El único problema con este algoritmo es que no se puede realizar. Al momento del fallo de página, el sistema operativo no tiene forma de saber cuándo será la próxima referencia a cada una de las páginas (vimos una situación similar antes, con el algoritmo de planificación del trabajo más corto primero: ¿cómo puede saber el sistema cuál trabajo es más corto?). Aún así, al ejecutar un programa en un simulador y llevar la cuenta de todas las referencias a páginas, es posible implementar un algoritmo óptimo de reemplazo de página en la *segunda* corrida, al utilizar la información de referencia de páginas recolectada durante la *primera* corrida.

De esta manera, se puede comparar el rendimiento de los algoritmos realizables con el mejor posible. Si un sistema operativo logra un rendimiento de, por ejemplo, sólo 1 por ciento peor que el algoritmo óptimo, el esfuerzo invertido en buscar un mejor algoritmo producirá cuando mucho una mejora del 1 por ciento.

Para evitar cualquier posible confusión, hay que aclarar que este registro de referencias de páginas se refiere sólo al programa que se acaba de medir y sólo con una entrada específica. Por lo tanto, el algoritmo de reemplazo de páginas que se derive de esto es específico para ese programa y esos datos de entrada. Aunque este método es útil para evaluar los algoritmos de reemplazo de páginas, no es útil en sistemas prácticos. A continuación estudiaremos algoritmos que *son* útiles en sistemas reales.

3.4.2 El algoritmo de reemplazo de páginas: no usadas recientemente

Para permitir que el sistema operativo recolecte estadísticas útiles sobre el uso de páginas, la mayor parte de las computadoras con memoria virtual tienen dos bits de estado asociados a cada página. *R* se establece cada vez que se hace referencia a la página (lectura o escritura); *M* se establece cuando se escribe en la página (es decir, se modifica). Los bits están contenidos en cada entrada de la tabla de páginas, como se muestra en la figura 3-11. Es importante tener en cuenta que estos bits se deben actualizar en cada referencia a la memoria, por lo que es imprescindible que se establezcan mediante el hardware. Una vez que se establece un bit en 1, permanece así hasta que el sistema operativo lo restablece.

Si el hardware no tiene estos bits, pueden simularse de la siguiente forma. Cuando se inicia un proceso, todas sus entradas en la tabla de páginas se marcan como que no están en memoria. Tan pronto como se haga referencia a una página, ocurrirá un fallo de página. Entonces, el sistema operativo establece el bit *R* (en sus tablas internas), cambia la entrada en la tabla de páginas para que apunte a la página correcta, con modo de SÓLO LECTURA, y reinicia la instrucción. Si la página se modifica después, ocurrirá otro fallo de página que permite al sistema operativo establecer el bit *M* y cambiar el modo de la página a LECTURA/ESCRITURA.

Los bits *R* y *M* se pueden utilizar para construir un algoritmo simple de paginación de la siguiente manera. Cuando se inicia un proceso, ambos bits de página para todas sus páginas se establecen en 0 mediante el sistema operativo. El bit *R* se borra en forma periódica (en cada interrupción de reloj) para diferenciar las páginas a las que no se ha hecho referencia recientemente de las que si se han referenciado.

Cuando ocurre un fallo de página, el sistema operativo inspecciona todas las páginas y las divide en 4 categorías con base en los valores actuales de sus bits *R* y *M*:

Clase 0: no ha sido referenciada, no ha sido modificada.

Clase 1: no ha sido referenciada, ha sido modificada.

Clase 2: ha sido referenciada, no ha sido modificada.

Clase 3: ha sido referenciada, ha sido modificada.

Aunque las páginas de la clase 1 parecen a primera instancia imposibles, ocurren cuando una interrupción de reloj borra el bit *R* de una página de la clase 3. Las interrupciones de reloj no borran el bit *M* debido a que esta información se necesita para saber si la página se ha vuelto a escribir en el disco o no. Al borrar *R* pero no *M* se obtiene una página de clase 1.

El algoritmo **NRU** (*Not Recently Used*, No usada recientemente) elimina una página al azar de la clase de menor numeración que no esté vacía. En este algoritmo está implícita la idea de que es mejor eliminar una página modificada a la que no se haya hecho referencia en al menos un pulso de reloj (por lo general, unos 20 mseg) que una página limpia de uso frecuente. La principal atracción del NRU es que es fácil de comprender, moderadamente eficiente de implementar y proporciona un rendimiento que, aunque no es óptimo, puede ser adecuado.

3.4.3 El algoritmo de reemplazo de páginas: Primera en entrar, primera en salir (FIFO)

Otro algoritmo de paginación con baja sobrecarga es el de **Primera en entrar, primera en salir** (*First-In, First-Out*, **FIFO**). Para ilustrar cómo funciona, considere un supermercado con suficientes repisas como para mostrar exactamente *k* productos distintos. Un día, cierta empresa introduce un nuevo alimento de conveniencia: yogurt orgánico instantáneo liofilizado que se puede reconstituir en un horno de microondas. Es un éxito inmediato, por lo que nuestro supermercado finito se tiene que deshacer de un producto antiguo para tenerlo en existencia.

Una posibilidad es buscar el producto que el supermercado haya tenido en existencia por más tiempo (por ejemplo, algo que se empezó a vender hace 120 años) y deshacerse de él por la razón de que nadie está interesado ya. En efecto, el supermercado mantiene una lista ligada de todos los productos que vende actualmente en el orden en el que se introdujeron. El nuevo pasa a la parte final de la lista; el que está al frente de la lista se elimina.

Como un algoritmo de reemplazo de páginas, se aplica la misma idea. El sistema operativo mantiene una lista de todas las páginas actualmente en memoria, en donde la llegada más reciente está en la parte final y la menos reciente en la parte frontal. En un fallo de página, se elimina la página que está en la parte frontal y la nueva página se agrega a la parte final de la lista. Cuando se aplica en las tiendas, FIFO podría eliminar la gomina para el bigote, pero también podría eliminar harina, sal o mantequilla. Cuando se aplica a las computadoras, surge el mismo problema. Por esta razón es raro que se utilice FIFO en su forma pura.

3.4.4 El algoritmo de reemplazo de páginas: segunda oportunidad

Una modificación simple al algoritmo FIFO que evita el problema de descartar una página de uso frecuente es inspeccionar el bit *R* de la página más antigua. Si es 0, la página es antigua y no se ha