Alejandro Tovar - 201512531
Melissa Contreras - 202011876
Juan Sebastián Hoyos - 201822167

Caso de Estudio 2 – Memoria virtual

Tabla de contenido

Justificación estructuras de datos

Descripción de las estructuras de datos usadas para simular el comportamiento del sistema de paginación y cómo se usa dichas estructuras (cuándo se actualizan, con base en qué y en qué consiste la actualización).

Para el comportamiento del sistema de paginación simulamos la memoria real con un HashMap con llave y valor de tipo Long. La llave es el número de página y el valor los bits de referencia. Esta estructura de datos se actualiza cada vez que se referencia una página, nueva o antigua, ya que debe estar reflejado en los bits de referencia. Para una nueva página, se busca si hay espacio y si no se busca la página que menos se ha referenciado; Luego, se borra la llave y el valor elegido y se inserta la nueva página con sus bits de referencia correspondiente. Para una página antigua, se pone 1 en la posición que corresponde al número de página en el arreglo *rbits*, el cual se explicará más adelante, y así se actualizará los bits de referencia de esa página.

La tabla de páginas la simulamos haciendo un HashMap con el mismo tipo de datos y llave de memoria real, solo que el valor en este caso es el número de marco. En nuestra simulación el número de marco es la misma página ya que no podemos determinar en el HashMap en que posición esta. Esta estructura de datos se actualiza cada vez que la memoria real se actualiza cuando llega una nueva página. La actualización puede consistir en borrarle el marco a una página y/o poner el marco correspondiente a una página.

Para la TLB hicimos un ArrayList de tipo Long que nos indicaba a qué página recientemente habíamos accedido. Esta estructura se actualizaba cada que la página sí se encontraba en la tabla de páginas o si se agregaba a la memoria real. La actualización se basaba en el algoritmo FIFO. Si el tamaño del ArrayList era menor al indicado por el usuario significaba que había espacio en la TLB y simplemente se agrega el número de la página. De lo contrario, si la TLB estaba llena, se eliminaba el primer elemento del ArrayList y se agregaba la nueva página al final.

También, hicimos un arreglo *rbits* de tipo int de tamaño 64 (Número de páginas) que nos indicaba con 1s y 0s que página había sido referenciada en 1 tick, siendo la página una posición en el arreglo. Esta estructura se realizó con el fin de que el algoritmo de envejecimiento supiera que páginas se habían referenciado, actualizar la memoria real y hacer el corrimiento de un bit.

Esquema de sincronización

Esquema de sincronización usado. Justificación breve sobre dónde es necesario usar sincronización y por qué.

Se realizó exclusión mutua por medio de monitores. Es necesario usar sincronización sobre la estructura de datos correspondiente a la memoria real en los dos threads en el momento en que se utilice. La razón es que cada thread debe tener los datos de memoria real actualizados y debido a que ambos lo afectan cada que se ejecutan, debemos evitar cualquier fallo haciendo que los dos threads no accedan al tiempo a esa estructura.

Tabla con los tiempos recopilados

Tabla con los tiempos de carga de Datos

Proceso con localidad Alta:

TLB	TLB - 4	TLB - 8	TLB - 16	TLB - 32	TLB - 64
RAM - 8	19934	17070	17070	17070	17070
RAM - 16	18194	15506	13938	13764	13938
RAM - 32	15914	12968	11594	9298	9472

Proceso con localidad Baja:

TLB	TLB - 4	TLB - 8	TLB - 16	TLB - 32	TLB - 64
RAM - 8	43230	40792	40850	40908	40908
RAM - 16	39270	36806	33310	33136	33252
RAM - 32	31200	28556	25218	17592	18404

Tabla con los tiempos de traducción de Direcciones Virtuales

Proceso con localidad Alta:

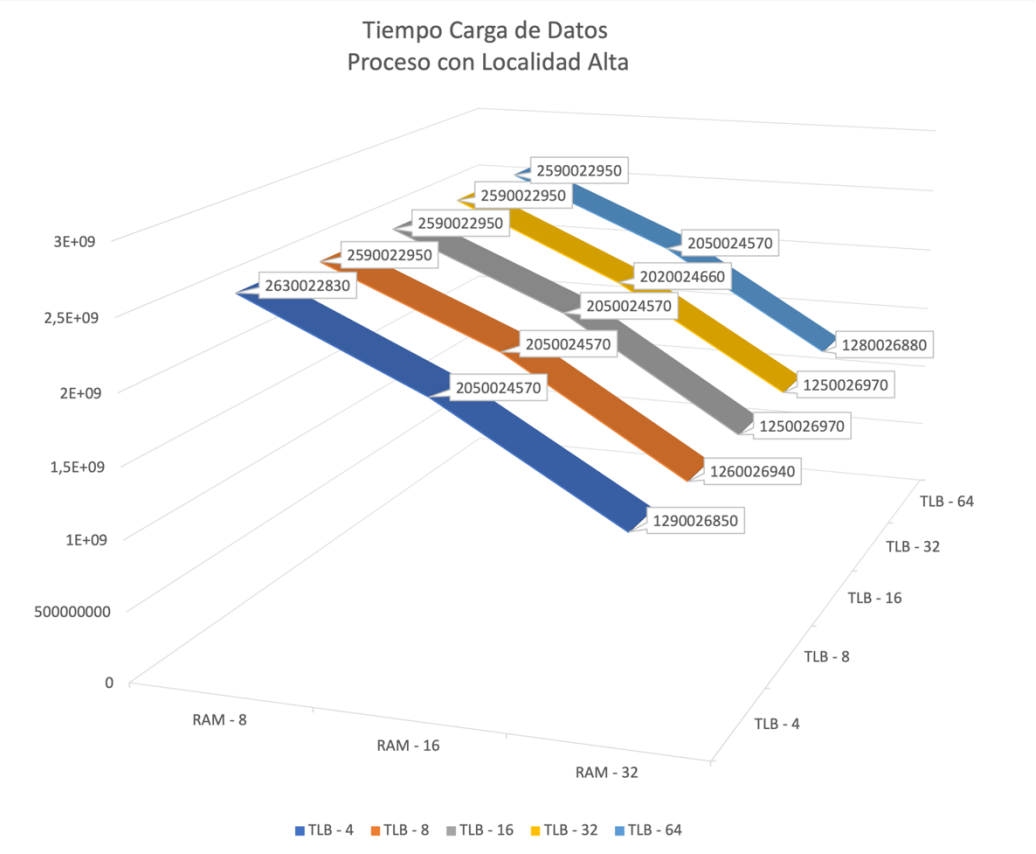
TLB	TLB - 4	TLB - 8	TLB - 16	TLB - 32	TLB - 64
RAM - 8	2630022830	2590022950	2590022950	2,59E+09	2,59E+09
RAM - 16	2050024570	2050024570	2050024570	2,02E+09	2,05E+09
RAM - 32	1290026850	1260026940	1250026970	1,25E+09	1,28E+09

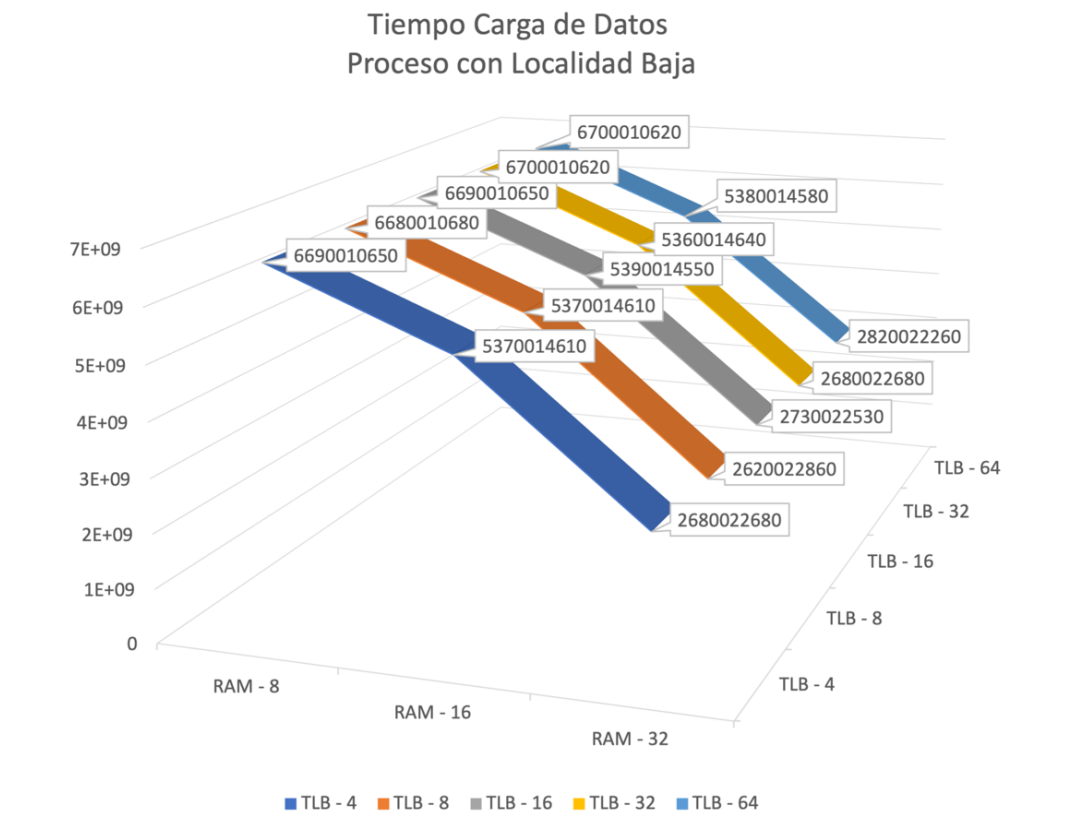
Proceso con localidad Baja:

TLB	TLB - 4	TLB - 8	TLB - 16	TLB - 32	TLB - 64
RAM - 8	6690010650	6680010680	6690010650	6700010620	6700010620
RAM - 16	5370014610	5370014610	5390014550	5360014640	5380014580
RAM - 32	2680022680	2620022860	2730022530	2680022680	2820022260

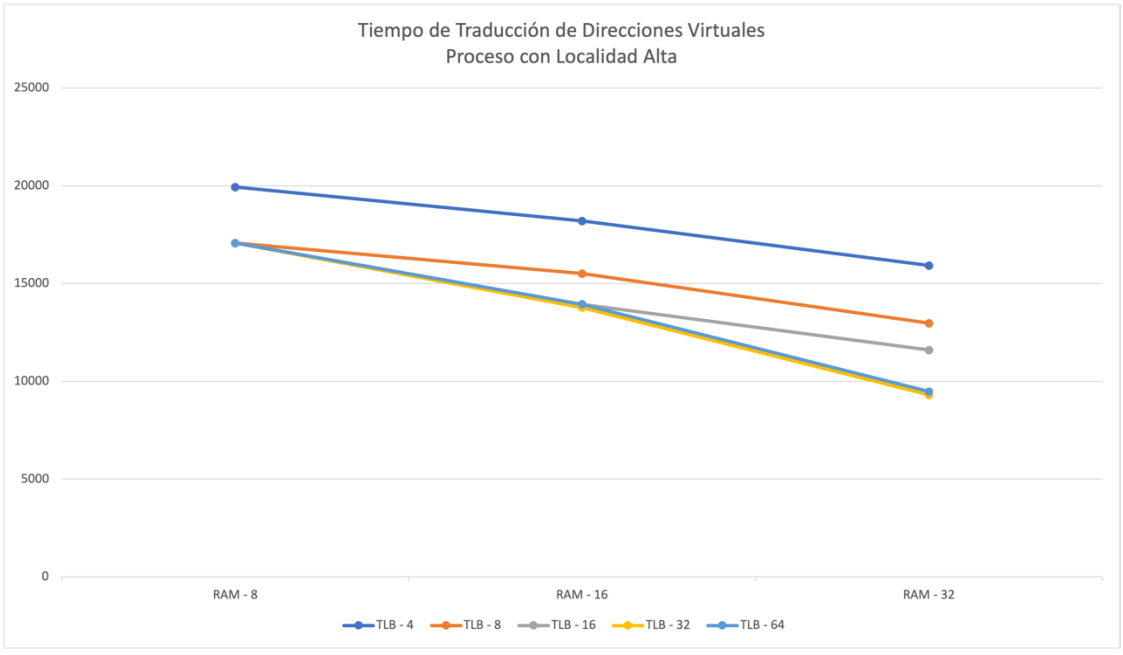
Gráficas

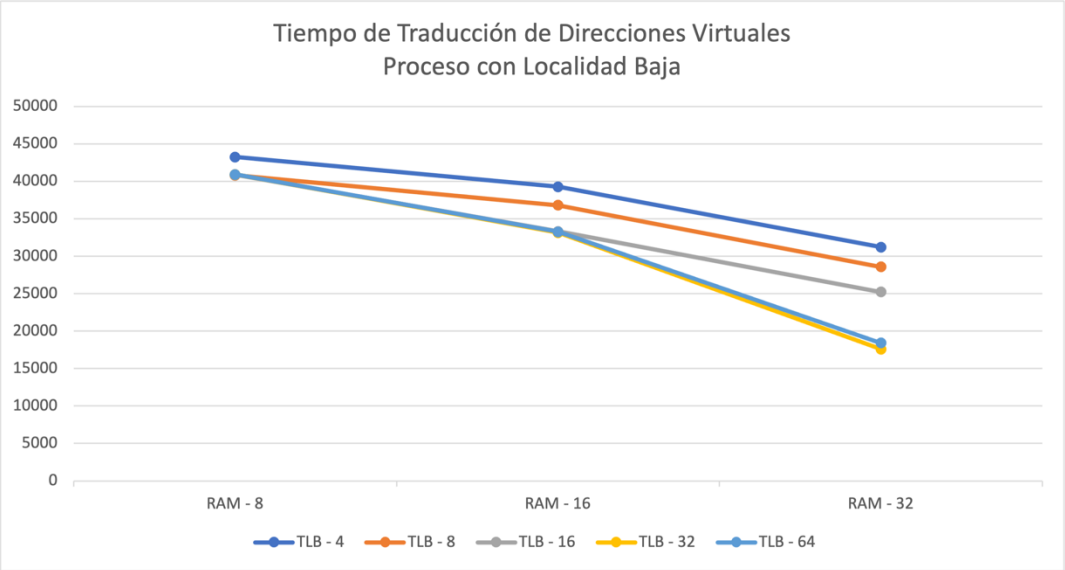
Marcos de página asignados vs. TLB vs. Tiempo de carga de datos





Marcos de página asignados vs. TLB vs. Tiempo para resolver direcciones





Otros casos

El tiempo de ejecución va a depender primariamente de la cantidad de fallos de página, pues al tener más fallos de página, con los datos que están previstos por el enunciado, se espera que se vaya a tener más tiempo de ejecución. Por otra parte, para que estos fallos de página sean altos, depende de que el tamaño de la RAM sea bajo. Cumpliendo esta condición, se va a tener un tiempo de ejecución mayor.

Interpretación de los resultados

Después de haber corrido el proyecto y haber probado varías configuraciones y distintos datos, podemos deducir de esto que:

- La cantidad de entradas en la TLB no afectan drásticamente el tiempo de procesamiento, pues sin importar si teníamos más o menos entradas en la TLB, el tiempo de procesamiento dependía principalmente del tamaño de la RAM.

- En un cierto punto, el tamaño de la RAM también empezaba a ser insignificante, pues al haber realizado las pruebas con 32 y 64 en la RAM, los resultados fueron casi iguales, la diferencia ya no era tan notoria.