|  |  |
| --- | --- |
| Imagen que contiene Texto  Descripción generada automáticamente | **Universidad de los Andes**  **Departamento de Ingeniería de Sistemas y Computación**  Santiago Hernández – 201922432  Dionny S. Cárdenas - 201921484 |

Caso 2:

## **Estructura de datos**

Para la realización del caso, se tuvieron que usar 2 estructuras de datos principales, ya que teníamos que representar tanto la tabla de páginas como la memoria física. Para la representación de la tabla de páginas se optó por usar una matriz, en donde en índice de cada fila es la página asignada. Esta matriz cuenta con 4 columnas en donde la primera (matriz[página][0]) representa el marco de página de la memoria física, esta es la columna que se actualiza constantemente cada vez que toca asignar un marco de página nuevo a la página, las páginas a las cuales se le retira el marco para dárselo a otra página, reciben el valor de -1, la segunda columna(matriz[página][1]) indica el bit de referencia de acuerdo con el algoritmo “páginas no usadas recientemente”, la tercera (matriz[página][2]) indica el bit de modificación según lo indicado por el algoritmo “páginas no usadas recientemente”, finalmente la última columna (matriz[página][3]) hace referencia a la suma de los 2 bits anteriores teniendo en cuenta el peso de los valores que representan las diferentes clases indicadas en el algoritmo.

La matriz en ejecución se ve de la siguiente forma:

Texto

Descripción generada automáticamente con confianza media

Esta es para la configuración en donde hay 16 páginas, por eso hay 16 filas en la tabla. El algoritmo para saber a cuál página sacar utiliza la última columna, eso se explicará más a detalle en la parte de sincronización.

Además de esta matriz, se usó una tabla de hash para representar la memoria física, se decidió usar una tabla de hash ya que esta retorna los valores de forma constante, lo cual es necesario para acceder de manera rápida y eficiente a los marcos de página y a las páginas que tiene, además de saber si una página está o no en la estructura. La llave de la tabla es el marco de página y el valor la página virtual, esta tabla se utiliza de muchas veces durante el programa, se utiliza para saber si una página ya se encuentra o no en la memoria física, para de esta forma saber si genera fallo de página o no, adicionalmente se usa para llevar un registro de que valor de página es asignado a cada marco, este valor se va actualizando en base a lo que el algoritmo “páginas no usadas recientemente” considere.

Un ejemplo de cómo se ve:



Esto es, para la configuración con 8 marcos de página.

## **Sincronización:**

Para la parte de la sincronización se utilizaron 2 clases que la manejaran. Como necesitábamos usar las 2 estructuras de datos previamente mencionadas, se crearon clases exclusivas para cada una de las estructuras, para que esta clase controlara todos los métodos que se hagan sobre las estructuras, Estas clases llevan los métodos synchronized.

El nombre de las clases son TablaPaginas para la encargada de manejar la matriz:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Y para la tabla de hash, la clase se llama MemoriaFisica:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Esto se decidió ya que vimos que era la mejor forma de controlar los accesos independientes a las estructuras de datos, ya que son accedidas por varios threads.

Los 2 threads (ReemplazoDatos y ActualizarVariable) reciben por parámetro el mismo objeto de la clase mencionadas anteriormente, de esta forma, si ambos deciden acceder a cualquiera de los métodos de modificación de la misma estructura, al estar manipulando el mismo objeto, alguno de los 2 threads quedará bloqueado por el método synchronized.

El thread ReemplazoDatos se encarga de usar el algoritmo “páginas no usadas recientemente”, que en el código es la clase Nru, para manejar todo el tema de control de las estructuras. Primeramente, la clase ReemplazoDatos se encarga de ver si la página solicitada está o no en la RAM, si sí entonces no hay problema, de lo contrario, empieza a usar el algoritmo Nru para saber si en la RAM hay espacio disponible y de ser así, tener la posición para asignar la página. Si no hay espacio, Nru se encarga de buscar la página más débil con respecto a las clases del algoritmo, estas clases están representadas por los valores de la 4 columna y son así:

|  |  |
| --- | --- |
| Clase | Valor en matriz |
| Clase 0 | 0 |
| Clase 1 | 1 |
| Clase 2 | 10 |
| Clase 3 | 11 |

El algoritmo calcula cual es la página con el menor valor para sacar. Con esta información el thread ReemplazoDatos se encarga de actualizar los valores de la tabla de páginas y de la memoria física. Esto se ejecuta cada milisegundo

El otro thread, ActualizarVariable, se encarga de cada 20 milisegundos, actualizar las variables de referencia para toda la matriz y ponerlas todas en 0 otra vez.

## **Datos recompilados**

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | Marcos de página | Páginas del proceso | # De fallos |
| 1 | 8 | 8 | 8 |
| 2 | 8 | 16 | 35 |
| 3 | 8 | 32 | 96 |
| 4 | 8 | 64 | 289 |
| 5 | 8 | 128 | 390 |
| 6 | 16 | 32 | 255 |
| 7 | 16 | 64 | 362 |
| 8 | 16 | 128 | 437 |
| 9 | 32 | 64 | 261 |

El número de fallos de páginas para un archivo no siempre es el mismo, pese a que, si se encuentra en un rango, no da un número exacto de fallos cada vez que se corre el programa. Analizando este problema y comparando los resultados de diferentes veces que se ejecutó el programa para un mismo archivo, nos dimos cuenta de que esto ocurría debido a que el algoritmo nru, no siempre decidía sacar la misma página, lo cual, a futuro en la ejecución del programa, podía suceder que apareciera nuevamente la página que sacamos de la RAM, generando un fallo de página, mientras que si el algoritmo hubiera decidido sacar otra página a la que no se le referencie en un futuro, este fallo nunca hubiera ocurrido.

Diagrama

Descripción generada automáticamentePero, ahí nos preguntamos, porque ocurría que el algoritmo sacara distintas páginas en algunas ocasiones, bueno, esto ocurre debido a que, en la ejecución del programa, cada 20 milisegundos ocurren una actualización de la variable de referencia. El problema es que esa actualización no siempre ocurre en el mismo punto del programa para cada vez que es ejecutado, acá un ejemplo.

En la foto anterior podemos ver que en la lectura del archivo 373, tiene en las primeras filas, la columna 2 llena de 1, mientras que en la siguiente iteración esa misma columna se ha actualizado a 0.

Un ejemplo donde para este punto del programa esta actualización no ha ocurrido:

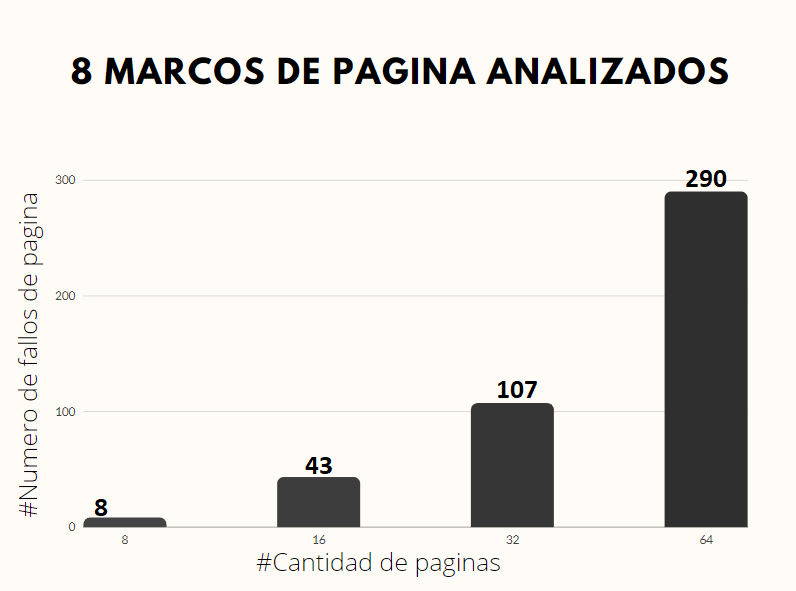
Diagrama, Esquemático

Descripción generada automáticamente con confianza media

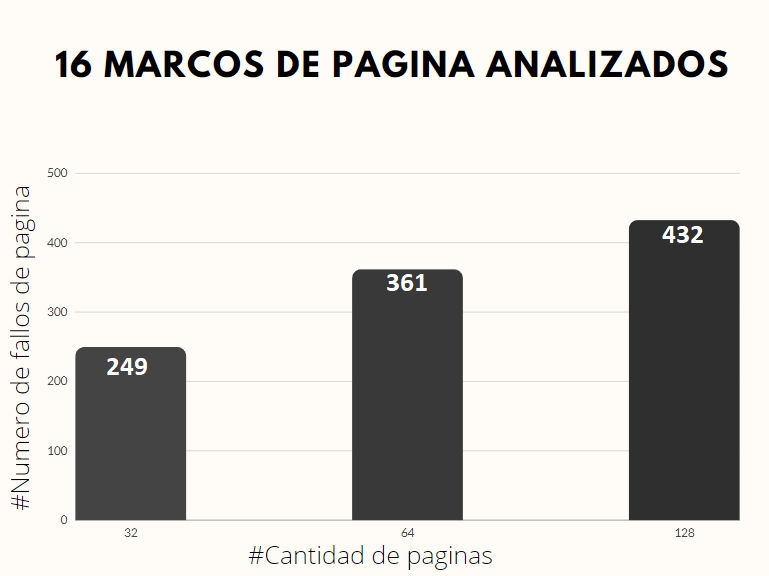
Debido a esto, en la última columna los valores con el peso se ven afectados y cuando el algoritmo entre a seleccionar una página a remover, la decisión será diferente en ambas ejecuciones del programa.

## **Graficas**

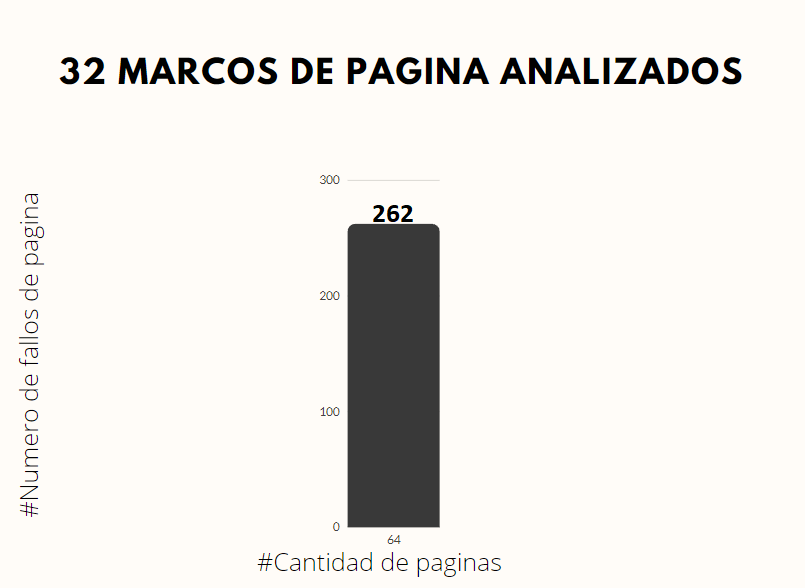
### 8 Marcos de página.



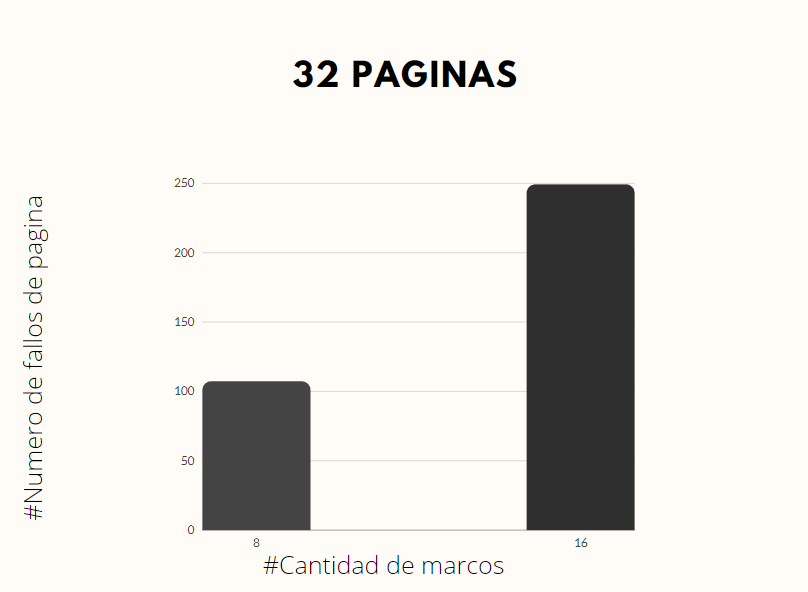
* 16 Marcos de página.



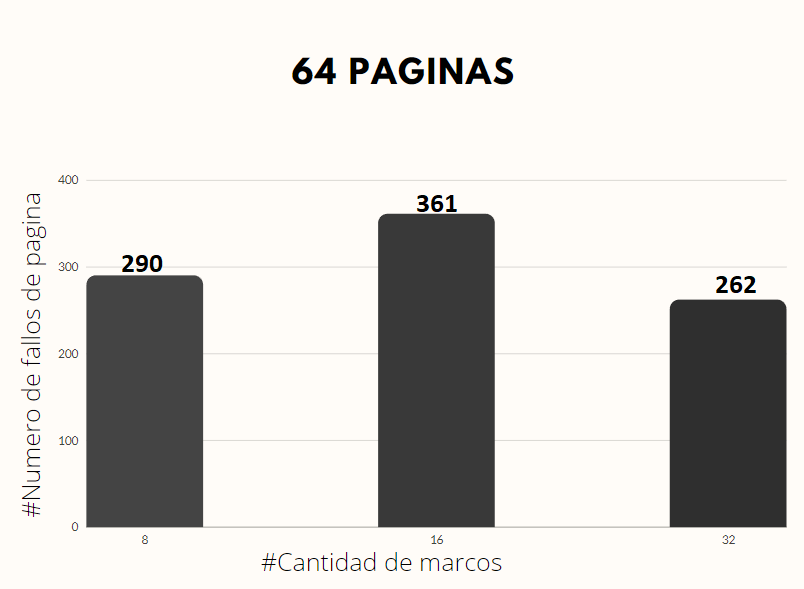
* 32 Marcos de página.



* Análisis de 32 páginas con 8 y 16 marcos.



* Análisis de 64 páginas con 8, 16 y 32 marcos.



Al revisar las gráficas y los análisis hecho por tamaño del programa, como también los resultados por marcos de página con sus respectivos valores de fallas, podemos concluir que, al utilizar un tamaño de programa de 32 páginas, la cantidad de fallos de página es considerablemente menor al usar 8 paginas en vez de 16, en 8 paginas se presentan 107 fallos, mientras que en 16 se presentan 249. En el caso de 64 páginas, quien presenta menor cantidad de fallos son los 32 marcos de página, con 262 fallos, frente a 361 al usar 16 marcos y 290 al usar 8.

Con esto resultado nos podemos dar cuenta que debe haber una relación importante entre el número de marcos de página y entre las páginas, gracias a las graficas nos dimos cuenta de que la proporción entre el número de marcos de página y las páginas asignadas no debe ser muy grande ya que aumenta el número de fallas. Algo importante es denotar que, en un ambiente real, generalmente las referencia a las páginas van juntas en ciertas partes del código, de ahí viene la razón de usar el algoritmo, pero durante estas pruebas, que las páginas fueron generadas por un script en Python mediante la elección de números aleatorios, no se cumple ese principio de agrupación de las páginas, por lo que los resultados obtenidos no son del todo preciosos. Sin embargo, todos estos análisis y a generar un código que haga una alusión a la realidad ayudaron enormemente a la comprensión de los temas vistos en clase.