INFORME DEL PROYECTO FINAL

GESTION DE MEMORIA COMBINADA “SEGMENTACIÓN PAGINADA”

Julián David Rojas Callejas

20182020153

Iván Darío López Díaz

20182020141

Universidad Distrital Francisco José de Caldas

Facultad de ingeniería, Ingeniería de sistemas

I.S., M.Sc. Oswaldo Alberto Romero Villalobos

Bogotá DC, Colombia

06 de junio de 2023

INTRODUCCION

Actualmente la gestión de memoria es un aspecto crucial en los sistemas operativos pues permite manejar y optimizar los recursos y asimismo mejorar el rendimiento de las aplicaciones. Existen diferentes maneras de gestionar la memoria, sin embargo, hay unos más eficientes que otros. Dos de los más importantes son la segmentación y la paginación y estos a su vez fueron combinados para desarrollar lo que hoy conocemos como **segmentación paginada.** la segmentación paginada es un enfoque mucho más avanzado y sofisticado de la memoria en los sistemas operativos modernos porque aprovechan las ventajas de la segmentación y de la paginación para proporcionar un equilibrio óptimo entre la flexibilidad y la eficiencia en el uso de los recursos. para implementarse se necesita cuidado y experiencia porque de otra manera no se puede garantizar una gestión de memoria eficaz.

OBJETIVO GENERAL

- Usando JS en el navegador, desarrollar un aplicativo que muestre como se gestiona un sistema de 16 MiB de RAM (24 bits) usando el modelo de gestión de memoria combinada SEGMENTACIÓN PAGINADA, donde los primeros 12 bits de la dirección virtual corresponden al número de segmento y los restantes para el desplazamiento.

OBJETIVOS ESPECÍFICOS

- Implementar un algoritmo de gestión de memoria combinada segmentación paginada en JavaScript, considerando los diferentes items de memoria en segmentos y páginas, así como tablas de segmentos y páginas.
- Diseñar una interfaz gráfica utilizando HTML Y CSS para visualizar la simulación de la administración de memoria.
- Desarrollar las diferentes funcionalidades del simulador.
- Analizar y comparar los resultados obtenidos.

JUSTIFICACIÓN

Ese proyecto tiene un aprendizaje práctico porque permite la simulación de la gestión de memoria combinada de una manera Diferente a que si se tratara solo de un tema teórico. Esto permite comprender mejor cómo funcionan los algoritmos y las estructuras de datos involucradas en la gestión de memoria. También es importante la realización de éste porque permite a nosotros los estudiantes experimentar con diferentes parámetros y variando los lenguajes de programación que usualmente usamos.

Por otra parte, la visualización y el análisis de los resultados facilita la comprensión de los procesos que hay tras los diferentes sistemas operativos, puede que existan muchísimos errores en el transcurso del proyecto, pero de esta manera es como se irán creando las bases.

MATERIALES

1. Computadora con requisitos mínimos
2. Editor de texto o entorno de desarrollo, en nuestro caso Visual Studio Code con sus respectivas extensiones para JavaScript, HTML y CSS.
3. Bibliotecas de javascript

MÉTODOS

1. Búsqueda de la información acerca de la segmentación paginada
2. Diseño de un bosquejo en Excel
3. Desarrollo de los mockups que se van a utilizar
4. Implementación del algoritmo de gestión de memoria
5. Desarrollo de la interfaz gráfica
6. Configuración de los parámetros
7. Pruebas y ajustes

MARCO TEÓRICO

La gestión de memoria es muy importante porque se encarga de asignar y administrar de manera eficiente los recursos que tenemos nosotros disponibles en un sistema informático. El enfoque que estamos usando en nuestro laboratorio es el de combinación de memoria de segmentación y paginación o más conocido como segmentación paginada.

La segmentación es un método que divide la memoria en diferentes segmentos lógicos y cada uno está asociado a una parte específica del programa. a segmento está representando una unidad lógica que en otras palabras serían el código, los datos o la pila de un proceso. Esas divisiones proporcionan flexibilidad y permiten que los segmentos se puedan compartir en diferentes procesos y eso reduce la duplicación de memoria.

Por otro lado, la paginación divide la memoria en bloques de tamaño fijo y a esas las denominamos páginas y son básicamente unidades de asignación que se utilizan para dividir tanto a los segmentos como a la memoria física en trozos más pequeños y manejables. a la página se le asigna un marco de página en la memoria física lo que permite una gestión eficiente de los recursos. En la segmentación paginada cada segmento se divide en páginas del mismo tamaño y ofrece ventajas como que se mantiene la flexibilidad de la segmentación y a su vez permite que se maneje con mayor eficiencia la memoria física gracias a la paginación, ya que las páginas se pueden asignar y desasignar de forma independiente. El sistema operativo debe mantener una tabla con los segmentos y la información necesaria para traducir de direcciones de lógicas a físicas. esta tabla incluye la dirección base y el límite de cada segmento, así como las páginas asociadas a cada segmento las tablas de páginas a su vez contienen la información de traducción necesaria para cada página como la dirección base del marco de página.

RESULTADOS

Después de investigar en muchas fuentes, logramos implementar lo que para nosotros es la segmentación paginada, tratamos de mantenernos al margen de las indicaciones que nos damos los libros y las clases y el resultado fue el siguiente:

1. Se implementó una interfaz donde inicialmente estamos mostrando una ventanilla para ingresar los diferentes datos de los procesos, seguido de una tabla que después de oprimir el botón agregar va adicionándoles procesos a ella misma y adicionalmente a la mano derecha encontramos la tabla de la cantidad de páginas necesarias para representar.

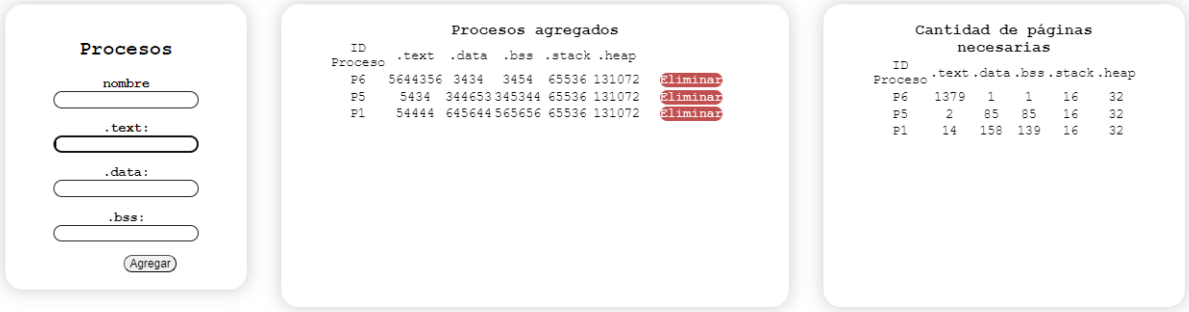
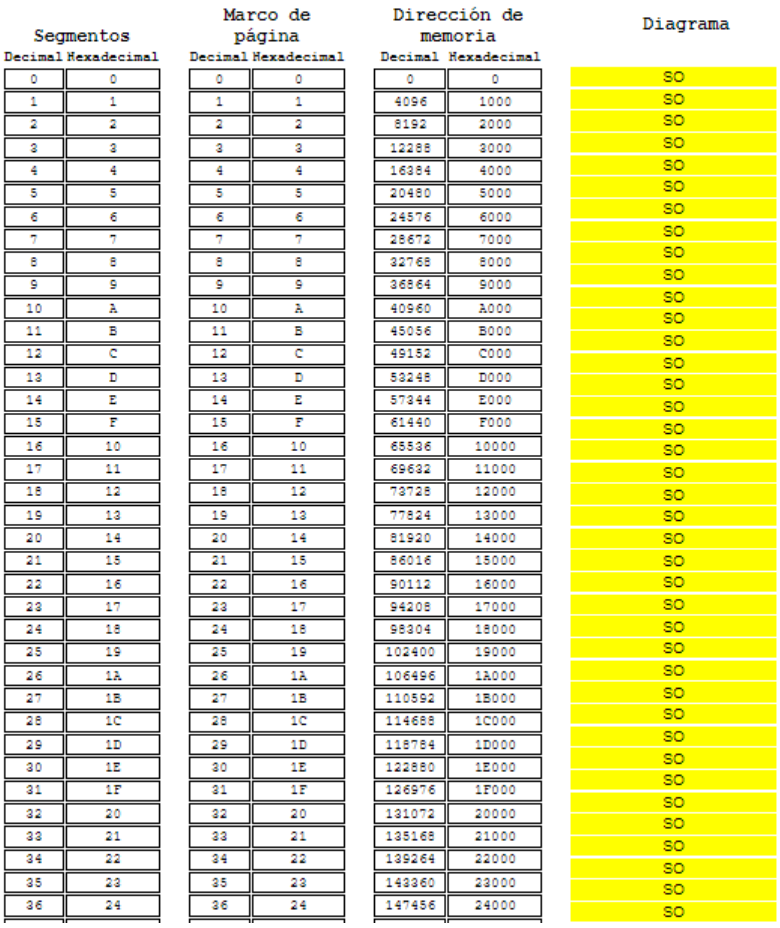


Figura 1. Interfaz gráfica desarrollada, se han agregado 3 procesos

2. Luego se presentan las diferentes indicaciones para acceder a las direcciones de memoria, en esta imagen podemos ver Cada uno de los segmentos los Marcos de página y las direcciones de memoria física. Al lado derecho, en color amarillo se puede observar la subdivisión del sistema operativo que en nuestro caso fue de 1 MiB.



Segmentos		Marco de página		Dirección de memoria		Diagrama
Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	
0	0	0	0	0	0	SO
1	1	1	1	4096	1000	SO
2	2	2	2	8192	2000	SO
3	3	3	3	12288	3000	SO
4	4	4	4	16384	4000	SO
5	5	5	5	20480	5000	SO
6	6	6	6	24576	6000	SO
7	7	7	7	28672	7000	SO
8	8	8	8	32768	8000	SO
9	9	9	9	36864	9000	SO
10	A	10	A	40960	A000	SO
11	B	11	B	45056	B000	SO
12	C	12	C	49152	C000	SO
13	D	13	D	53248	D000	SO
14	E	14	E	57344	E000	SO
15	F	15	F	61440	F000	SO
16	10	16	10	65536	10000	SO
17	11	17	11	69632	11000	SO
18	12	18	12	73728	12000	SO
19	13	19	13	77824	13000	SO
20	14	20	14	81920	14000	SO
21	15	21	15	86016	15000	SO
22	16	22	16	90112	16000	SO
23	17	23	17	94208	17000	SO
24	18	24	18	98304	18000	SO
25	19	25	19	102400	19000	SO
26	1A	26	1A	106496	1A000	SO
27	1B	27	1B	110592	1B000	SO
28	1C	28	1C	114688	1C000	SO
29	1D	29	1D	118784	1D000	SO
30	1E	30	1E	122880	1E000	SO
31	1F	31	1F	126976	1F000	SO
32	20	32	20	131072	20000	SO
33	21	33	21	135168	21000	SO
34	22	34	22	139264	22000	SO
35	23	35	23	143360	23000	SO
36	24	36	24	147456	24000	SO

Figura 2. Interfaz gráfica desarrollada, se evidencian los segmentos, marcos de página y dirección de memoria

3. Luego, podemos observar en color verde cuando empezamos a pintar los diferentes procesos en el orden de que se van asignando.

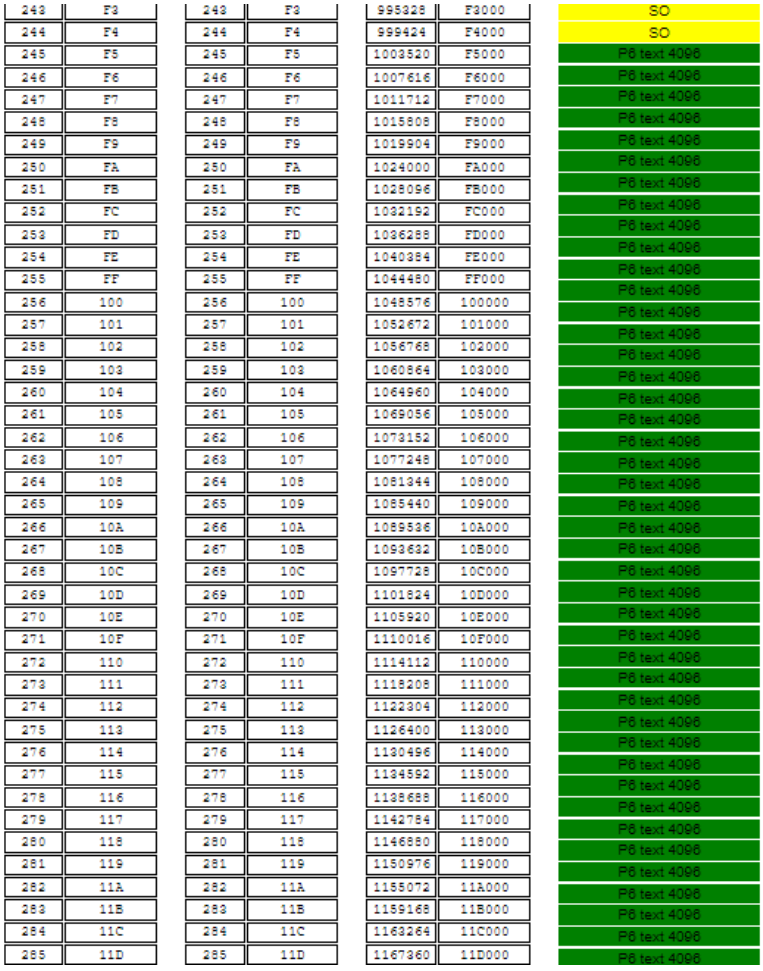


Figura 3. Interfaz gráfica desarrollada, se han empezado a pintar los procesos

4. También es posible ver un resumen de los procesos que hemos agregado de forma independiente mediante un combo box.

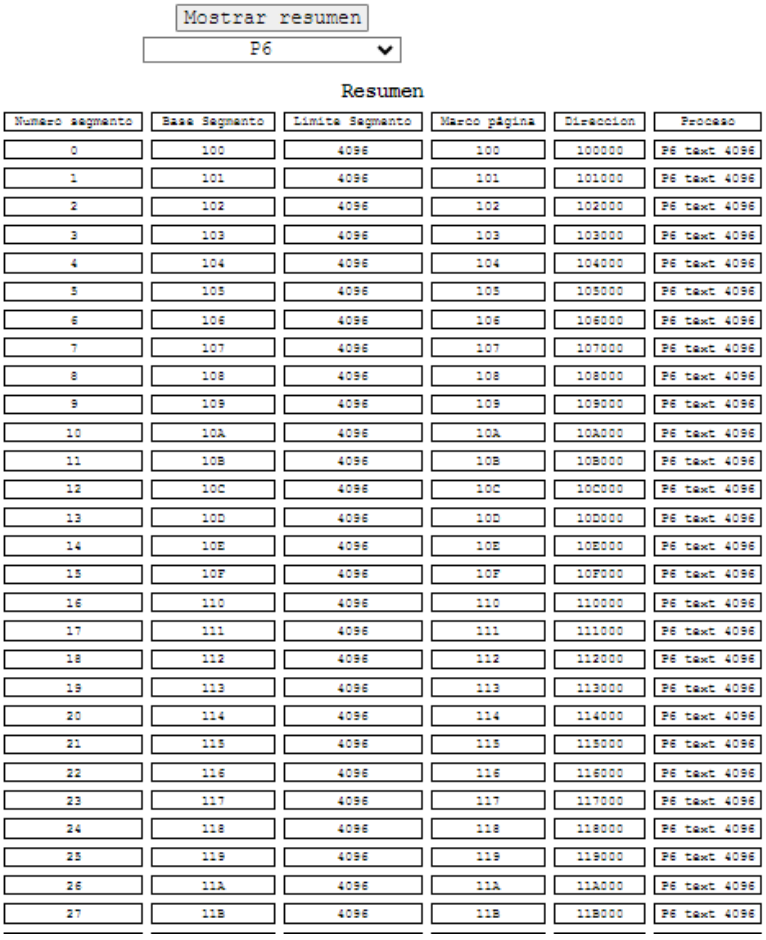


Figura 4. Interfaz gráfica desarrollada, resumen de los procesos

5. También es posible eliminar los procesos, esto ocasiona que quede en pedazos sin llenar, por eso, cuando creemos el siguiente se van a llenar los espacios que están vacíos, si llegado el caso no alcanzan y aparece el otro proceso, pasaremos a la última posición y lo seguiremos agregando.

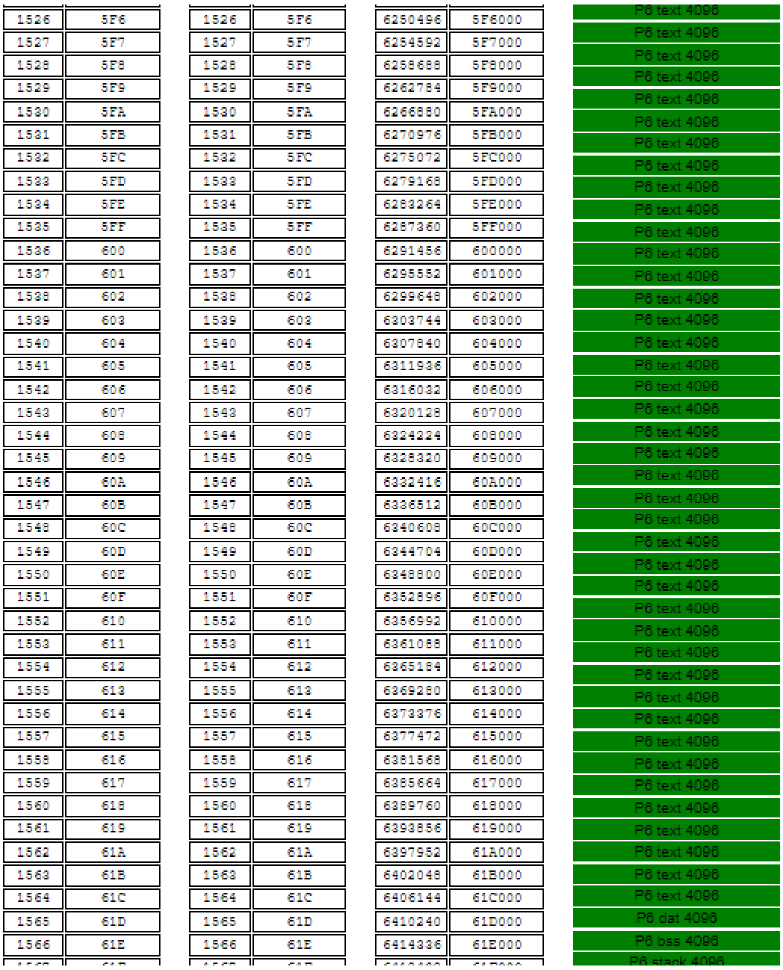


Figura 5. Interfaz gráfica desarrollada, eliminación e inserción procesos

CONCLUSIONES

La gestión de memoria combinada segmentación paginada es una de las más eficientes para manejar los recursos que tenemos disponibles en los sistemas operativos, es un método que recopila las ventajas de la segmentación y la paginación y nos permite más flexibilidad y mejor administración.

La segmentación facilita la traducción de direcciones virtuales a direcciones físicas mediante las tablas de segmentos y las tablas de páginas. Y a pesar de las muchas ventajas que tiene, es importante tener en cuenta que la gestión de memoria de segmentación paginada puede traer

cierta complejidad para implementarse y en y en la traducción de las direcciones, se tiene que traducir correctamente para garantizar el funcionamiento correcto del sistema.

Es muy importante empezar a comprender y aplicar los conceptos aprendidos en la materia porque si nos damos cuenta, actualmente nuestra sociedad es dependiente de cualquier sistema operativo.

A pesar de que la experiencia en programación ya es suficiente, sí costó un poco más adaptarnos a la parte web y empezar a desarrollar el gestor de memoria. es verdad que javascript es un lenguaje de programación muchísimo más flexible, sin embargo, para la corrección de errores es un poco más complejo pues no se sabe exactamente dónde se ha situado.

REFERENCIAS

Stallings, W. (2018). Operating Systems: Internals and Design Principles. Pearson.

Tanenbaum, A. S., & Bos, H. (2014). Modern Operating Systems. Pearson.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts. Wiley.

Abraham, S. S. (2015). Operating Systems. PHI Learning Pvt. Ltd.