Simulador de Políticas de Ubicación, Sustitución y Escritura de Memoria Caché

Jorge López O - jlo.lopez.ortega@gmail.com
Prof.: M. en C. Israel Rivera Zárate. – irivera@ipn.mx,
Prof.: M. en C. Miguel Hernández Bolaños. – mbolanos@ipn.mx
Centro de innovación y desarrollo tecnológico en cómputo - IPN

Resumen
El siguiente trabajo presenta un prototipo basado en un practicario con énfasis a la materia de arquitectura de computadoras, el practicario cuenta con 27 marcadores impresos en todo el cuadernillo. Un marcador es una imagen impresa que la computadora procesa y de acuerdo a la programación definida para el marcador, le incorpora un objeto 3D que podrá verse en la pantalla de un dispositivo móvil. La finalidad de este proyecto es aplicar el concepto de realidad aumentada que es la combinación del mundo real con el mundo virtual aplicado en el proceso enseñanza-aprendizaje.

Abstract

Los programas manifiestan una propiedad que se explota en el diseño de sistemas de gestión de memoria de un sistema de cómputo en general y de la memoria caché en particular, la localidad de referencias: los programas tienden a reutilizar los datos e instrucciones que utilizaron recientemente. Para implementar el mecanismo de actualización de la caché con los datos con mayor probabilidad de ser referenciados se divide la memoria principal en bloques de un número de bytes y la caché en marcos de bloque o líneas de igual tamaño. Finalmente, a la hora de diseñar un sistema de memoria caché se debe elegir entre una serie de alternativas que garanticen una adecuada sustitución de bloques de memoria así como coherencia de contenidos entre memoria caché y memoria principal

Introducción

Un efectivo tiempo de acceso al que el procesador debe trabajar debe tender a t0 genera por consecuencia un efectivo acceso a memoria principal y por ende a la memoria caché debido a que la mayoría de los accesos entre el procesador y la caché se encuentra a mismo nivel. Para tener la máxima eficiencia depende de factores como la arquitectura del procesador, la estructura de programas y el tamaño y tipo de administración de la memoria caché. Existen tres principios fundamentales sobre la caché sobre el comportamiento de un programa. Localidad espacial, temporal y secuencial.

Esta organización de la caché se divide en 3.

- Correspondencia directa entre Mp y Mc.

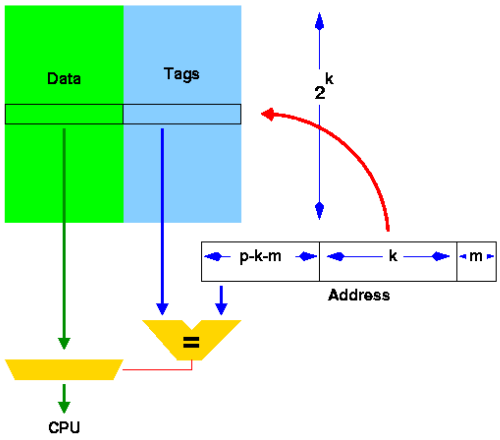


Fig. 1.1. Correspondencia de asignación directa a caché

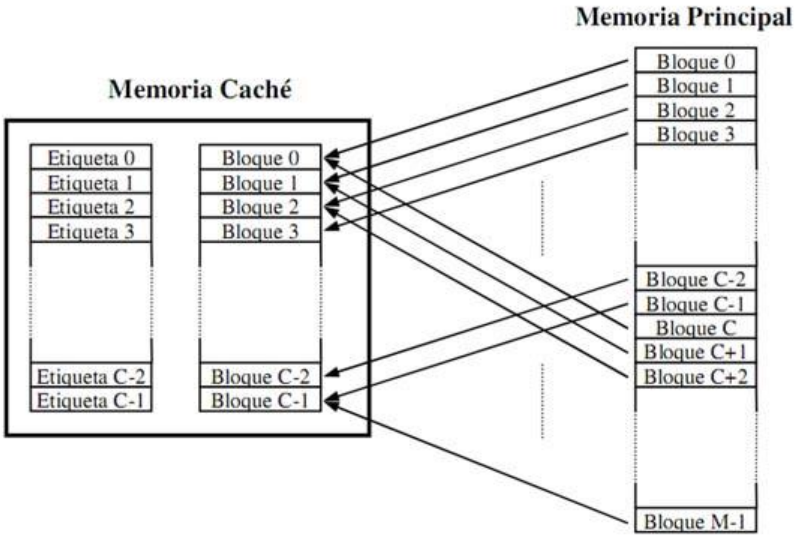


Figura 5.7 Asignación de bloques de la M_p en la M_{ca} con correspondencia directa.

Donde la función de transformación está dada por:

$$i = j \ Mod \ C$$

i = # de bloque asignado an la Mc al bloque de la Mp
j = # de bloque de la Mp
C = # de bloque que tiene la Mc

- Correspondencia totalmente asociativa entre Mp y Mc

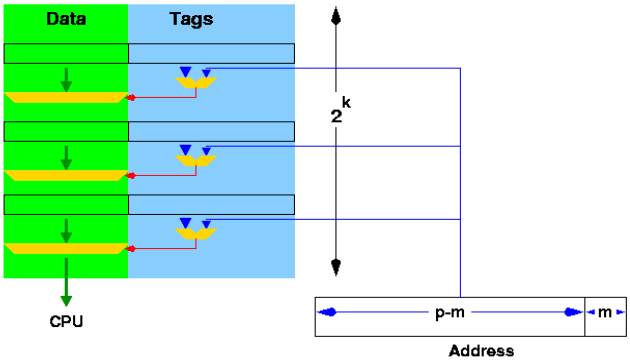


Fig. 1.2. Correspondencia totalmente asociativa.

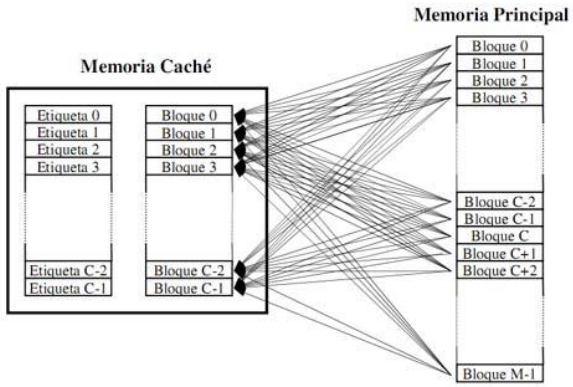


Figura 5.9 Asignación de bloques de la M_p en la M_{ca} con correspondencia totalmente asociativa.

- Correspondencia asociativa por conjuntos entre Mp y Mc

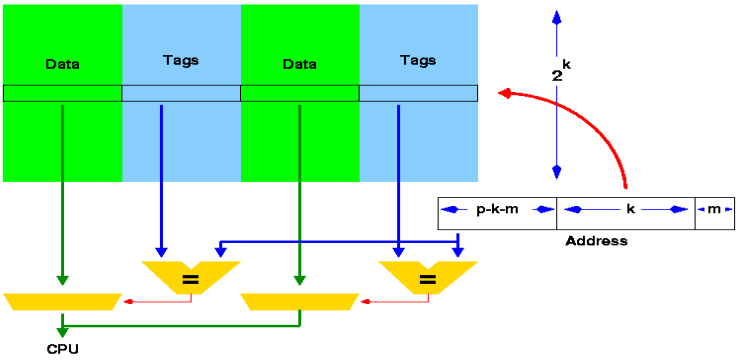


Fig. 1.3. Correspondencia asociativa por conjuntos.

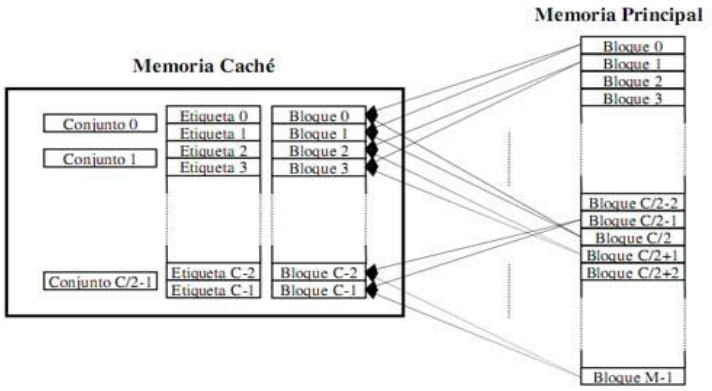


Figura 5.11 Asignación de bloques de la M_p en la M_{ca} con correspondencia asociativa por conjuntos.

Donde la función de transformación está dada por:

$$c = q \times r$$
$$i = j \ Mod \ q$$

c= # de bloques de Mc
q = # de conjuntos en los que se divide la Mc
r = # de bloques por conjunto
i = # de bloque en la Mc
j = # de bloque en la Mp

Desarrollo

La Fig. 1.4 muestra de simulación de bloque por remplazamiento entre la memoria caché y la memoria principal como referencia en: http://www.ecs.umass.edu/ece/koren/architecture/Caché/frame1.htm

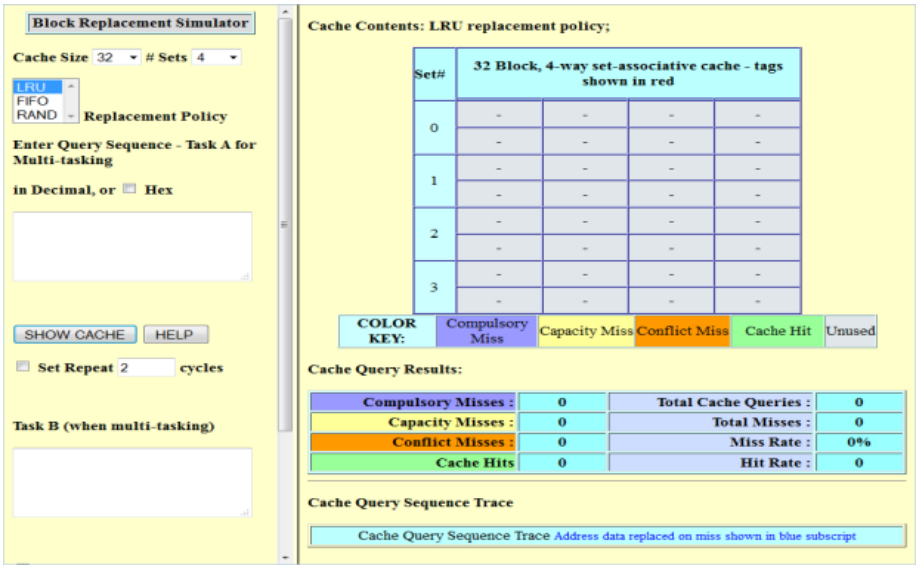


Fig. 1.4. Bloque de simulación de remplazamiento de memoria caché

En ella se debe especificar el tamaño de la memoria caché y el número de bloques (sets) que la conforman. Así mismo se solicita la selección del algoritmo de reemplazamiento y secuencia de consulta:

- LRU: bloque usado menos recientemente
- FIFO: bloque más antiguo en caché
- RAND: bloque regido de manera aleatoria

El área de la derecha nos muestra el tamaño de la memoria caché separada por sets, así como un contador con los diferentes sucesos que se pueden generar:

- Pérdida obligatoria
- Pérdida en conflicto
- Éxito en caché

La Fig. 1.5 muestra la misma ventana programada en Visual Basic 6.0 tratando de igualar las mismas herramientas que tiene el simulador de remplazamiento y los contadores de eventos. Se puede elegir de entre las políticas de reemplazamiento LRU y FIFO junto con el acceso de secuencia de consulta de la memoria principal.

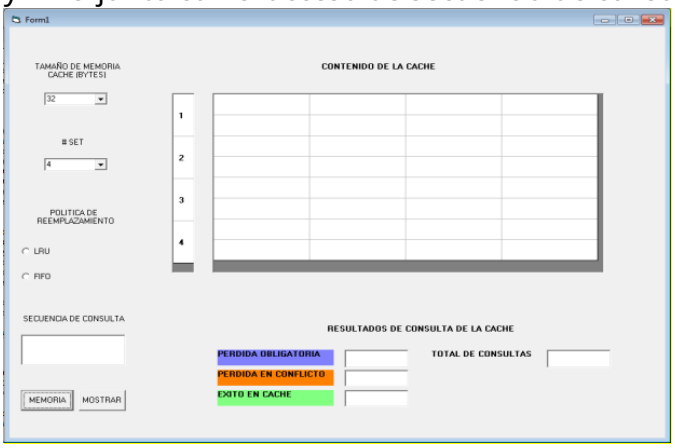


Fig. 1.5. Ventana de remplazamiento

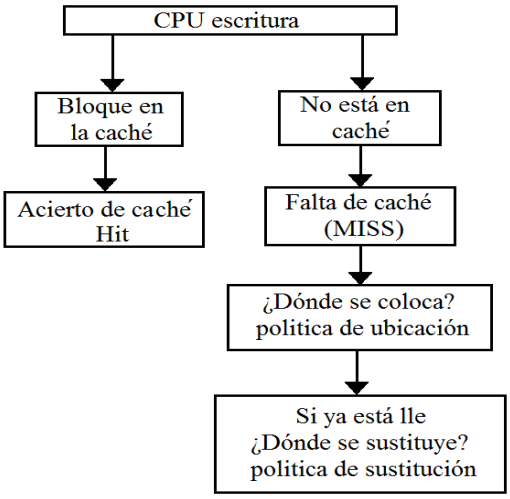
Se muestra el programa y los comentarios sobre los bloques más sobresalientes.

Bloque cero: se define las variables globales del programa así como las matrices de apuntadores para las listas de actualización. Clic para ver anexo

Bloque uno: Se determinan los espacios de memoria caché en ciertos valores, así como los definidos para los sets (combobox) Clic para ver anexo1

Bloque dos: en base a los botones de selección de define el tipo de política de remplazamiento, se lee el valor de memoria para consultar se toma una decisión en

base a si existe o no elementos. El sig. Diagrama a bloques muestra esta toma de decisiones:



Clic para ver anexo2

Bloque tres: definición de distribución de espacio de memoria caché en base al tamaño y número de sets, así como la inicialización de los apuntadores por set para actualización y despliegue de la tabla.

Clic para ver anexo3

A continuación se muestras dos ejemplos con los diferentes algoritmos de remplazamiento:

Bloque más antiguo en caché FIFO: una vez que se cargan consultas consecutivas en el set uno y se llena, se hace una nueva consulta con un mismo valor, provocando un hit por acierto Fig. 1.6. En un siguiente valor para el set 1, tomará la posición más antigua que en este caso en la dirección 0, set 1 que previamente fue un hit, generando sustitución con perdida en conflicto Fig. 1.7.

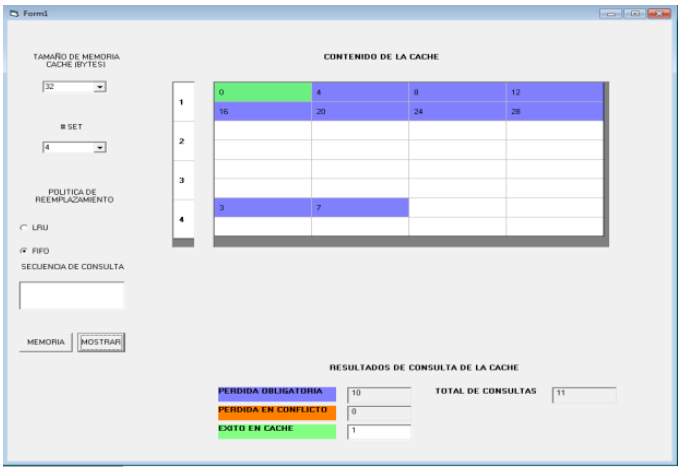


Fig. 1.6. Prueba de FIFO en Mc

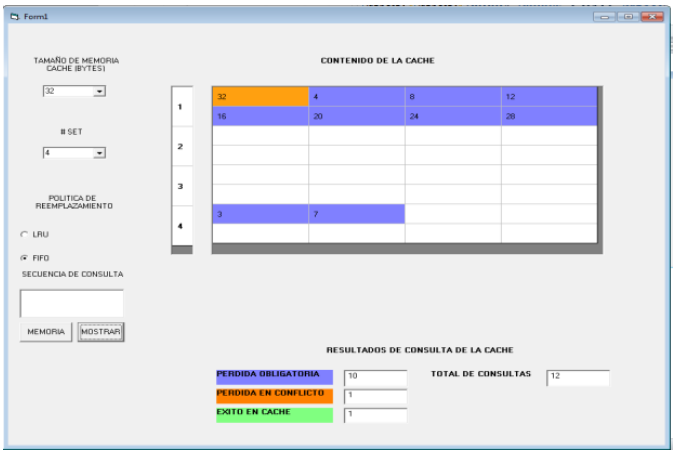


Fig. 1.7. Sustitución por bloque más antiguo en caché.

Bloque usado menos recientemente LRU: en este algoritmo, nuevamente se llena la memoria caché en su primer set y generando un hit en la posición 0, set 1. Fig. 1.8. Posteriormente en la siguiente consulta y debido a que la en la posición 0, set 1 hubo una consulta, se incrementó el uso de ese espacio de memoria, por lo que queda descartada como bloque más antiguo en caché y se toma la subsecuente posición 1, set 1 que tiene el valor de 4 para ser sustituido, generando una perdida por conflicto. Fig. 1.9.

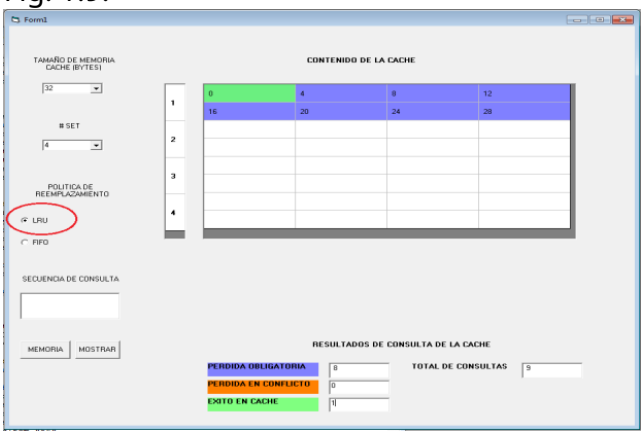


Fig. 1.8. Prueba de LRU en Mc

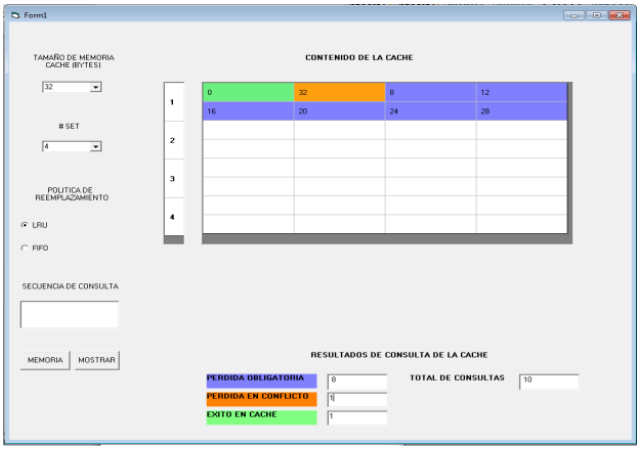


Fig. 1.9. Sustitución por bloque usado menos recientemente

Con el boton de memoria se pueden definir el tamaño y número de espacios de memoria Mc en base al tamaño y número de sets deficidos por el usuario. Así un Mc de 64 bloques y 8 sets se muestra en la Fig. 1.10.

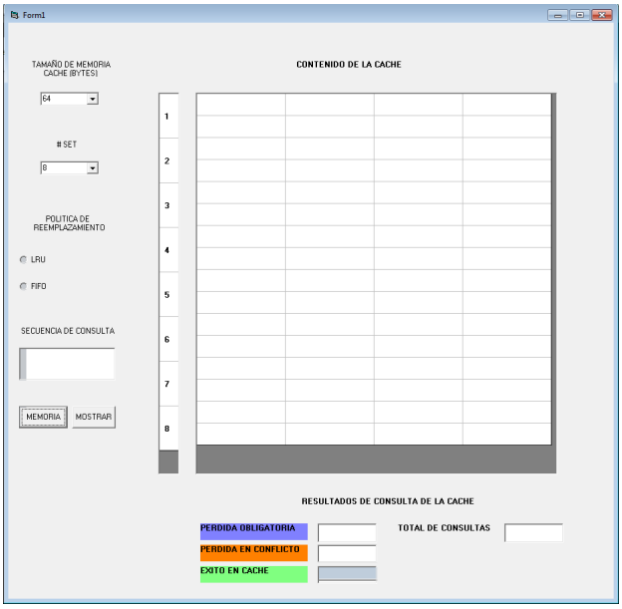


Fig. 1.10. Definición de la Mc en base al espacio y número de sets

CONCLUSIONES

Dado que el sistema caché proporciona un almacenamiento temporal de los recursos solicitados por una aplicación directamente por el usuario, se puede dar el caso que una nueva aplicación solicite ese mismo recurso, devolviéndose este por la caché, evitando así la sobrecarga de los servicios.

La programación de la administración ente la memoria principal y la memoria caché ayuda a entender mucho de su necesidad ya que este ayuda al rendimiento de las aplicaciones, reduciendo así el tiempo necesario para obtener un recurso solicitando que como objetivo ideal es que el tiempo de acceso entre procesador y memoria sea cero.

La forma cómo se realiza el almacenamiento en case depende de su organización, la cual consiste de establecer la función de correspondencia que asigna a los bloques de la memoria principal posiciones definidas en la memoria caché.

Finalmente es importante conocer la forma en cómo trabaja la memoria caché dentro de la arquitectura de una computadora como un principio básico además de otros puntos como el funcionamiento del procesador y la forma en cómo se comunica con la memoria y los periféricos de entrada salida.