Caso 2 – Infraestructura Computacional Sección 01

Integrantes:
Andres Arévalo Fajardo - 201923853
Luccas Rojas Becerra - 201923052
Alejandro Salgado – 201923134

1. Descripción Estructuras de Datos

La implementación del caso de estudio 2 es comprender a partir de una simulación en Java el comportamiento básico de traducción de direcciones virtuales a reales y la carga de datos haciendo uso de la memoria virtual. Por lo tanto, es importante antes de comenzar a simular el manejo de las estructuras, definir cuáles son estas y la relación entre las mismas.

A. *Definición y funcionamiento teórica de las estructuras de datos*

RAM: Hace referencia a la memoria real, donde cada vez que un proceso necesita de un recurso se almacena la página, y se le asigna una dirección real. Este es un recurso que es limitado y debe ser administrado correctamente para asegurar la ejecución correcta de los pedidos y procesos que se le hagan. Por medio de la virtualización de la memoria se logran ventajas en el uso del recurso para compartir el recurso, facilitar el manejo de direcciones, entre otras. La unidad de almacenamiento de la RAM es un marco de página

TP: También conocida como tabla de páginas, es un componente que se encuentra en la RAM. Este indica la correspondencia entre las páginas de memoria virtual asignado a un proceso y los marcos de página en la RAM que lo almacenan. Así mismo, la TP tiene conocimiento de si la página se encuentra cargada en la memoria real, ya que si no se encuentra no tiene asignado un valor definido la entrada, lo cual es esencial para la implementación realizada.

TLB: Su nombre es Translation Lookaside Buffer, también conocida como la zona de memoria caché. Es una zona de memoria que permite almacenar las traducciones recientes. El tiempo de respuesta para la búsqueda de una dirección que se encuentre en la memoria cache se reduce considerablemente a si busca en la tabla de páginas.
La razón de usar una memoria cache es gracias al principio de localidad, donde se asume que si una pagina ha sido usada recientemente es muy posible que se vuelva a usar en el futuro cercano.
Finalmente, se entiende que la TLB almacena tanto la dirección virtual como la dirección virtual de las últimas traducciones.

Así mismo, es importante entender como funcionan los algoritmos de cambio de paginas que se van a usar en el caso de estudio:

Algoritmo FIFO: Este es un algoritmo que se conoce como “First in First out”. Por su nombre se entiende que lo primero que entre es lo primero que en el momento de que se llene se va a salir. Este algoritmo se implementa sobre las traducciones de la TLB, por lo tanto se entiende que cuando una traducción se almacena en la TLB esta se va a poner en una cola en el siguiente puesto que este disponible, y a medida que se vaya necesitando guardar una nueva que no este almacenada ya, se debe retirar la primera de la cola.

Algoritmo de Envejecimiento: El algoritmo de envejecimiento es aquel que toma en cuenta no solo el orden en el que se ingreso por primera vez una pagina, sino que ademas de eso tiene presente cuantas veces ha sido referenciada desde que se encuentra cargada. Este algoritmo se implementa para manejar el almacenamiento de las paginas en la memoria RAM. Es importante entender como funciona el contador de envejecimiento:
El contador de envejecimiento tiene un numero de bits, usemos por ejemplo 5 bits. Siempre que una pagina se carga se inicializa con un 1 a la izquierda: 10000
De igual manera, cada ciclo de reloj se va “envejeciendo” el contador de cada uno de las paginas guardadas en el momento, por lo que para hacer esto se realiza un corrimiento a la derecha del valor: 01000
(Importante recordar que el envejecimiento se aplica a TODAS las paginas que estén en RAM)

Cuando una pagina se referencia, estando ya en la memoria RAM, se le agrega un 1 a la izquierda a lo que ya se tenga: 11000
Finalmente, cuando una pagina nueva se vaya a ingresar, y no haya espacio en la RAM, se recorre la lista de datos en RAM y se elimina la pagina con el menor contador (Esta pagina cuando vuelva a ingresar no tiene en cuenta los bits de antes, solo desde que entro).

B. Definición e implementación simulación de la estructura de datos

Primero que todo, es importante definir como se hizo para simular las estructuras de datos descritas anteriormente. Esto es teniendo en cuenta que en el enunciado se indica que no es necesario actualizar el estado del área de SWAP, sino solamente modelar la TP y la TLB

TP y RAM: La TP, como se explicó antes, tiene la relación de direcciones virtuales con los marcos de página que almacenan la información en la memoria real. Por lo tanto, la TP conoce cuales paginas se encuentran en la memoria real y, así mismo, a cuáles marcos de página y direcciones virtuales de la memoria RAM se encuentran asignadas.
Por lo anterior, decidimos modelar únicamente la TP, ya que esta conoce los marcos de página en RAM. Así mismo, decidimos usar como estructura de dato una HashMap de Java, la cual es una relación de llave (Numero de pagina referenciada) y valor (Envejecimiento). Se uso como valor el numero que indica el envejecimiento de una pagina en la RAM, el cual se explicara como algoritmo después; esto ya que no es necesario almacenar ningún dato sino que se simula obtener la pagina, por lo que aumenta la eficiencia de la implementación.

TLB: La TLB, primero que todo, se modelo como un HashMap. En el HashMap se manejo que la llave sea la posición de 0 hasta el tamaño de la TLB y que el valor sea la pagina que hace referencia, simulando que la TLB contiene la traducción entre dirección virtual y la dirección real. Sin embargo, se usa un algoritmo FIFO para determinar que traducciones se mantienen en la TLB, por lo que se utilizó además del Hash una cola Queue, para manejar y tener cuenta del orden de ingreso de las traducciones a la TLB.

Ahora, teniendo en cuenta como se decidió implementar las estructuras a simular, es importante también tener en cuenta como se plantearon los threads que van a permitir simular el proceso de carga de referencias de un proceso:

1. El primer thread que se usa es el del main, el cual se encarga en general de cargar las referencias y actualizar el estado de la TLB y de la TP. Inicialmente este thread lee el archivo y guarda las referencias en un arreglo. A partir de esto el thread se encarga de revisar la TLB y la TP para asi determinar si el dato debe ser cargado o simplemente referenciado, ademas este es el que determina cuanto se le agrego a los tiempos de lectura y de traducciones. Este thread corre cada 2 milisegundos, para simular que corre cada 2 ciclos de reloj. Este thread es el que se ocupa de la mayoría del trabajo.
2. El segundo thread tiene una implementación muy básica pero esencial para el correcto funcionamiento de la RAM. Este se encarga de, cada 1 milisegundo (1 ciclo de reloj), hacer el corrimiento a la derecha, por lo que permite que cuando el otro thread necesite eliminar una pagina de la RAM se haya aplicado el algoritmo de envejecimiento y se retire el dato más viejo. Por lo tanto, también se entiende que por cada vez que se carga una referencia se ha aplicado el algoritmo de envejecimiento 2 veces.

También es importante tener en cuenta que para medir el tiempo de simulación tanto de carga de datos como de traducción de direcciones se manejan 2 contadores diferentes, a los cuales se les van sumando los tiempos respectivos con respecto a la operación que se realice al referenciar la pagina.

C. Simulación Paso a Paso de llamado de las referencias

Para entender como funciona la implementación del llamado de referencias consideramos que es mejor evaluar todos los casos posibles y que es lo que ocurre en el algoritmo realizado, por lo tanto a continuación se presentan todos los casos:

¿Qué ocurre si se llama una referencia que no se encuentra en la RAM y esta tiene espacio al igual que la TLB?

Lo primero que hace el algoritmo cuando se ingresa una referencia es llamar a un método que consulta si la pagina se encuentra en RAM. En el caso de que no este, se llama a un método que modifica la TP, donde cambia el valor de la llave correspondiente a la pagina referenciada, donde pasa de tener el contador de envejecimiento en Nulo a iniciarlo con un 1 a la izquierda (Sumándole 2 a la 32). Teóricamente esto representa que ocurrió un fallo de pagina, donde es necesario obtener la pagina de disco y que se encuentre en la memoria real.

Así mismo, se guarda, en una de las llaves disponibles de la TLB la pagina que acaba de ser solicitada. Al mismo tiempo, se agrega a la cola en su siguiente posición.

Al ocurrir esto se le agrega un valor de 60 nanosegundos al contador de tiempo de traducción, que representa 2 llamados a la memoria de real porque ocurre un fallo de pagina.

Así mismo, se le agrega un valor de 10 milisegundos al contador de tiempo de carga, que es el tiempo que se demora en resolver un fallo de pagina.

¿Qué ocurre si se llama una referencia que no se encuentra en la RAM y esta no tiene espacio?

Nuevamente, al entender que no esta en RAM se tiene que agregar a memoria real, pero primero es necesario eliminar alguna pagina y devolverla a disco. Por lo tanto, se tiene que revisar cual es la pagina con el menor contador de envejecimiento, por lo que se recorre todo el arreglo y se encuentra la pagina a quitar de disco. Después de removerla de disco se agrega la nueva referencia y se inicializa el contador de envejecimiento.

Así mismo, la pagina que se quite de RAM tambien se tiene que retirar de la TLB, ajustando la cola acorde a cual fue la eliminada. Mientras que la que se acaba de referenciar se agrega al HashMap y al contador de la TLB. Por lo tanto, se entiende que una pagina solo puede estar traducida en la TLB si tambien esta en la RAM.

Se le agrega nuevamente un valor de 60 nanosegundos al contador de tiempo de traducción y 10 milisegundos al contador de tiempo de carga.

¿Que ocurre si se llama una referencia que se encuentra en la RAM pero no en la TLB que tiene espacio?

En este caso nuevamente se consulta y obtiene que la RAM contiene la pagina referenciada, por lo tanto no es necesario que ocurra un fallo de pagina. Sin embargo, toca llevar a cabo un paso extra, donde se va a agregar la dirección real a la TLB como valor de alguna de las llaves vacías. Así mismo, se va a agregar a la cola la referencia virtual, llave de la TLB, a la cola de esta (Para poder aplicar el algoritmo FIFO). De igual manera, se le suma un 1 a la izquierda del algoritmo de envejecimiento.

Se agregan 30 nanosegundos al contador de tiempo de traducción, que representa el llamado a la memoria real.

Se agregan 30 nanosegundos al contador de tiempo de carga, que representa el tiempo que se demora en cargar un dato que se encuentra en la RAM.

¿Qué ocurre si se llama una referencia que se encuentra en la RAM pero no en la TLB que no tiene espacio?

La única variación del anterior caso es que, asi como en el 2do caso, es necesario eliminar una pagina de la TLB. Sin embargo, en este caso toca eliminar la primera traducción en la cola de la TLB, dando paso al uso del algoritmo FIFO. Así mismo, se agrega la referencia al HashMap y la cola. De igual manera, se le suma un 1 a la izquierda del algoritmo de envejecimiento.

En este caso, nuevamente se agregan 30 nanosegundos tanto al contador de tiempo de traducción y el tiempo de carga.

¿Qué ocurre si se llama una referencia que se encuentra en la TLB?

Si se llama a una referencia que esta en la TLB no es necesario actualizar la TLB. Sin embargo, teniendo en cuenta que debe estar presente en la RAM se debe sumar al contador de envejecimiento en la RAM.
Asi mismo, se le agregan 2 nanosegundos al tiempo de traducciones y 30 nanosegundos al tiempo de carga debido a que es el tiempo que se demora en cargar el dato que se encuentra en RAM.

D. Manejo de la sincronizacion en el proyecto

Es necesario tener presente que se usan 2 threads en el proyecto, los cuales van a trabajar de forma concurrente, por lo que es necesario implementar el proyecto de cierta manera que se maneje la TLB y la TP de forma concurrente.
Inicialmente, fue necesario implementar la concurrencia entre threads comprendiendo cada cuanto debe trabajar un thread. Se indica que el thread de envejecimiento trabaja cada milisegundo, por lo que se puso un sleep de un milisegundo antes de que se aplique el corrimiento a todos los valores que se encuentren en la RAM. En cambio, el thread principal que actualiza el estado de la TP y la TLB corre cada 2 milisegundos, por lo que se puso un sleep de 2 milisegundos antes de que se intente cargar cada referencia.
Finalmente, fue necesario establecer que todos los métodos tanto de la TP como de la TLB estuvieran sincronizados, ya que son objetos que van a revisar y manipular los threads, por lo que debe sincronizarse el uso de estos.

2. Resultados Implementación:

A.Caso de prueba logicos

Archivo de 1051 refencias, con llamado a una unica referencia:

```
Ingrese el n?mero de entradas de la TLB:
4
Ingrese el n?mero de marcos de pagina de la RAM:
16
Ingrese el nombre del archivo, sin el .txt, del cual se obtendran las referenci
as (Esta almacenado en la carpeta data):

Referencia Unica
Se obtuvo el archivo: Referencia Unica.txt
La carga de referencias fue exitosa

Tiempo carga: 10031470
Tiempo traduccion: 2158
```

64 Marcos de pagina para la RAM:

```
Ingrese el n?mero de entradas de la TLB:
8
Ingrese el n?mero de marcos de pagina de la RAM:
64
Ingrese el nombre del archivo, sin el .txt, del cual se obtendran las referenci
as (Esta almacenado en la carpeta data):

ej_Alta_64paginas
Se obtuvo el archivo: ej_Alta_64paginas.txt
La carga de referencias fue exitosa

Tiempo carga: 590028950
Tiempo traduccion: 11070
(base) Air-de-Andres-6:InfracompCaso2 andresarevalo$
```

64 Numero de entradas para la TLB:

```
Ingrese el n?mero de entradas de la TLB:
64
Ingrese el n?mero de marcos de pagina de la RAM:
32
Ingrese el nombre del archivo, sin el .txt, del cual se obtendran las referenci
as (Esta almacenado en la carpeta data):

ej_Baja_64paginas
Se obtuvo el archivo: ej_Baja_64paginas.txt
La carga de referencias fue exitosa

Tiempo carga: 2760022440
Tiempo traduccion: 18056
```

B. Tablas y graficas Tiempos:

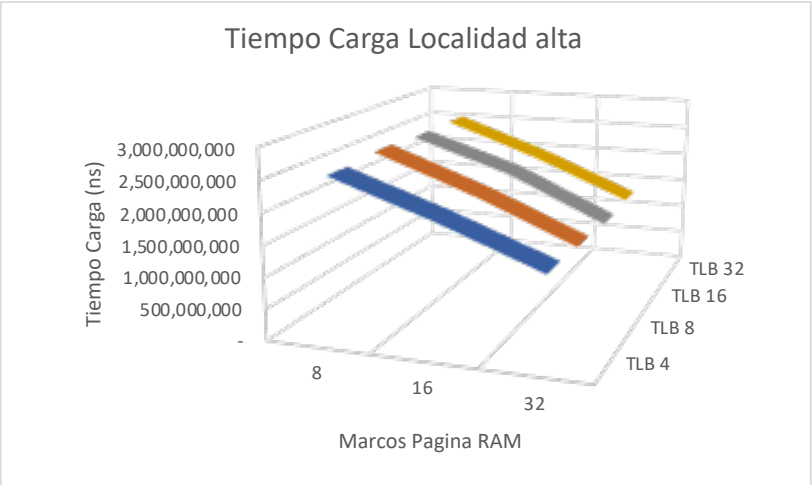
Localidad Alta:

Tiempo de Carga Localidad Alta			
TLB/RAM	**8**	**16**	**32**
4	2,530,023,130	1,980,024,780	1,360,026,640
8	2,580,022,980	1,990,024,750	1,310,026,790
16	2,530,023,130	2,010,024,690	1,270,026,910
32	2,540,023,100	1,970,024,810	1,300,026,820

Figura 2.1

Dif porcentual Tiempo Carga Localidad Alta			
TLB/RAM	**8**	**16**	**32**
4	0.00%	-2.06%	-16.24%
8	-1.98%	-2.58%	-11.97%
16	0.00%	-3.61%	-8.55%
32	-0.40%	-1.55%	-11.11%

Figura 2.2

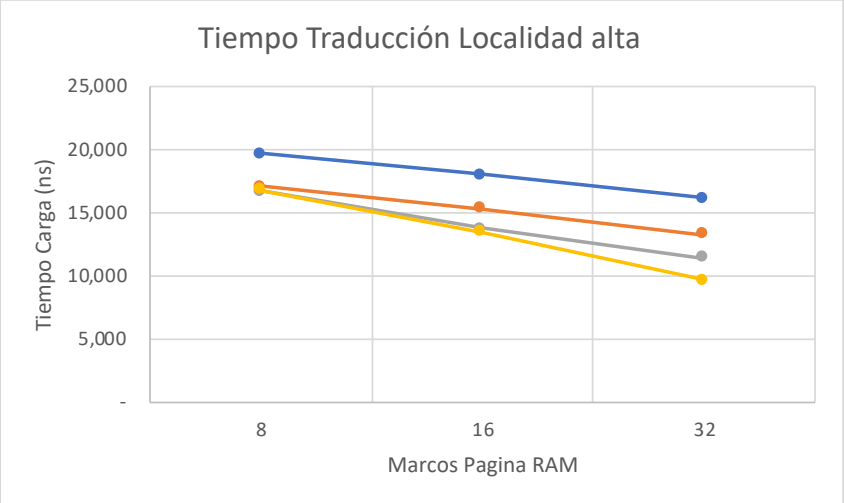


Tiempo de Traducción Localidad Alta			
TLB/RAM	**8**	**16**	**32**
4	19,634	17,984	16,124
8	17,012	15,270	13,230
16	16,722	13,706	11,346
32	16,780	13,474	9,588

Figura 2.3

Dif porcentual Tiempo traducción Localidad Alta			
TLB/RAM	**8**	**16**	**32**
4	1.83%	0.09%	-0.78%
8	2.79%	-1.80%	5.50%
16	3.34%	2.10%	-3.15%
32	3.01%	3.76%	4.12%

Figura 2.4



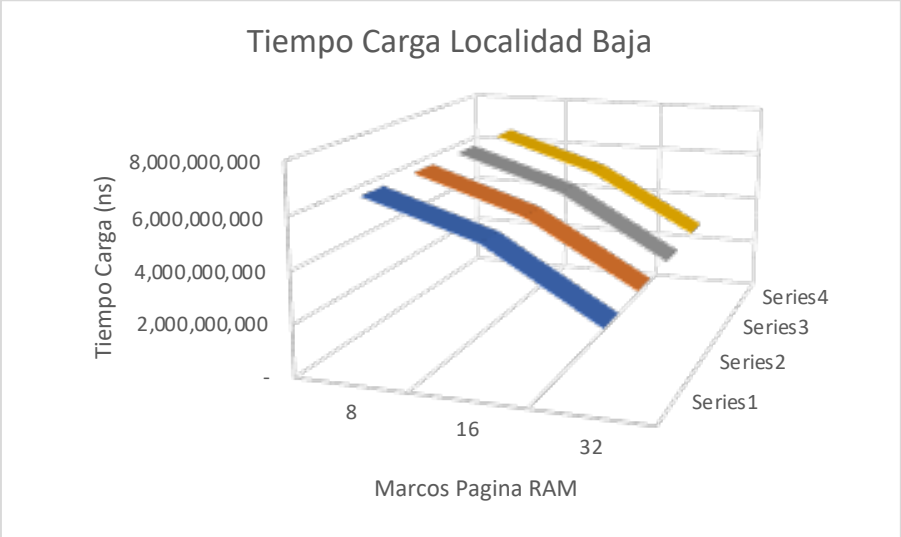
Localidad Baja:

Tiempo de Carga Localidad Baja			
TLB/RAM	8	16	32
4	6,630,010,830	5,300,014,820	2,710,022,590
8	6,630,010,830	5,250,014,970	2,790,022,350
16	6,630,010,830	5,270,014,910	2,790,022,350
32	6,640,010,800	5,290,014,850	2,910,021,990

Figura 2.5

Dif porcentual Tiempo Carga Localidad Baja			
TLB/RAM	8	16	32
4	-0.15%	0.00%	1.45%
8	-0.15%	0.94%	-1.45%
16	-0.15%	0.57%	-1.45%
32	-0.30%	0.19%	-5.82%

Figura 2.6

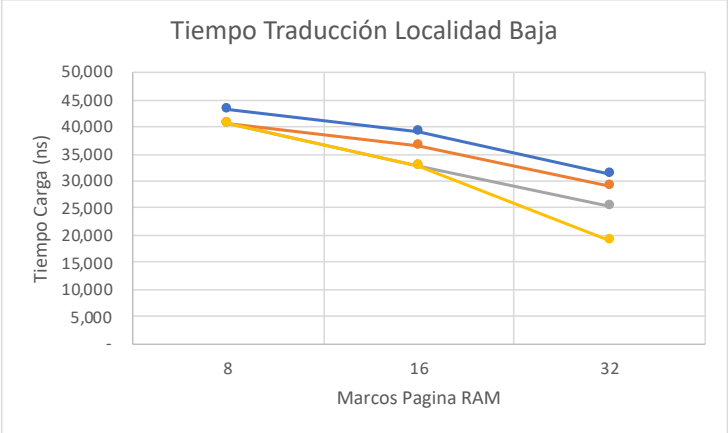


Tiempo de Traducción Localidad Alta			
TLB/RAM	8	16	32
4	43,050	39,060	31,290
8	40,502	36,418	29,038
16	40,502	32,614	25,230
32	40,560	32,730	18,926

Figura 2.7

Dif porcentual Tiempo traducción Localidad Baja			
TLB/RAM	8	16	32
4	2.16%	-0.15%	2.22%
8	1.21%	1.57%	-0.13%
16	1.21%	4.08%	2.96%
32	1.07%	2.30%	5.37%

Figura 2.8



C. Explicación resultados:

Procesos logicos frente a fallos de pagina:
Se evidencia efectivamente el buen comportamiento de las estruturas de datos para la simulación del proceso de virtualización de la memoria. En este sentido, dados los resultados en los tiempos se puede observar que cada vez qeu se realiza un fallo de pagina tambien se geenra una carga de datos a RAM. Por otro lado, se evidencia que la simulación minimiza los fallos de pagina, es decir, realiza solo los necesarios, esto se puede observar, debido a que cuando los marcos de pagina son igual a 64, solo se generan 64 fallos de paginas, es decir, se generan los fallos de pagina correspondiente a la pirmera vez que se cargan los datos a RAM por cada referencia. Esto último tambien es evidenciable en el caso de que solo se haga alución a una unica referencia, solo se genera un fallo de pagina correspondiente a la primera (y unica vez) que se cargan los datos a RAM para la referencia. En general, el comportamiento ilustrado por la simulación es el adecuado y las diferencias porcentuales se deben a factores estocasticos en el algoritmos de envejecimiento y los tiempos de ejecución sobre la concurrencia.

Tiempo de carga Constante independiente a la TLB:
La TLB posee una cota respecto a los marcos en RAM. Es decir, dado que la TLB contiene direcciones a RAM, si alguna referencia no esta en RAM no puede estar en la TLB. De este modo, asi las entradas de la TLB sean 64, esto no indica que no se generen fallos de pagina, por el contrario, lo que sucede es que si la RAM posee marcos de pagina menores a 64, entonces el espacio total de la TLB no se va a aprovechar al máximo. En conclusión, los fallos de pagina no dependen de las entradas en la TLB, dependen exclusivamente de los marcos de pagina que posea la RAM.

Cambio en el tiempo de traducción, dada una RAM y una TLB:
Es evidente que a mayor cantidad de entradas, se disminuye el tiempo de traducción. No obstante, esta tendencia se ve acotada según la cantidad de marcos de pagina que posea la memoria RAM. Es decir, si se evidencia la tendencia en los tiempos de traducción, cuando el numero de entradas es menor o igual al marco de paginas, el tiempo de traducción se reduce en magnitud, pero cuando las entradas superan a los marcos el tiempo de traducción se adquiere una tendencia constante. Esto argumenta, bajo otra perspectiva, lo enunciado en el literal anterior. En este orden de ideas, no es beneficioso poseer una TLB con mayor numero de entradas que marcosde paginas de memoria RAM.

A mayor RAM, menor tiempo de carga:
Con una mayor cantidad de marcos de pagina es evidente un menor tiempo de carga. Esto se debe a que, con mayor cantidad de marcos, mayo cantidad de referencias la memoria RAM puede guardar, en consecuencia, hay una menor cantidad de fallos de paginas y un menor tiempo utilizado en cargar datos de disco a memoria RAM. Ademas, es evidenciable que mayor marcos de paginas en referencias con localidad baja, disminuye en mayor medida el tiempo de cara a comparación de referencias con lacidad alta. En este sentido, es deseable tener una memoria RAM lo suficientemente grande para reducir la cantidad de fallos de pagina, no obstante, eso mejora el recurso pero aumenta los costos de este. De esta forma, apoyarse en buenos algoritmos de

envecimiento para reducir la cantidad de fallos de pagina, partiendo del principio de localidad parese ser la estartegia más viable.