Memoria virtual

El concepto de la **memoria virtual** es que es el método para conseguir que la suma de los espacios de pila, datos y texto de un programa pueda ser mayor que el tamaño físico de la memoria disponible para él. El S.O. mantiene en memoria únicamente las partes del programa que se están utilizando y mantiene en disco el resto. Denominamos **conjunto residente** a la parte del proceso que está en la memoria principal.

La memoria virtual permite optimizar el uso de memoria principal, mantiene '**más procesos**' en memoria principal albergando solo una parte (suficiente) de cada uno de ellos, permite que el proceso sea más grande que toda la memoria principal.

La memoria virtual tiene problemas tales como:

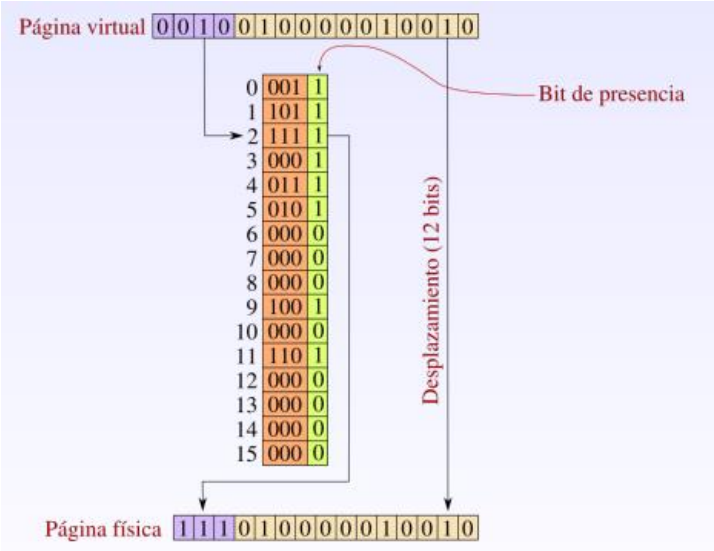
- Fallos de direccionamiento: Se intenta acceder a una posición que no está en memoria principal.
    1. Esto genera una interrupción.
    2. Pasa el estado del proceso a bloqueado.
    3. El S.O. emite una solicitud de E/S al disco.
    4. El S.O expide a otro proceso para que se ejecute (planificador del S.O.).
    5. La operación E/S se realiza, actualizándose la memoria principal.
    6. Se devuelve el control al S.O., que pasa el proceso a **estado listo**.
- Puede ser que se de el caso de que ocurra hiperpáginación (hyperthrashing).

En la **paginación**, la memoria física se divide en bloques de tamaño fijo (marcos), la memoria virtual se divide en bloques del mismo tamaño (páginas). Al ejecutar un proceso, se cargan sus páginas en los marcos disponibles, la vinculación de direcciones requiere de hardware, (manejador de memoria).

En un sistema con paginación no se produce fragmentación externa y si fragmentación interna. Si se trata de acceder a una página virtual que no esté asociada a un marco produce un señalamiento al S.O., llamado fallo de página.

Normalmente cada proceso tiene su **tabla de páginas**, esta requiere un soporte de hardware (manejador de memoria - MMU).

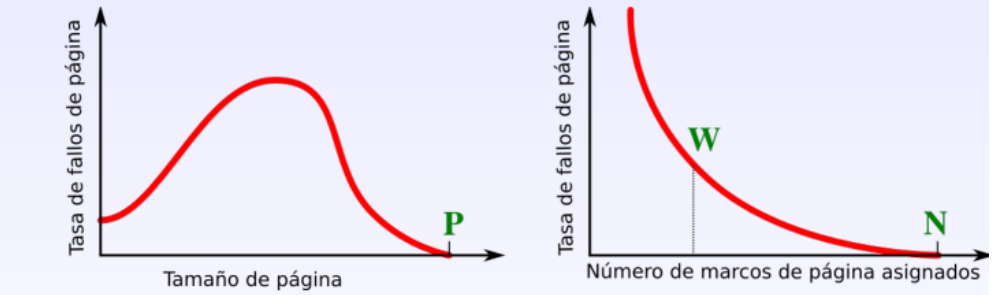
La tabla de páginas contiene el número de marco que corresponde a cada página virtual del proceso. Las tablas de páginas de los procesos tienen longitud variable (dependen del tamaño del proceso). La tabla se carga en memoria principal y, si es muy grande, puede estar sujeta a paginación → Paginación multinivel.



El tiempo de asociación entre página-marco debe ser reducido, para esto se usan soluciones basadas en hardware (utilizando registros), son las más rápidas, pero esto solo es válido si las tablas de páginas son pequeñas. Cuanto menor sea el tamaño de la página, menor será la cantidad de fragmentación interna.

Las páginas pequeñas, estarán en la memoria principal un gran número de estas por cada proceso. Después de un tiempo todas las páginas tendrán parte de las referencias más recientes del proceso. La tasa de fallos será menor.

Las páginas grandes, hacen que sea más eficaz la memoria secundaria al transferir eficazmente los bloques de datos de mayor tamaño. La tasa de fallos será mayor.



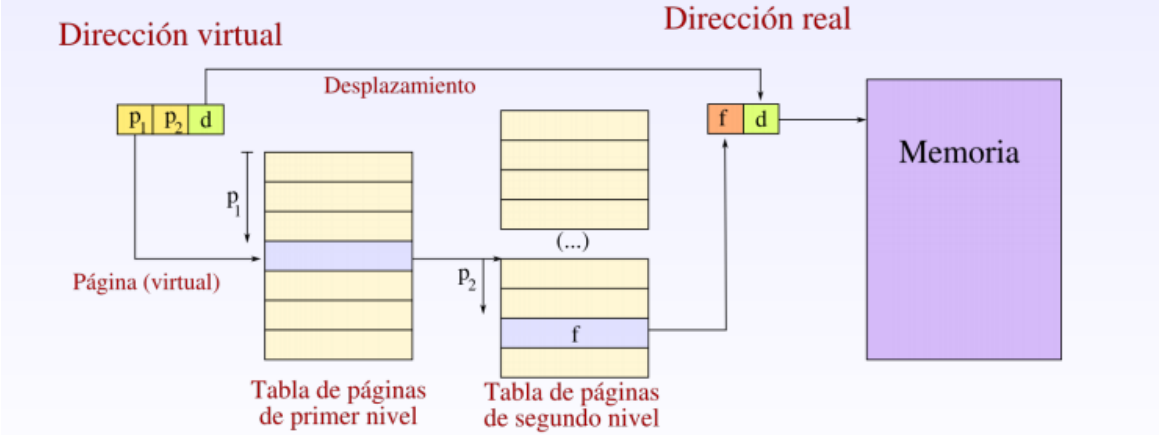
Donde:

- **P**: es el tamaño del proceso completo.
- **W**: es el tamaño del conjunto de trabajo.
- **N**: es el número total de páginas del proceso.

Para tratar las páginas grandes, se usa la paginación multinivel o las tablas de páginas invertidas.

La **paginación multinivel**:

- **Objetivo**: evitar tener siempre en memoria tablas de páginas completas.
- **Solución**: dividir la tabla en sub-tablas y mantener en memoria sólo las que sean necesarias en cada momento.



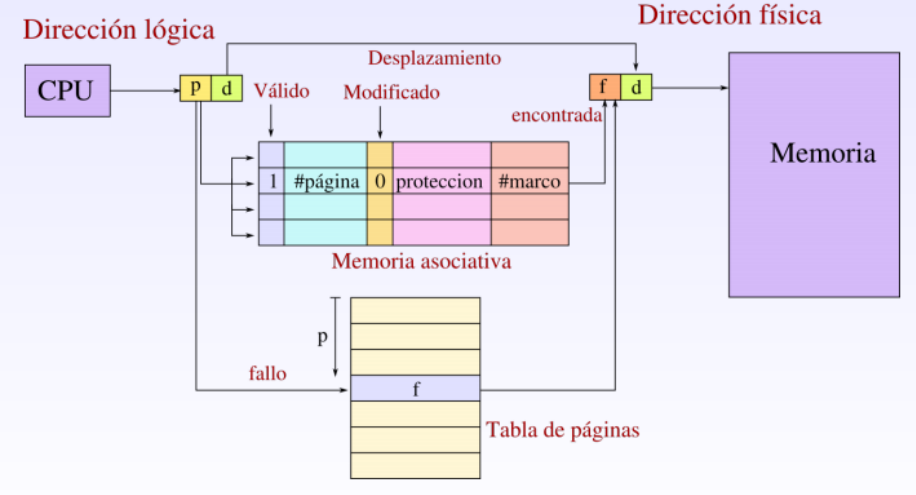
En las tablas de páginas el tiempo de acceso efectivo a memoria es:

$$t_{ae} = t_b + (1 - p) \cdot t_{am} + p \cdot t_{fallo} + t_{am}$$

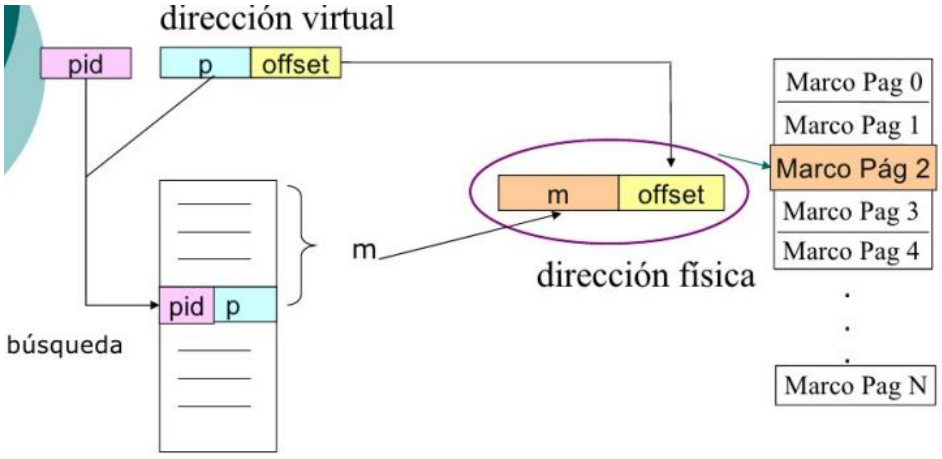
Siendo $\mathbf{t_b}$ el tiempo medio de búsqueda en la tabla de páginas, **p** la probabilidad de que ocurra un fallo de página, $\mathbf{t_{am}}$ el tiempo de acceso a la memoria y $\mathbf{t_{fallo}}$ el tiempo de resolución de un fallo de página.

$$t_{ae} = \sum_{i=1}^{N} \left[t_{bi} + (1 - p_i) \cdot t_{am} + pi \cdot t_{fallo}\right] + t_{am}$$

En la **memoria asociativa** tenemos lo que denominamos **TLB (Translation Lookaside Buffer)**, esta es una solución para acelerar la traducción de direcciones y, por tanto, el acceso a los marcos, cuando las tablas de procesos son muy grandes (búsqueda lenta) y/o están organizadas en niveles (requiere múltiples accesos a memoria).



Las tablas de **páginas invertidas** son sistemas con direccionamiento por 64 bits a nivel de Byte (2^{64}B). En sistemas donde el número de marcos físicos es sustancialmente menor, permite organizar la tabla de entradas alrededor de la memoria física en lugar de la memoria virtual. La tabla de páginas invertida tiene tantas entradas como marcos, y cada entrada contiene la dirección virtual de la página. Se utiliza siempre con una memoria asociativa. Si una dirección no se encuentra en la tabla invertida se hace una búsqueda en una tabla convencional, que puede estar en memoria o disco, si se encuentra se obtiene el número de marco.



Una tabla de página en el SO para todos los procesos

Las **políticas del gestor de memoria** están encaminadas a minimizar el porcentaje de fallos de páginas para maximizar el rendimiento. El rendimiento depende del tamaño de memoria principal, la velocidad relativa de memoria principal y secundaria, del tamaño y número de procesos que compiten por los recursos y del comportamiento de programas individuales. Todas las políticas son igual de buenas que las otras.

- Políticas de lectura (fetch):
  - Paginación por demanda: Se carga una página solo cuando se produce un fallo en esa página. Se produce un número elevado de fallos al inicio.
  - Paginación previa: Se carga la página que ha producido el fallo y las páginas cercanas a ésta.
- Políticas de ubicación: Carece de importancia en sistemas con paginación (con y sin segmentación).
- Políticas de reemplazo: Cuando la memoria está ocupada se reemplazan páginas. Para esto se usan algoritmos de reemplazo. El objetivo del algoritmo es de reemplazar la página con menos posibilidad de ser referenciada en un futuro cercano.
- Políticas de gestión del conjunto residente: El S.O. decide cuánta memoria asignar a un proceso.
  - Tamaño del conjunto residente. Cuanta menos memoria necesite cada proceso, mayor cantidad de procesos en memoria, mayor probabilidad de procesos en estado listo y aumenta el grado de

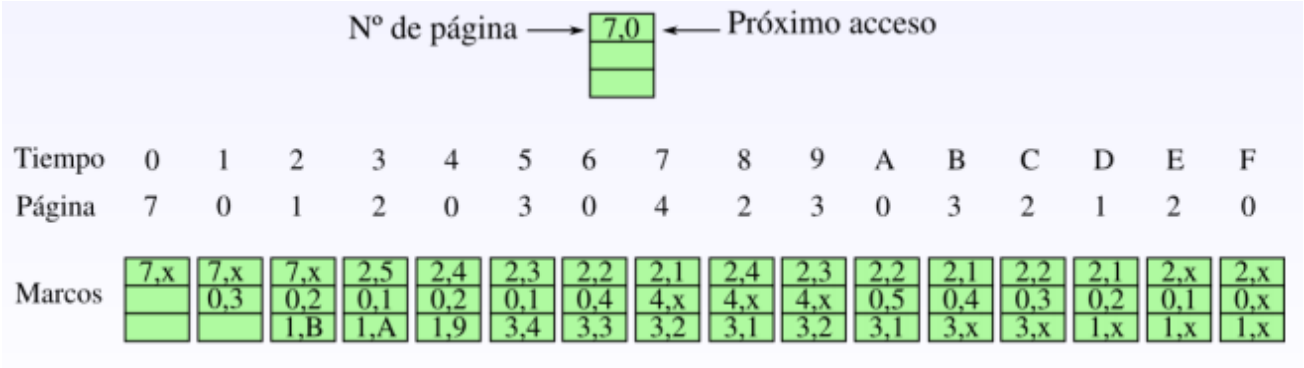
multiprogramación. Lo malo es que, si hay pocas páginas de un proceso en memoria, aumenta la probabilidad de fallos de página.
Asignación Fija. Se otorga a cada proceso un número fijo de páginas. Se decide la cantidad de memoria asignada al proceso en la carga inicial, según el tipo de proceso o directrices del programador o administrador, cuando hay fallos de página, siempre se reemplaza una página del mismo proceso. Si se asigna por exceso, se desperdicia espacio y es posible que el procesador sea ocioso (hay pocos procesos en memoria principal).
Si se asigna por defecto, alto porcentaje de fallos de páginas, aunque en el sistema haya marcos vacíos.

- Asignación variable. El número de marcos de un proceso cambia durante su vida. Cambia en función de la tasa de fallos de página. Se asignará dinámicamente el nº de marcos atendiendo a las necesidades del proceso. La asignación variable es más potente pero costosa.
- Los algoritmos de remplazo de páginas pueden ser de:
  - Ámbito global. Un proceso puede utilizar marcos que pertenecen a otro proceso.
  - Ámbito local. Un proceso sólo puede utilizar marcos que le han sido asignados a dicho proceso. Se desperdicia espacio si el proceso no necesita todos los marcos que se le han asignado.
- Asignación fija y remplazo de ámbito local. Al cargar un nuevo proceso, se le asigna un número de marcos fijo. Si se asignan pocas páginas se produce un alto porcentaje de fallos de páginas. Si se asignan muchas páginas hay pocos procesos en memoria, el procesador es ocioso.
- Asignación variable y remplazo de ámbito local. Al cargar un nuevo proceso se le asignan un nuevo número de marcos, cuando hay un fallo de página con reemplazo se selecciona una página del proceso que produjo el fallo de página.
- Asignación variable y reemplazo de ámbito global. Es sencilla de implementar. Cuando hay un fallo de página se crea un nuevo marco libre para el proceso, donde se carga la página. Se usa memoria intermedia para las páginas reemplazadas. Mejora el tiempo de recuperación de páginas recientemente reemplazadas.

- Política del conjunto de trabajo: Se utiliza para determinar el tamaño del conjunto residente y el momento de los cambios. Si el número de marcos disponibles es inferior al tamaño del conjunto de trabajo, se producirán frecuentes fallos de página. Un proceso hiperpáginado pasa más tiempo intercambiando páginas que ejecutándose.
  La finalidad de este conjunto es la de reducir la tasa de fallos de páginas. Hace uso del **principio de localidad/cercanía** donde las páginas que se utilizan en cada se momento pueden agrupar en grupos que varían a lo largo de la ejecución del proceso.
  La estrategia se basa en supervisar el conjunto de trabajo de cada proceso y eliminar periódicamente del conjunto residente las páginas que no pertenezcan a su conjunto de trabajo.
  Otra estrategia seria el uso del **algoritmo de frecuencia de fallos de página**, en este algoritmo cada página tiene un bit de uso que se pone a 1 cuando accede a la página. Al producirse un fallo de página el S.O. mira el tiempo transcurrido desde el último fallo de página para el proceso, si el tiempo es menor que el umbral se añade una página al conjunto residente, en caso contrario se descartan las páginas con el bit de uso a 0, ó se restaura el valor del bit de uso de las páginas restantes.
- Políticas de vaciado:
  - Vaciado por demanda: Se escribe la página en disco sólo cuando se va a reemplazar. Esto minimiza las escrituras en disco, pero cada fallo produce 2 transferencias.
  - Vaciado previo: Escribe las páginas modificadas antes de que se necesiten sus marcos, escribiéndolas por lotes. Aprovecha las ventajas de escribir en el disco por lotes, aunque se pueden producir posibles operaciones innecesarias.
  - Otra alternativa sería incorporar almacenamiento intermedio, mantiene temporalmente las páginas recién reemplazadas en una zona de la memoria reservada para ello. Hay 2 tipos de zonas:
    - Lista de zonas para páginas modificadas.
    - Lista de zonas para páginas no modificadas.
- Control de carga: Si hay pocos procesos es probable que todos estén en estado bloqueado, en cambio si hay muchos, el tamaño medio del conjunto de trabajo no es el adecuado. El grado de multiprogramación se debe aumentar cuando el número de fallos de página sea pequeño y disminuir cuando sea grande. Los criterios a tener en cuenta son: baja prioridad, muchos fallos de página, ultimo activado, conjunto residente más pequeño, el proceso mayor y la mayor ventana de ejecución restante. La elección depende de los objetivos y el tipo de programa.

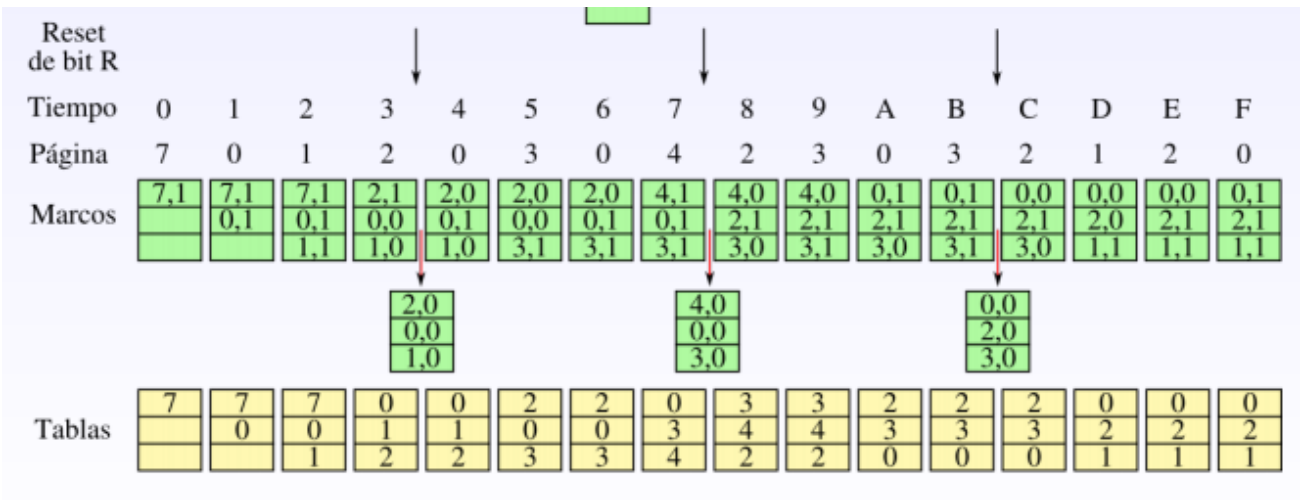
Los **algoritmos de remplazo** son los que eligen que marco de página debe ser intercambiado a disco para hacer sitio a la nueva página que se solicita. En general se sustituyen los marcos poco usados.

- La solución **optima** seria que cada pagina contenga una etiqueta con el número de instrucciones que transcurrirán hasta el próximo acceso a la página. Se reemplaza la página que tenga la etiqueta más alta. El problema es que es irrealizable ya que el SO no tiene forma de saber cuándo se va a realizar un nuevo acceso a una página.

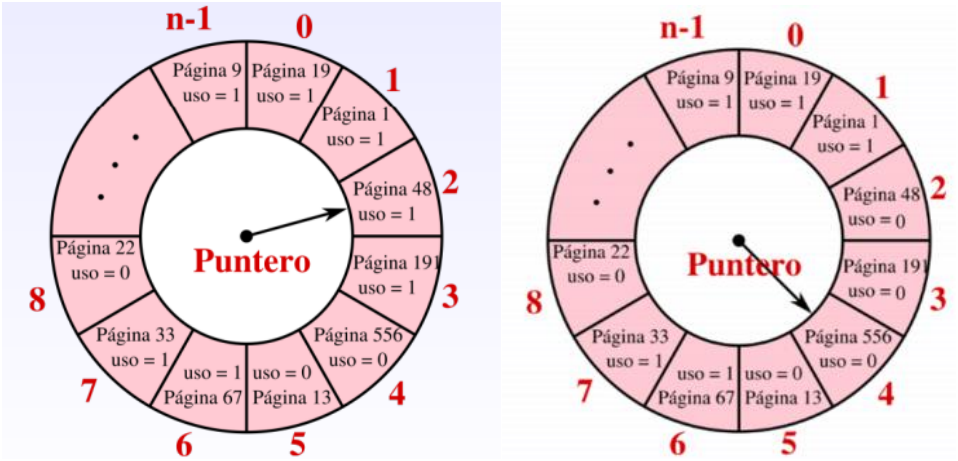


- **LRU** implementa una aproximación del algoritmo óptimo ya que se basa en mirar al pasado y a partir de él estimar cuál podría ser el uso de la página. Se selecciona el marco en el que está la página que lleva más tiempo sin acceso. Hay varias implementaciones posibles:
  - La tabla se implementa como una pila de números de página, en esta se conserva en la salida la página más reciente usada y en la base la menos recientemente usada.
  - Se usan contadores. Las entradas de la tabla de páginas tienen un campo de "tiempo de uso" en el que el manejador de memoria escribe el tiempo de cada referencia. Cada acceso a memoria requiere una búsqueda en la tabla de páginas y una escritura en memoria por cada acceso a memoria.
- **FIFO** cada entrada de la tabla de páginas tiene un registro asociado con el instante de carga de la página, o el SO mantiene la tabla de páginas orden de antigüedad. Cuando se produce un fallo de página y no hay marcos libres se intercambia a disco la página que lleve más tiempo en la tabla.

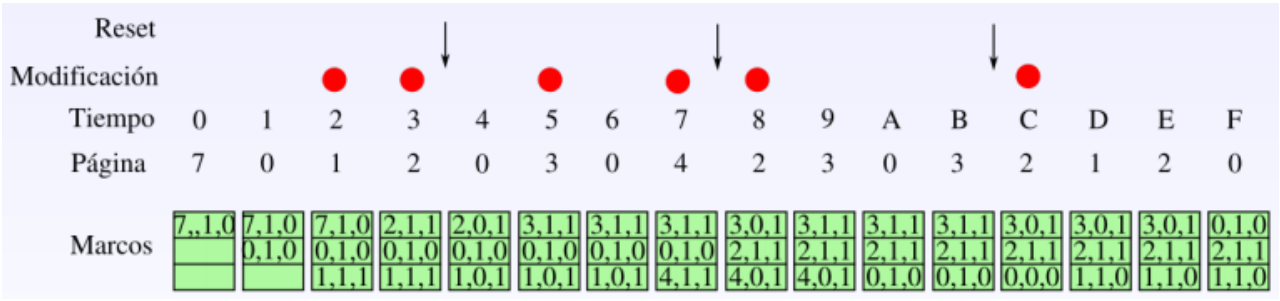
- **Segunda oportunidad (modificación de FIFO)**. Se revisa el bit de uso o referencia de la entrada más antigua, si es 1, se pone a 0 y se sitúa al final de la cola de páginas.



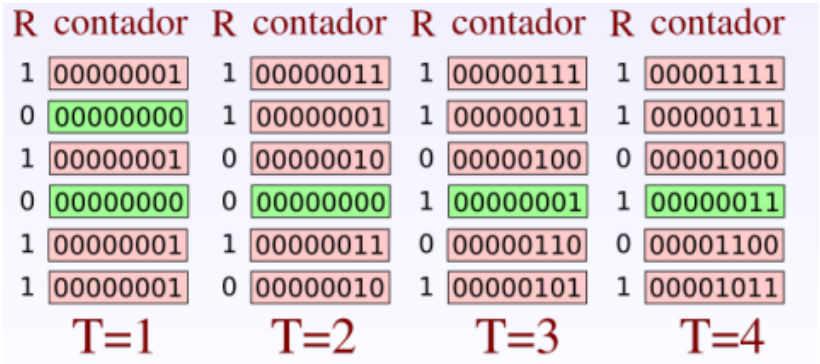
- **De reloj**. Las páginas se mantienen en una cola circular, con un puntero a la página más antigua. Si hay un fallo de página y el bit de uso del marco apuntado es U=0, la página que estaba en ese marco se reemplaza y se avanza el puntero. Si U=1, se pone U=0 y se avanza hasta encontrar una página con U=0.



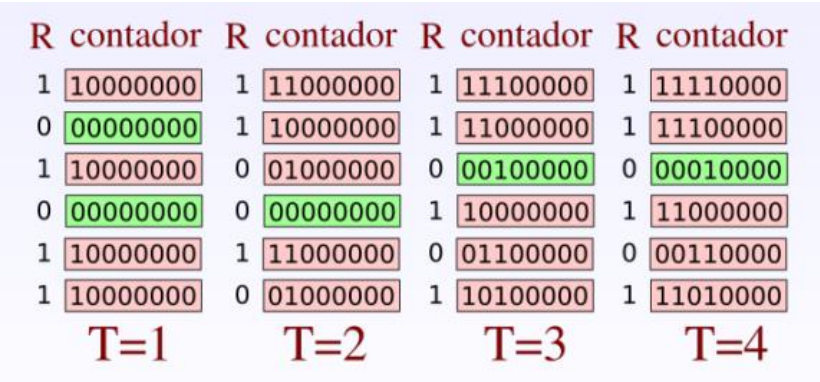
- **NRU (Not recently used)**. Utiliza bits de uso y modificado de las entradas de la tabla de páginas.



- **NFU (Not Frequently used)**. Sigue una aproximación de LRU utilizando solo hardware. Se activa un contador software por cada página, en cada interrupción de reloj. Cuando existe un fallo de página se selecciona la página con el contador más bajo y envía a disco.



Para solucionar el problema de memoria usamos maduración. Se activa un contador de software por cada página y en cada interrupción de reloj se desplaza el contador un bit a la derecha y luego se suma el bit R al bit más significativo del contador.



Algunos marcos de la memoria principal pueden bloquearse imponiendo una restricción a la política de reemplazo. La página situada en un marco bloqueado no puede ser reemplazada. Se utiliza para el núcleo del S.O. y otras áreas críticas. Se asocia un bit de bloqueo a cada marco.

La **segmentación** es una técnica para mantener espacios independientes de direcciones virtuales, llamados segmentos. Las distintas partes del proceso tienen espacios virtuales independientes entre sí.

La segmentación permite asignar permisos distintos a las partes del proceso y compartir ciertos datos o procedimientos entre procesos.

Los segmentos se direccionan desde 0 hasta una dirección máxima, que puede variar. La segmentación pura puede producir problemas de fragmentación externa.

Cuando añadimos paginación a la segmentación, los espacios de direcciones se dividen en segmentos, cada segmento dividido en páginas de tamaño fijo = tamaño de marco en memoria y la memoria principal se divide en marcos. Si el segmento es menor que una página, ocupa el marco entero.