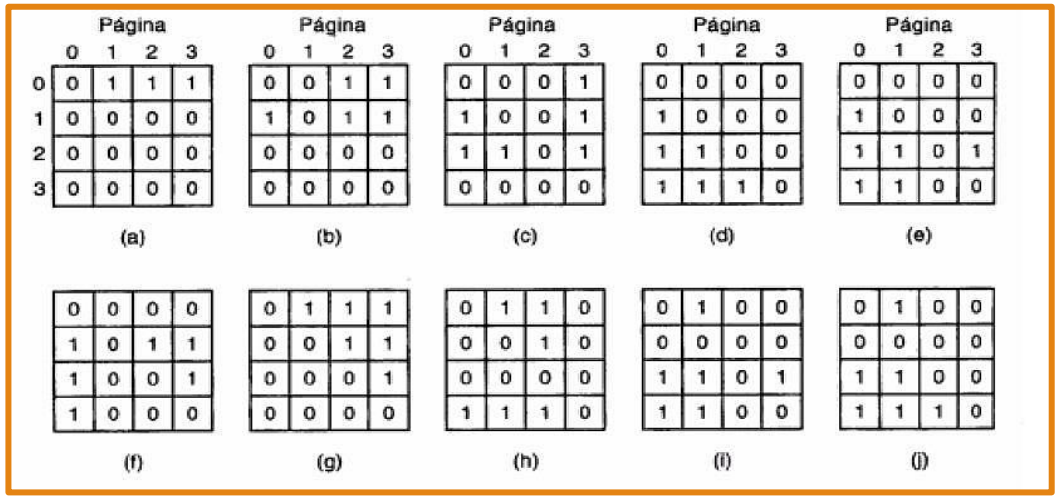


# Políticas de Gestión del SO

## Algoritmo de Reemplazo Página: Menos Recientemente Usada (Least Recently Used)

### Matrices

- Con *n* marcos, se utiliza una matriz *n x n* bits, inicializada en cero
- Cada vez que se referencia al marco k, se coloca a 1 todos los bits de la fila k, y luego coloca a 0 los de la columna k
- La fila cuyo valor binario sea más bajo, será el marco menos recientemente usado




0123210323

**Problema: Es factible pero su implementación es costosa**

# Políticas de Gestión del SO

## Algoritmo de Reemplazo Página: Segunda Oportunidad

Evita el problema de desalojar una página muy utilizada

- Se examina el bit R de la página más antigua:
  - Si R=0, indica que, aparte de antigua no se usa mucho. Entonces es reemplazada
  - Si R=1, se apaga el bit y la página es tratada como una recién cargada. Luego sigue la búsqueda
- Si se ha hecho referencia a todas las páginas, el algoritmo se comporta como un FIFO puro
- La lista analiza tiempos de llegada

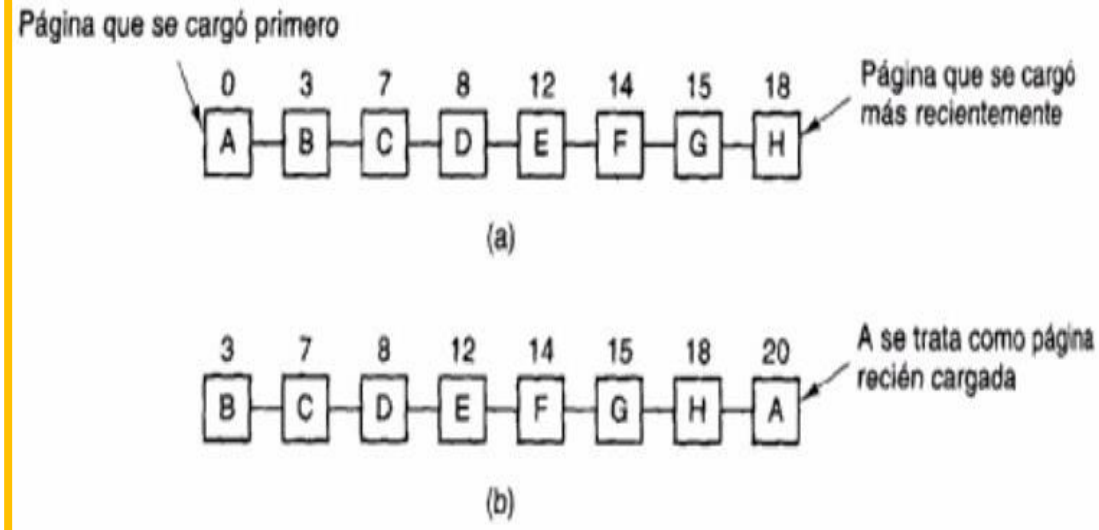



**Figura 4-13.** Funcionamiento del algoritmo de segunda oportunidad. (a) Páginas ordenadas en orden FIFO. (b) Lista de páginas si ocurre una falla de página en el tiempo 20 y *A* tiene su bit *R* encendido.

**Problema: Es ineficiente, porque constantemente cambia páginas de lugar**

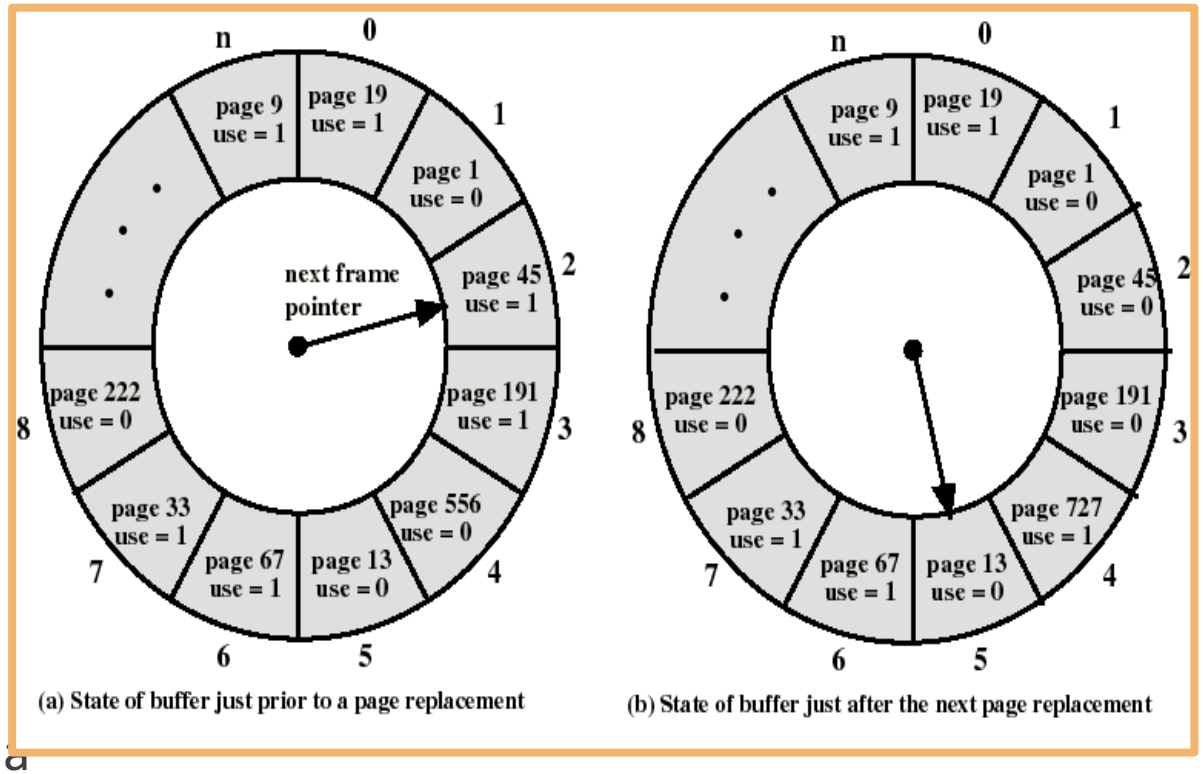
# Políticas de Gestión del SO

## Algoritmo de Reemplazo Página: El Reloj

Consiste en mantener las páginas en una lista circular con la forma de un reloj, una manecilla apunta hacia la más antigua.

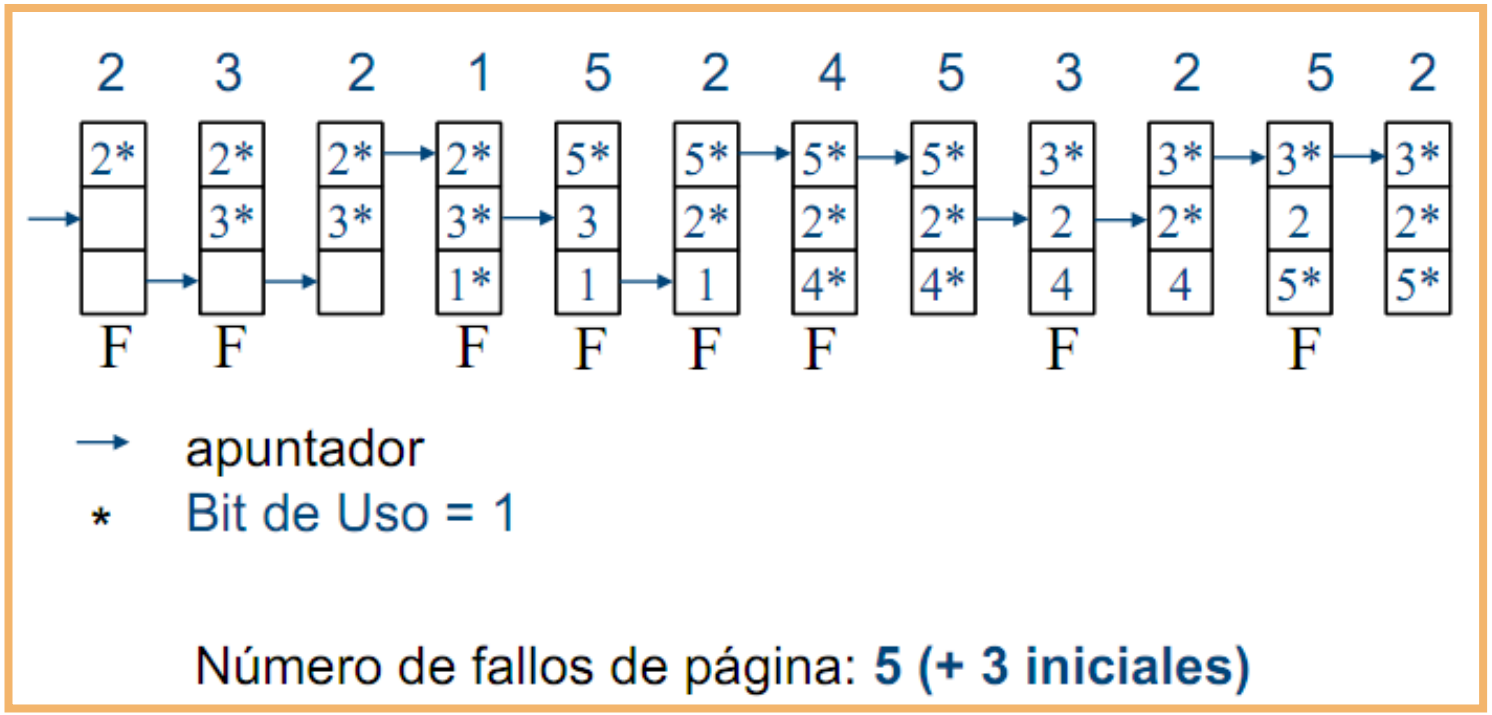
Al ocurrir un fallo de página se inspecciona la página a la que apunta la manecilla:

- R=0 se retira de la memoria, se inserta la nueva página en su lugar en el reloj y la manecilla avanza una posición
- R=1 la manecilla avanza una posición y el bit se limpia, esto continua hasta encontrar una página con R=0



# Políticas de Gestión del SO

## Algoritmo de Reemplazo Página: El Reloj



# Políticas de Gestión del SO

## Algoritmo de Reemplazo Página: No Recientemente Usado (*Not recently used*)

Este algoritmo favorece a las páginas que fueron usadas recientemente

Funciona de la siguiente manera:

- Cuando una página es referenciada, fija el bit de referencia para esa página
- Similarmente, cuando una página es modificada, fija su bit de modificación

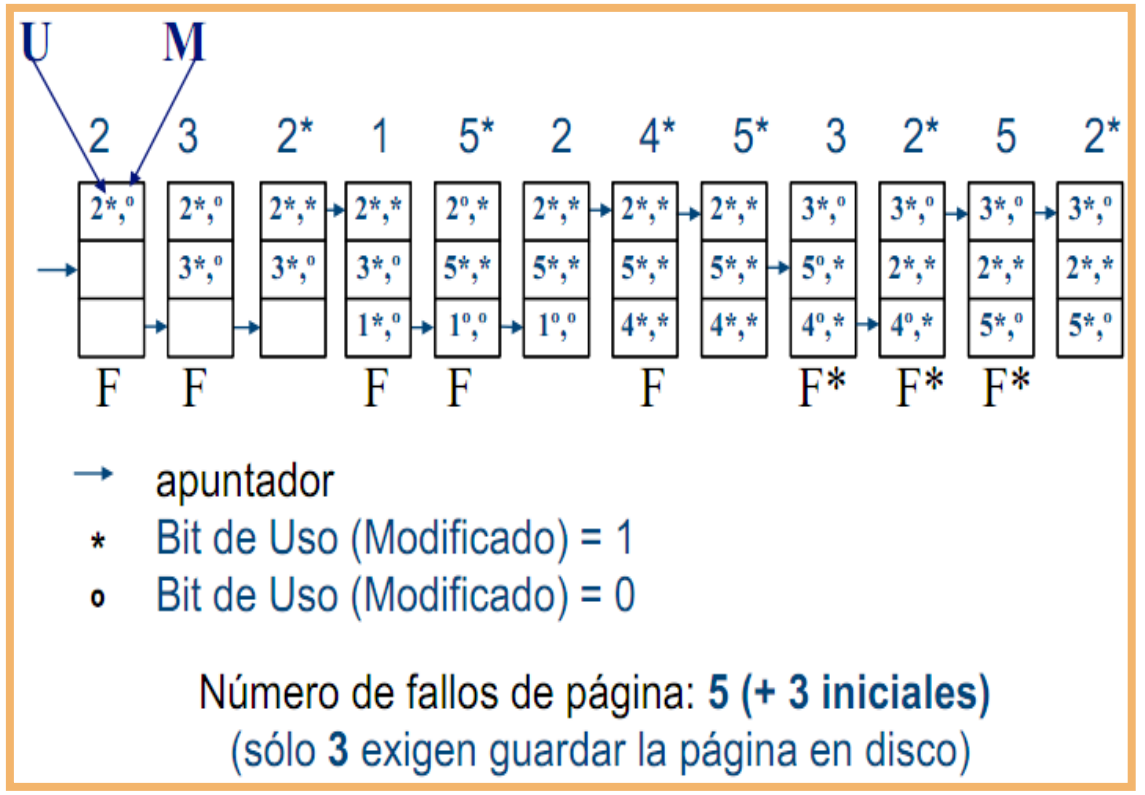
En un fallo de página , el sistema operativo inspecciona todas las páginas y las divide en cuatro categorías según los valores actuales de los bits R y M.

- **Clase 0:** No se ha hecho referencia ni ha sido modificada.
- **Clase 1:** No se ha hecho referencia pero ha sido modificada.
- **Clase 2:** Se ha hecho referencia pero no ha sido modificada.
- **Clase 3:** Se ha hecho referencia y ha sido modificada

# Políticas de Gestión del SO

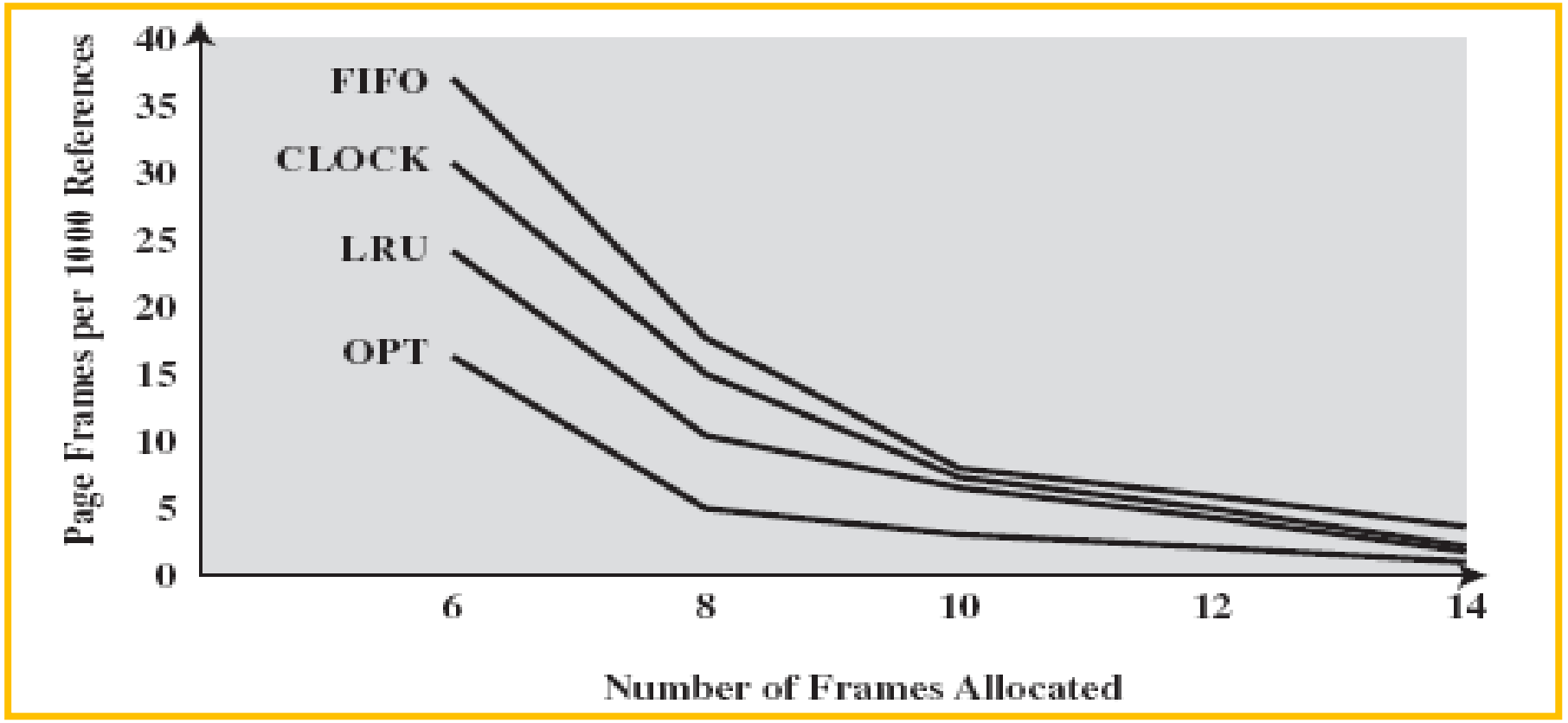
## Algoritmo Reemplazo Página: No Recientemente Usado (*Not recently used*)

1. Se recorre el buffer de marcos, sin cambiar el bit de referencia R
   - Se selecciona para reemplazo el primer marco encontrado de clase 0 (R=0, M=0)
2. Si falla 1, se recorre el buffer buscando marcos de clase 1 (R=0, M=1)
   - Durante este recorrido, se coloca R=0 en todos los marcos por los que se pasa
   - Se selecciona para reemplazo el primer marco encontrado (copiar en disco la página)
3. Si falla 2, se repite 1 y, si es necesario, 2 (ahora todos los marcos tienen R=0)



# Políticas de Gestión del SO

## Comparación de Algoritmos

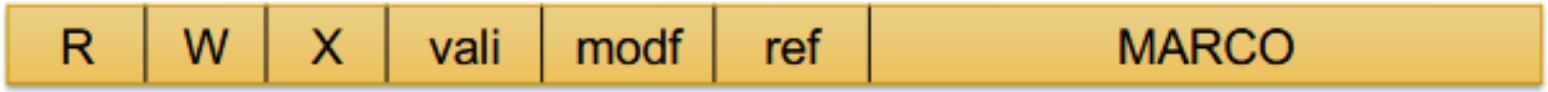


# Políticas de Gestión del SO

## Recapitulando: bits en la tabla de páginas

Para gestionar la memoria virtual, los fabricantes de MMU han inventado estos tres bits para la tabla de páginas:

- **Bit de validez:** nos dice si la entrada en la tabla apunta a una página residente en memoria.
- **Bit de modificación:** nos dice si la página se ha modificado.
- **Bit de referencia:** nos dice si la página se ha accedido recientemente.
- Aparte, la MMU suele ofrecer bits para establecer permisos de acceso a cada página (ej. RWX).



# Políticas de Gestión del SO

## Políticas de Asignación

- **Asignación Fija:** Asigna un número fijo de marcos que permanecen constantes con el tiempo
  - El número está determinado en momento de carga (instante de creación del proceso)
  - Depende del tipo de proceso (interactivo, batch, tipo de aplicación) o está en función de las directrices del programador o del administrador del sistema
- **Asignación Variable:** El número de marcos asignados a un proceso, puede variar con el tiempo
  - Puede aumentar si la proporción de fallo de página es alta
  - Puede disminuir si el porcentaje de fallo de página es muy baja
  - Requiere más overhead del SO, para evaluar el comportamiento de los procesos activos

# Políticas de Gestión del SO

## Gestión del Conjunto Residente

### Definición de Conjunto Residente

El SO debe decidir cuánta memoria asignar a un proceso (nº págs. a cargar en memoria principal):

- Cuanto menos memoria necesite cada proceso, mayor cantidad de procesos en memoria entonces mayor probabilidad de procesos listos en memoria
- Si hay pocas páginas de un proceso en memoria, aumenta la probabilidad de fallo de página

Alcance del reemplazo

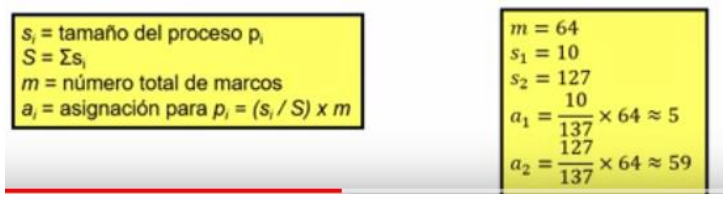
Tamaño del conjunto residente

| | Reemplazo Local | Reemplazo Global |
|---|---|---|
| **Asignación Fija** | El número de marcos asignados es fijo.<br>La página ha reemplazar se elige de entre los marcos asignados al proceso. | No es posible |
| **Asignación Variable** | El número de marcos asociado a un proceso puede cambiar de un momento a otro para mantener su conjunto de trabajo.<br>La página a reemplazar se elige de entre las paginas asignadas al proceso | La pagina a reemplazar se elige de entre todos los marcos disponibles en memoria principal.<br>Esto hace que se cambie el conjunto residente de los procesos. |

# Políticas de Gestión del SO

## Gestión del Conjunto Residente

### ❑ Asignación Fija con Reemplazo Local




- Asigna un número fijo de marcos al proceso: equitativo o proporcional
- Determinado al momento de la carga del proceso y depende del tipo de aplicación
- Cuando un fallo de página ocurre: la página a reemplazar se elige de entre los marcos asignados al proceso

**Problema: Difícil decidir por adelantado un buen número de marcos a asignar.**
**Si es demasiado bajo, el porcentaje de fallo de páginas será alto**
**Si es demasiado grande, el nivel de multiprogramación será demasiado bajo**

# Políticas de Gestión del SO

## Gestión del Conjunto Residente

- **Asignación Fija con Reemplazo Global**
  - Imposible de lograr
  - Si todos los marcos no bloqueados (unlocked) son candidatos para el reemplazo, el número de marcos asignados a un proceso necesariamente variará con el tiempo

# Políticas de Gestión del SO

## Gestión del Conjunto Residente

### ❑ Asignación Variable con Reemplazo Local

- Puede ser la mejor combinación (usado por Windows NT)
- Asigna en tiempo de carga un cierto número de marcos a un nuevo proceso, basado en el tipo de aplicación, necesidades del programa u otros criterios
- La asignación puede cubrirse tanto con paginación previa, como con paginación por demanda
- Cuando un fallo de página ocurre, selecciona la página a reemplazar del conjunto residente del proceso que sufre la falla
- Periódicamente reevalúa la asignación otorgada al proceso y, aumenta o disminuye para mejorar el performance global

# Políticas de Gestión del SO

## Gestión del Conjunto Residente

### ❑ Asignación Variable con Reemplazo Global

- Simple de implementar, adoptado por muchos SO (como Unix SVR4)
- Requiere del manejo de una lista de marcos libres
- Cuando un proceso emite un fallo de página, asigna un marco libre de la lista
- Los procesos que producen fallos de página incrementan gradualmente su tamaño, lo que ayuda a reducir el número global de fallos de página en el sistema
- La escogencia del proceso que perderá una página de su conjunto residente (perderá un marco) es arbitraria: lejos de ser óptimo
- El almacenamiento intermedio de páginas puede aliviar este problema por cuanto, una página puede ser recuperada si se produce una referencia antes de que se escriba el bloque con el siguiente grupo de páginas

# Políticas de Gestión del SO

## Políticas de Vaciado

Responde a cuándo una página modificada debe ser escrita a disco

- **Vaciado por demanda (Demand cleaning)**
  - Una página sólo se escribe a disco, cuando su marco se selecciona para reemplazo
  - Un proceso que sufre una fallo de página, puede tener que esperar por 2 transferencias de página
- **Vaciado Previo (Precleaning)**
  - Escribe páginas modificadas antes de que se necesiten sus marcos, de forma que las páginas puedan escribirse por lotes (batch)
  - Tiene poco sentido escribir a disco un conjunto de páginas, que serán modificadas próximamente

# Políticas de Gestión del SO

## Estrategia del Conjunto de Trabajo

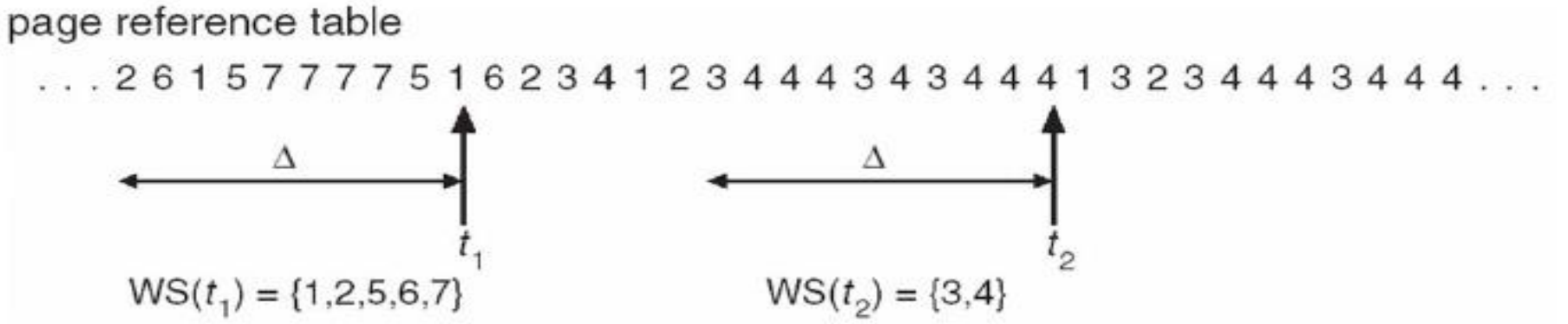
Conjunto de trabajo de un proceso en un instante t,W(t, Δ), se define como el conjunto de páginas a las que el proceso ha hecho referencia en las últimas Δ unidades de tiempo virtual

- t = Tiempo virtual = tiempo que transcurre (time elapsed) mientras el proceso está realmente en ejecución (se puede considerar el número de instrucciones ejecutadas)
- Δ es una ventana de tiempo para la observación del proceso
- W(Δ,t) es una aproximación de la localidad del programa
- W crece  con Δ
- W depende del instante t y puede valer desde1 hasta nº total de páginas del proceso

# Políticas de Gestión del SO

## Estrategia del Conjunto de Trabajo

- Su tamaño puede variar durante la ejecución
- Estabilidad: principio de cercanía
- Oscilaciones: indican desplazamiento del programa a otra ubicación
- Es un método de asignación variable con alcance local, basado en principio de localidad (cercanía) de referencias



# Políticas de Gestión del SO

## Estrategia del Conjunto de Trabajo

El concepto del conjunto de trabajo hace pensar en la siguiente estrategia, para determinar el tamaño del conjunto residente

- Supervise el conjunto de trabajo para cada proceso
- Periódicamente quite del conjunto residente de un proceso, esas páginas que no están en el conjunto de trabajo
- Cuando el conjunto residente de un proceso es menor que su conjunto de trabajo, asigne más marcos a él
- Si no están disponibles suficientes marcos libres, suspenda el proceso (hasta que más marcos estén disponible)
- Entonces: un proceso sólo puede ejecutarse si su conjunto de trabajo está en memoria principal

# Políticas de Gestión del SO

## Estrategia del Conjunto de Trabajo

Problemas prácticos con la estrategia del conjunto de trabajo

- La medida del conjunto de trabajo para cada proceso no es práctica
- Es necesario marcar (time stamp) las páginas referenciadas en cada referencia a memoria
- Es necesario mantener una cola ordenada en el tiempo de páginas referenciadas de cada proceso
- El valor óptimo para Δ es desconocido y, en cualquier caso, variable

**Solución: ¡en lugar de monitorear el conjunto de trabajo, monitoree el porcentaje de fallos de página de un proceso!**

# Políticas de Gestión del SO

## Estrategia del Conjunto de Trabajo

| tSequence of Page References | Window Size, Δ | | | |
|---|---|---|---|---|
| | 2 | 3 | 4 | 5 |
| 24 | 24 | 24 | 24 | 24 |
| 15 | 24 15 | 24 15 | 24 15 | 24 15 |
| 18 | 15 18 | 24 15 18 | 24 15 18 | 24 15 18 |
| 23 | 18 23 | 15 18 23 | 24 15 18 23 | 24 15 18 23 |
| 24 | 23 24 | 18 23 24 | • | • |
| 17 | 24 17 | 23 24 17 | 18 23 24 17 | 15 18 23 24 17 |
| 18 | 17 18 | 24 17 18 | • | 18 23 24 17 |
| 24 | 18 24 | • | 24 17 18 | • |
| 18 | • | 18 24 | • | 24 17 18 |
| 17 | 18 17 | 24 18 17 | • | • |
| 17 | 17 | 18 17 | • | • |
| 15 | 17 15 | 17 15 | 18 17 15 | 24 18 17 15 |
| 24 | 15 24 | 17 15 24 | 17 15 24 | • |
| 17 | 24 17 | • | • | 17 15 24 |
| 24 | • | 24 17 | • | • |
| 18 | 24 18 | 17 24 18 | 17 24 18 | 15 17 24 18 |

# Políticas de Gestión del SO

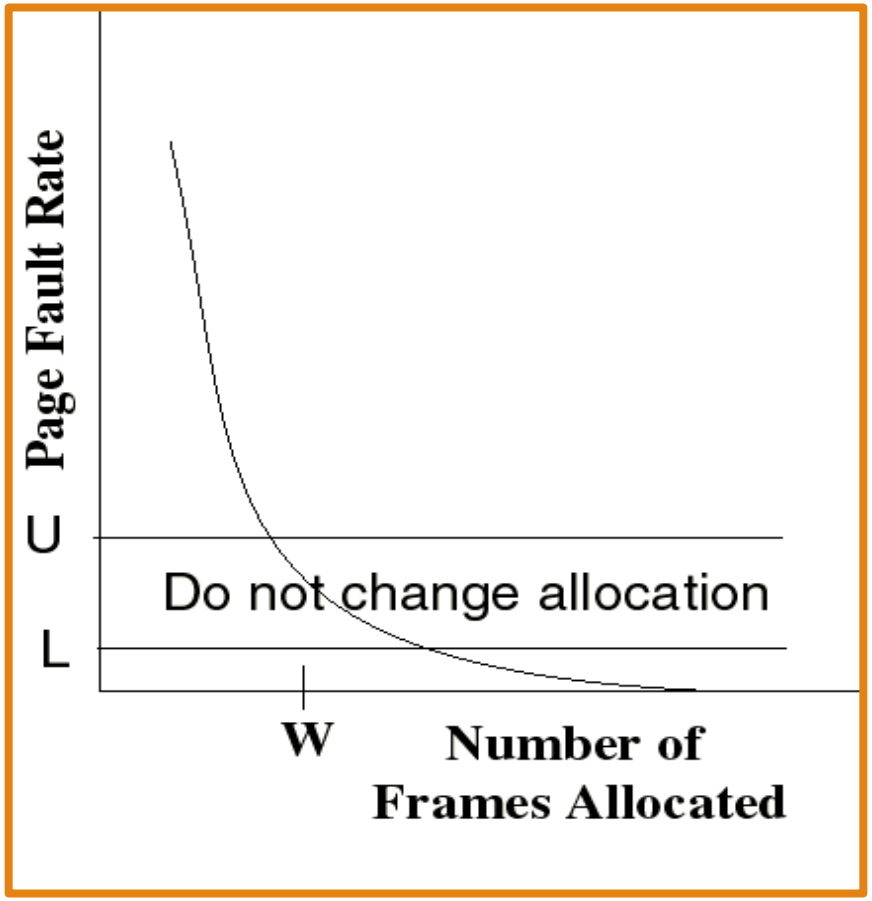
## Hiperpaginación (thrashing)

- Si el sistema tiene un grado de multiprogramación excesivo, puede ocurrir que se generen muchos fallos de página, porque ninguno de los procesos tiene «suficiente memoria para vivir tranquilo».
- Ejemplo: un proceso tiene un fallo de página y se elige como víctima la página de otro proceso que a su vez la necesita. Esto puede causar una reacción en cadena que hace que todo el sistema esté paginando continuamente.
- Los sistemas primitivos se tropezaron con este problema y no se sabía muy bien cómo darle solución... **¿cuándo sabes que un proceso «no tiene suficiente memoria»?**

# Políticas de Gestión del SO

## Frecuencia de Fallo de Página

- Define un limite superior U y un limite  inferior L para los porcentajes de fallo de página
- Asigna más marcos a un proceso si el porcentaje de fallo es mayor que U
- Asigna menos marcos si el porcentaje de fallo es < L
- El tamaño del conjunto residente debe estar cerca del tamaño del conjunto de trabajo W
- El proceso se suspenderá si el PFF (Page Fault Frecuency) > U y no hay marcos libres disponibles



# Políticas de Gestión del SO

## Control de Carga

Reducir el número de procesos que compiten por la memoria se puede lograr intercambiando algunos de ellos al disco y liberar todas las páginas que tenían asignadas

Si el *thrashing* cesa, el sistema podrá operar por un tiempo bajo estas condiciones

Si el *thrashing* no cesa, habrá que intercambiar otro proceso al disco, y así hasta que desaparezca

- Un algoritmo de conjunto de trabajo o frecuencia de fallo de página implícitamente incorpora control de carga
- Solamente se permitirá la ejecución de aquellos procesos cuyo conjunto residente sea suficientemente grande
- Otro enfoque es ajustar explícitamente el grado (nivel) de multiprogramación, para que el tiempo medio entre fallos de página sea igual al tiempo medio exigido para procesar un fallo de página
- Algunos estudios de performance indican que éste es el punto donde el uso del procesador alcanza un máximo

# Políticas de Gestión del SO

## Suspensión de Procesos

El Control de Carga explícito requiere que a veces se expulsen (suspenda) a los procesos

**Posibles criterios de selección de víctima**

- **Procesos con fallos de página**

  Este proceso puede no tener su conjunto de trabajo en memoria principal; en cuyo caso, dicho proceso se bloqueará de todas formas

- **Último proceso activado**

  Este es el proceso con menos posibilidades de tener su conjunto de trabajo residente

# Conclusiones

## Consideraciones actuales

- La localidad de los programas ha ido disminuyendo (OOP, recolector de basura, tablas hash...) ==> el concepto de «conjunto de trabajo» es más débil que hace 40 años
- Hoy día la memoria virtual y la caché de disco se han unificado en un único sistema
- Tamaño de página variable para adaptarse al WS
- Los fallos de TLB y de página son más graves (hay más diferencias de velocidad entre niveles de la jerarquía de memorias)