Pablo Ortega – 202021700

Miguel Santiago Castillo - 201633992

Ponto Andrés Moreno – 202224525

Caso de Estudio 2
Memoria Virtual

Descripción del algoritmo usado para generar las referencias de página (modo uno)

El algoritmo usado para generar las referencias de página simula las operaciones de acceso a la memoria durante el proceso de recuperación de un mensaje oculto en una imagen BMP utilizando memoria virtual. El algoritmo sigue los siguientes pasos:

1. Entrada de parámetros

- El usuario ingresa dos parámetros:
    - Tamaño de página (TP): El tamaño en bytes que tendrá cada página de la memoria virtual.
    - Ruta del archivo de imagen: La ubicación de la imagen BMP que se utilizará para ocultar o recuperar el mensaje.

2. Lectura de la imagen BMP

- Se crea un objeto de la clase Imagen que:
    - Lee los píxeles de la imagen en formato BMP de 24 bits.
    - Almacena los datos de la imagen en una matriz que contiene la información RGB de cada pixel.
    - Extrae el alto y ancho de la imagen a partir de los metadatos del archivo BMP.

3. Generación de las referencias de página

El algoritmo recorre la imagen pixel por pixel y genera referencias de página correspondientes a las operaciones de lectura necesarias para recuperar la imagen y el mensaje oculto. Los pasos son:

- Row-major order: La imagen es almacenada en orden por filas, lo que significa que el algoritmo recorre las filas una a una. Cada fila de la imagen contiene píxeles con tres componentes (rojo, verde, azul) que deben ser leídos.

- Referencia de la imagen: Por cada byte que se lee de la imagen, se genera una referencia de página que incluye:
    - La página virtual en la que se encuentra el byte (calculada según el tamaño de página).
    - El desplazamiento dentro de la página (indica en qué posición dentro de la página está el byte).
    - El tipo de operación, que en este caso será una lectura (R).
- Referencia del mensaje: Además de las referencias de la imagen, también se generan referencias para el vector de bytes que contiene el mensaje oculto. Cada vez que se accede a un byte del mensaje para recuperarlo, se genera una referencia de escritura (W), indicando que ese byte está siendo escrito en el mensaje resultante.

4. El algoritmo también calcula el numero de filas (NF), el numero de columnas (NC), el numero de referencias (NR) y el numero de paginas virtuales (NP)

5. Escritura en archivo referencias.txt

- Toda la información generada, como las dimensiones de la imagen, el número de referencias, y las referencias mismas, se escribe en un archivo de salida (referencias.txt).

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

A continuación se delinean las clases que se utilizan para simular el sistema de paginación, su proposito y como se actualizan :

1. Clase Ram.
    - Propósito: Representa la memoria física (RAM) del sistema. Contiene un conjunto de páginas reales (de la clase PaginaReal) y proporciona métodos para interactuar con ellas.
    - Actualización:
        - Las páginas reales en la RAM se referencian y actualizan cada vez que se accede a una página virtual. Si la página ya está presente en la RAM (indicador de presencia), se marca como referenciada.
        - Si la RAM está llena, el sistema intentará reemplazar una página real según las políticas de manejo de la memoria (NRU, Not Recently Used). El hilo actualizarR se encarga de poner a 0 los bits de referencia de las páginas reales para que puedan ser elegidas para reemplazo

2. Clase MemoriaVirtual
    - Propósito: Representa la memoria virtual completa que incluye todas las páginas posibles.
    - Actualización:
        - Cuando se realiza una solicitud de página, se verifica si la página está en memoria real. Si no está presente, se buscará en la memoria virtual y se traerá a la RAM.
3. Clase PAginaReal
    - Propósito: Representa una página física en la RAM. Cada página tiene varios atributos como el número de página, el bit de referencia, etc.
    - Actualización:
        - El bit de referencia se actualiza cada vez que la página es accedida. Si el bit es 0, se marca como candidata para ser reemplazada. El hilo actualizarR revisa periódicamente las páginas y ajusta el valor de referencia.
        - Si una página virtual es traída a la RAM, se reemplaza una página real que tenga su bit de referencia en 0.
4. Clase PaginaVirtual
    - Propósito: Representa una página virtual en la tabla de páginas. Tiene información sobre si la página está presente en RAM (bit de presencia) y otras propiedades necesarias para la gestión de la memoria.
    - Actualización:
        - Si la página está en RAM, el sistema incrementa el contador de "hits" (accesos exitosos). Si no, se incrementan los "misses" y se reemplaza una página de la RAM con la nueva página traída de la memoria virtual.
5. Clase TablaDePaginas
    - Propósito: Gestiona la relación entre las páginas virtuales y las páginas reales. Contiene métodos para recuperar páginas, buscar páginas en la RAM, y reemplazarlas cuando es necesario.
    - Actualización:

- Cada vez que una página virtual es accedida, se actualiza su presencia en la tabla de páginas. Si una página no está presente, se debe traer de la memoria virtual y asignarla a una página real en la RAM.
- El proceso de reemplazo se realiza cuando la RAM está llena, y se selecciona una página candidata usando el bit de referencia (gestión a través de la clase actualizarR).

6. ActualizarR
   - Propósito: Este hilo se encarga de manejar la actualización periódica de las páginas en la RAM, específicamente sus bits de referencia, siguiendo la política de reemplazo NRU (Not Recently Used).
   - Actualización:
     - Cada 2 milisegundos, el hilo recorre las páginas reales y ajusta su bit de referencia. Si el bit de referencia está en 0, se marca la página como candidata para ser removida en caso de que sea necesario realizar un reemplazo.
7. ManejadorMemoria
   - Propósito: Es el controlador principal que coordina el acceso a la memoria RAM y la memoria virtual. Se encarga de gestionar los "hits" y "misses", y de asegurar que las páginas sean reemplazadas adecuadamente cuando sea necesario.
   - Actualización:
     - Si una página no está en la RAM, el sistema busca una página candidata para ser reemplazada y la reemplaza por la página virtual que se necesita cargar.

Esquema de sincronización usado. Justifique brevemente dónde es necesario usar sincronización y por qué.

La sincronización es necesaria para evitar condiciones de carrera y garantizar la consistencia de los datos cuando múltiples hilos intentan acceder o modificar las estructuras de datos compartidas, como la RAM y la tabla de páginas.

1. Uso de bloques sincronizados
   - En varias secciones del código, como en la clase ManejadorMemoria y en el hilo actualizarR, se utiliza la palabra clave synchronized para asegurarse de que los accesos a las estructuras de datos compartidas estén protegidos.
   - Ejemplo: En el método informacion de la clase ManejadorMemoria, se usa un bloque sincronizado sobre la tabla de páginas:

```
public void informacion(int pagina, int referencia)  {

    synchronized(tablaDePaginas){

    PaginaVirtual paginaVirtual = tablaDePaginas.RecuperarPaginaV(pagina, memoriaVirtual);
    if (paginaVirtual.getPresencia() == 1){
        tablaDePaginas.buscarPagina(paginaVirtual.getNumereDePagina(), ram, paginaVirtual);

        tiempor += 0.000025;
        hits++;
```

Esto es necesario porque varias operaciones podrían estar accediendo a la tabla de páginas simultáneamente, y sin la sincronización, se podrían producir condiciones de carrera que afecten la coherencia de la tabla, especialmente cuando se realiza la operación de reemplazo de páginas.

La tabla de páginas es un recurso compartido que podría ser accedido por múltiples hilos a la vez. Si un hilo está recuperando o reemplazando una página, otro hilo no debería interferir hasta que la operación haya terminado, ya que esto podría resultar en datos corruptos o resultados inconsistentes.

- Todos los métodos críticos en la clase `Ram` están sincronizados para garantizar la integridad de las operaciones que modifican la memoria física. Esto incluye la adición y eliminación de páginas reales (`agregarPaginaReal`, `quitarPaginaReal`), la consulta de una página específica (`conseguirPaginaReal`), y la verificación de si la RAM está llena (`estaLlena`). Además, se sincronizan los accesos a las estructuras de datos internas como las listas de páginas reales y posiciones libres (`getPaginasReales`, `setPaginasReales`, `getSacar`, `setSacar`). Esta sincronización es necesaria para evitar condiciones de carrera cuando múltiples hilos acceden y modifican la memoria compartida simultáneamente.

```java
synchronized public void agregarPaginaReal(PaginaReal paginaReal) {

    paginas ++;

    paginasReales.set (pociconesLibres.get(index:0), paginaReal) ;
    paginaReal.setNumero(pociconesLibres.get(index:0));
    paginaReal.setPresencia(presencia:1);
    paginaReal.setReferencia(referencia:1);
    pociconesLibres.remove(index:0);
}

synchronized public void quitarPaginaReal(PaginaReal paginaReal) {
    paginas --;
    pociconesLibres.add(paginaReal.getNumero());
    paginasReales.set(paginaReal.getNumero(), element:null);
    paginaReal.setNumero(-1);
    paginaReal.setPresencia(presencia:0);
}

synchronized public PaginaReal conseguirPaginaReal(int numero) {
    return paginasReales.get(numero);
}




public int getTamaño() { return marcos; }
public void setTamaño(int tamaño) { this.marcos = tamaño; }

public int getTamañoPaginas() { return tamañoPaginas; }
public void setTamañoPaginas(int tamañoPaginas) { this.tamañoPaginas = tamañoPaginas; }

public synchronized ArrayList<PaginaReal> getPaginasReales() {
    return paginasReales;
}

public synchronized void setPaginasReales(ArrayList<PaginaReal> paginasReales) {
    this.paginasReales = paginasReales;
}

public synchronized ArrayList<Integer> getSacar() {
    return sacar;
}

public synchronized void setSacar(ArrayList<Integer> prioridad) {
    this.sacar = prioridad;
}

public synchronized boolean estaLlena() {
    return (paginas >= marcos);
}
```

2. Uso de wait() y notify()
    - En el método informacion de la clase ManejadorMemoria, también se emplean los métodos wait() y notify() para gestionar situaciones donde la RAM esté llena. Si no hay espacio disponible en la RAM para cargar una nueva página, el hilo entra en espera hasta que otra parte del sistema libere espacio.

```
while (x){

            if (!ram.estaLlena()){
                PaginaReal paginaReal = new PaginaReal(modificacion:0, referencia:0, presencia:0, ram.getTamañoPaginas() );
                tablaDePaginas.reemplazarPagina(paginaReal, paginaVirtual, ram);
                x = false;
            }
            else {

                try {
                    tablaDePaginas.wait();
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
```

Cuando una página es reemplazada o eliminada, el método notify() se llama para despertar a los hilos en espera, permitiéndoles continuar con su ejecución:

```
synchronized public void  quitarPagina(Integer vpagianReal, Ram ram, MemoriaVirtual memoriaVirtual) {
    synchronized (ram) {
    PaginaReal paginaReal = ram.conseguirPaginaReal(vpagianReal);
    if (paginaReal == null){
        int x = 1;
    }
    memoriaVirtual.RecuperarPagina(listaDeRelacion[paginaReal.getNumero()]).setPresencia(presencia:0);
    listaDeRelacion[paginaReal.getNumero()] = -1;

    ram.quitarPaginaReal(paginaReal);
    this.notify();}


}
```

Este mecanismo es necesario para asegurar que cuando la RAM está llena, los hilos no sigan intentando reemplazar páginas innecesariamente, lo que podría generar una sobrecarga en el sistema. En cambio, los hilos esperan hasta que se libere espacio, lo que optimiza el uso de la memoria y evita condiciones de carrera.

3. Sincronización del hilo actualizarR
    - El hilo actualizarR, encargado de gestionar los bits de referencia de las páginas reales en la RAM, también usa bloques sincronizados para evitar que otros hilos interfieran mientras se actualizan las páginas reales. La sincronización aquí asegura que mientras se recorren y actualizan las páginas reales, ningún otro hilo pueda modificar la RAM, garantizando la integridad de las operaciones sobre la memoria física.

```java
public class actualizarR extends Thread {

    private Ram ram;
    private boolean termina = true;

    public actualizarR(Ram ram) {
        this.ram = ram;
    }

    public void run() {
        while (termina) {
            try {
                sleep(millis:2);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            ArrayList<Integer> sacar = new ArrayList<Integer>();
            int x = 0;
            int pr = 0 ;
            synchronized (ram) {
                for (PaginaReal paginaReal : ram.getPaginasReales()) {
                    if (paginaReal == null){
                        x++;
                        continue;
                    }
                    pr ++;
                    if (pr == ram.getPaginas()) {
                        break;
                    }
                    if (paginaReal.getReferencia() == 1) {
                        paginaReal.setReferencia(referencia:0);
                    }
                    else if (paginaReal.getReferencia() == 0){
                        sacar.add(index:0, x);
                    }
                    x++;

                }

                ram.setSacar(sacar);


        }
    }
    }

    public void terminar() {
        termina = false;
    }

}
```

Una tabla con los datos recopilados (y porcentaje de hits y misses por cada escenario simulado).

La primera tabla muestra la generación de referencias de las imágenes para imágenes y mensajes de diferentes tamaños. NFIlas y NColumnas dicen el tamaño de las imágenes.

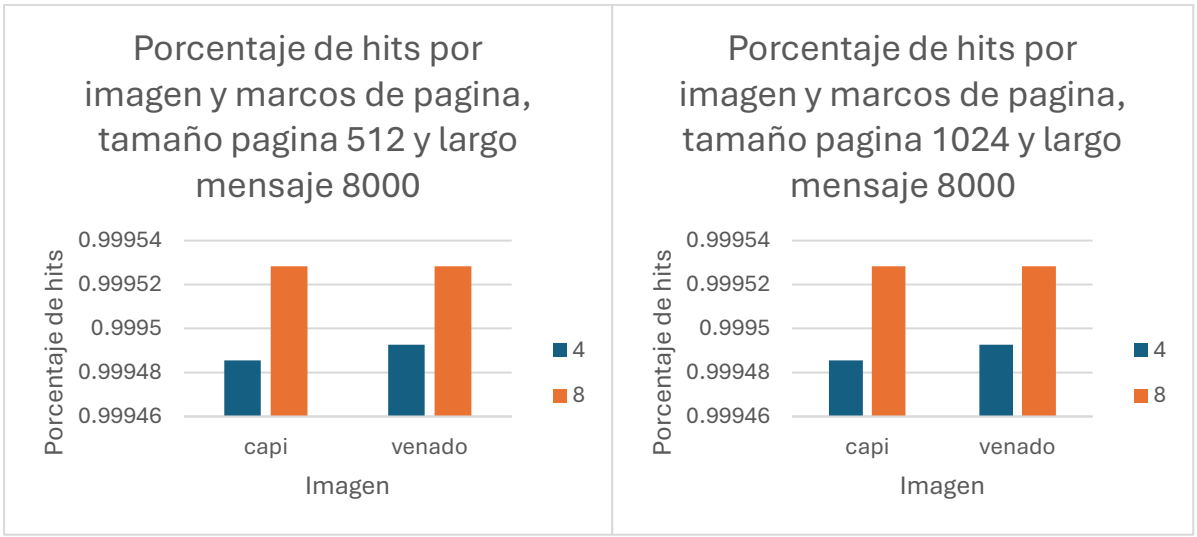
Imagen	Largo mensaje	TamanoPagina	NFilas	NColumnas	NRegistros	NP
caso2-parrots_mod	4820	512	256	384	81956	576
capi	100	512	205	246	1682	295
capi	1000	512	205	246	16999	295
capi	2000	512	205	246	38997	295
capi	4000	512	205	246	70498	295

capi	8000	512	205	246	139943	295
venado	100	512	407	612	1682	1459
venado	1000	512	407	612	16999	1459
venado	2000	512	407	612	38997	1459
venado	4000	512	407	612	70498	1459
venado	8000	512	407	612	139943	1459
caso2-parrots_mod	4820	1024	256	384	81956	288
capi	100	1024	205	246	1682	147
capi	1000	1024	205	246	16999	147
capi	2000	1024	205	246	38997	147
capi	4000	1024	205	246	70498	147
capi	8000	1024	205	246	139943	147
venado	100	1024	407	612	1682	729
venado	1000	1024	407	612	16999	729
venado	2000	1024	407	612	38997	729
venado	4000	1024	407	612	70498	729
venado	8000	1024	407	612	139943	729

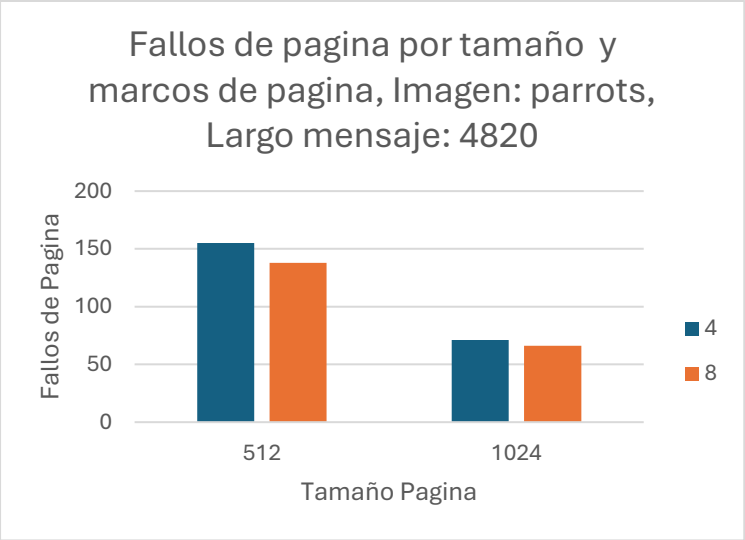
Tamano Pagina	Largo Mensaje	Imagen	Marcos Pagina	Hits	Misses	Tiempo	Total Accesos	Tiempo Ram	Tiempo Swap	Porcentaje Aciertos	Porcentaje Misses
512	4820	caso2-parrots_mod	4	81868	88	882	81956	2	819560	99.893%	0.107%
512	4820	caso2-parrots_mod	8	81876	80	802	81956	2	819560	99.902%	0.098%
512	100	capi	4	1682	0	0	1682	0	16820	100.000%	0.000%
512	100	capi	8	1682	0	0	1682	0	16820	100.000%	0.000%
512	1000	capi	4	16985	14	140	16999	0	169990	99.918%	0.082%
512	1000	capi	8	16989	10	100	16999	0	169990	99.941%	0.059%
512	2000	capi	4	38958	39	390	38997	0	389970	99.900%	0.100%
512	2000	capi	8	38964	33	330	38997	0	389970	99.915%	0.085%
512	4000	capi	4	70424	74	741	70498	1	704980	99.895%	0.105%
512	4000	capi	8	70432	66	661	70498	1	704980	99.906%	0.094%
512	8000	capi	4	139790	153	1533	139943	3	1399430	99.891%	0.109%
512	8000	capi	8	139805	138	1383	139943	3	1399430	99.901%	0.099%
512	100	venado	4	1682	0	0	1682	0	16820	100.000%	0.000%
512	100	venado	8	1682	0	0	1682	0	16820	100.000%	0.000%
512	1000	venado	4	16985	14	140	16999	0	169990	99.918%	0.082%
512	1000	venado	8	16989	10	100	16999	0	169990	99.941%	0.059%
512	2000	venado	4	38957	40	400	38997	0	389970	99.897%	0.103%
512	2000	venado	8	38964	33	330	38997	0	389970	99.915%	0.085%
512	4000	venado	4	70423	75	751	70498	1	704980	99.894%	0.106%
512	4000	venado	8	70432	66	661	70498	1	704980	99.906%	0.094%
512	8000	venado	4	139788	155	1553	139943	3	1399430	99.889%	0.111%

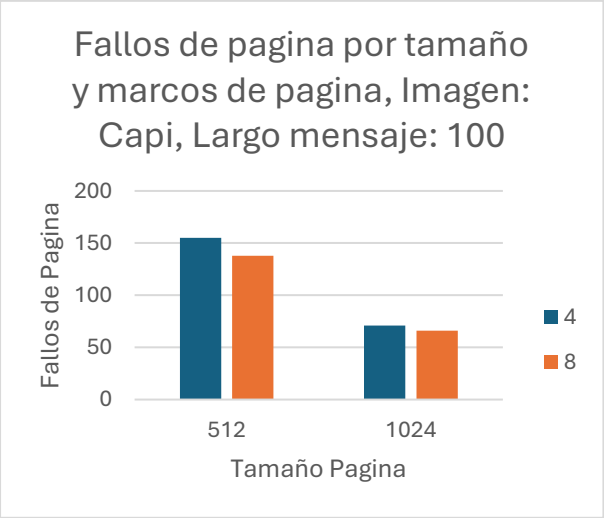
512	8000	venado	8	139805	138	1383	139943	3	1399430	99.901%	0.099%
1024	4820	caso2-parrots_mod	4	81916	40	402	81956	2	819560	99.951%	0.049%
1024	4820	caso2-parrots_mod	8	81921	35	352	81956	2	819560	99.957%	0.043%
1024	100	capi	4	1682	0	0	1682	0	16820	100.000%	0.000%
1024	100	capi	8	1682	0	0	1682	0	16820	100.000%	0.000%
1024	1000	capi	4	16994	5	50	16999	0	169990	99.971%	0.029%
1024	1000	capi	8	16998	1	10	16999	0	169990	99.994%	0.006%
1024	2000	capi	4	38979	18	180	38997	0	389970	99.954%	0.046%
1024	2000	capi	8	38984	13	130	38997	0	389970	99.967%	0.033%
1024	4000	capi	4	70462	36	361	70498	1	704980	99.949%	0.051%
1024	4000	capi	8	70468	30	301	70498	1	704980	99.957%	0.043%
1024	8000	capi	4	139871	72	723	139943	3	1399430	99.949%	0.051%
1024	8000	capi	8	139877	66	663	139943	3	1399430	99.953%	0.047%
1024	100	venado	4	1682	0	0	1682	0	16820	100.000%	0.000%
1024	100	venado	8	1682	0	0	1682	0	16820	100.000%	0.000%
1024	1000	venado	4	16994	5	50	16999	0	169990	99.971%	0.029%
1024	1000	venado	8	16998	1	10	16999	0	169990	99.994%	0.006%
1024	2000	venado	4	38980	17	170	38997	0	389970	99.956%	0.044%
1024	2000	venado	8	38984	13	130	38997	0	389970	99.967%	0.033%
1024	4000	venado	4	70462	36	361	70498	1	704980	99.949%	0.051%
1024	4000	venado	8	70468	30	301	70498	1	704980	99.957%	0.043%
1024	8000	venado	4	139872	71	713	139943	3	1399430	99.949%	0.051%
1024	8000	venado	8	139877	66	663	139943	3	1399430	99.953%	0.047%

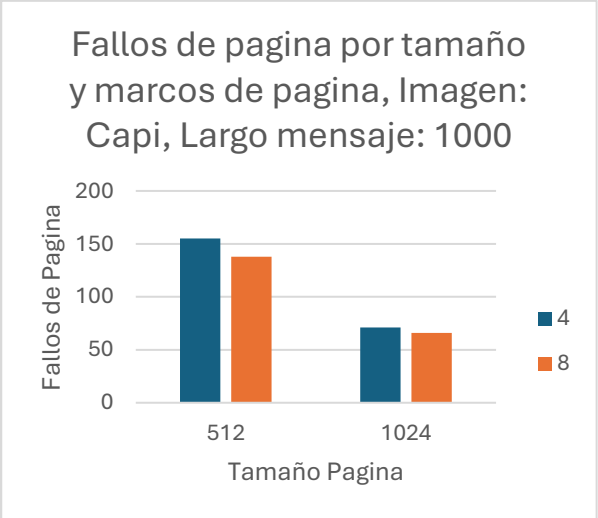
Una serie de gráficas que ilustren el comportamiento del sistema. Para eso muestre gráficas donde fije tamaño de página y grafique Tamaño de imagen vs. Marcos asignados vs. Porcentaje de hits. La gráfica al final del enunciado ilustra el tipo de gráfica que buscamos.

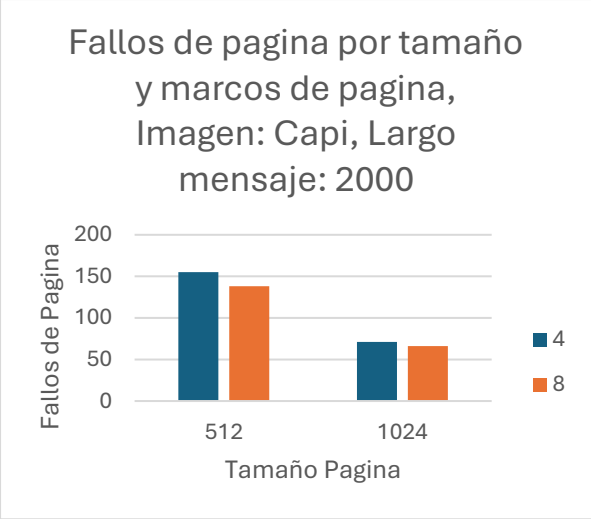


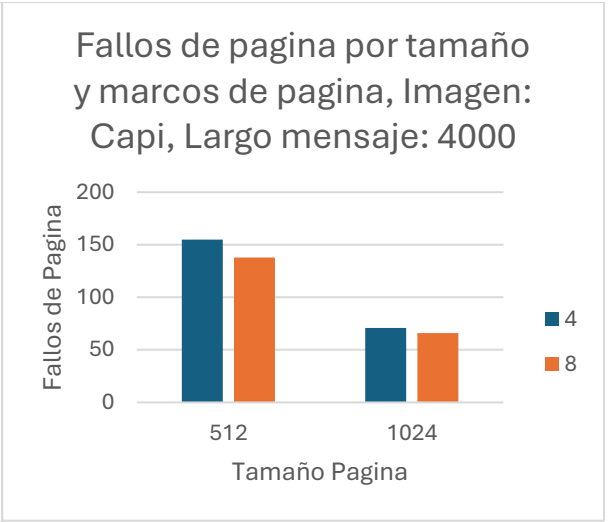
Corra los escenarios y genere gráficas que muestren los datos recopilados para los diferentes escenarios.

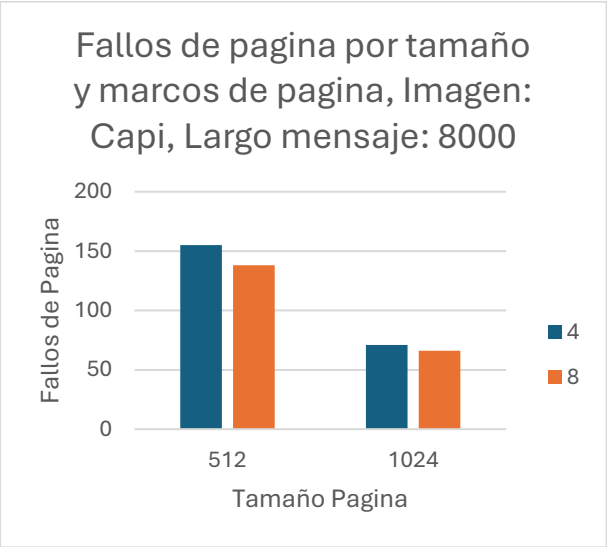


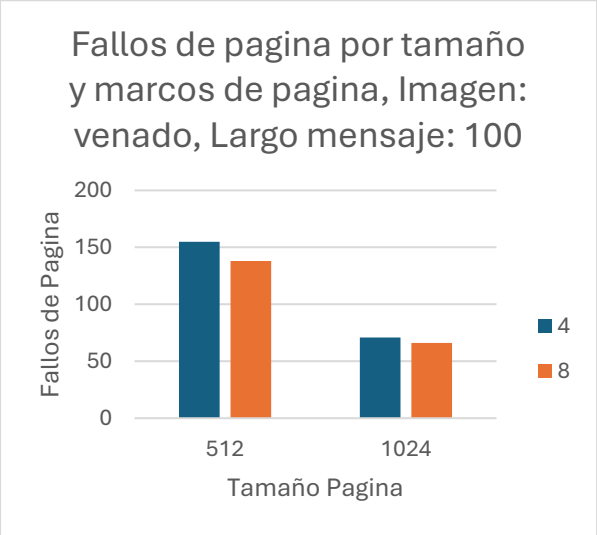


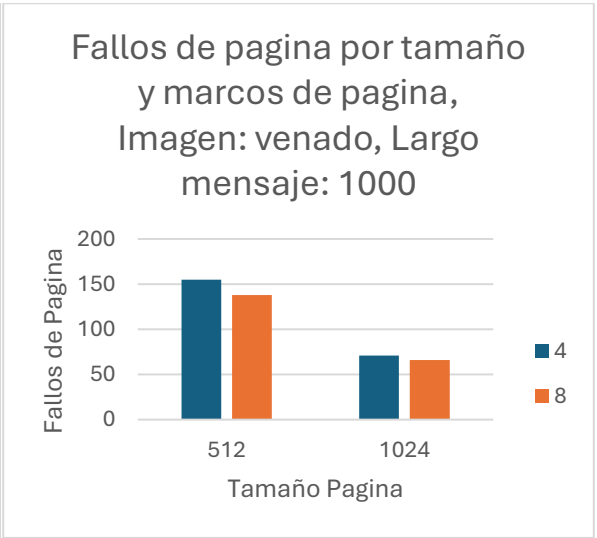


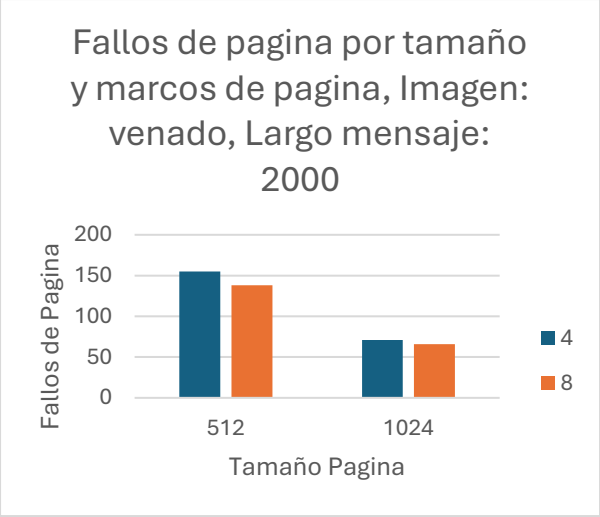

Fallos de pagina por tamaño y marcos de pagina, Imagen: Capi, Largo mensaje: 2000
Fallos de Pagina
200
150
100
50
0
512
1024
Tamaño Pagina
4
8

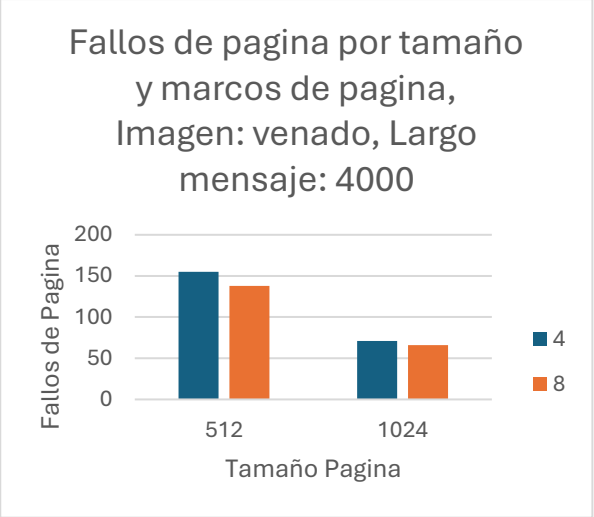

Fallos de pagina por tamaño y marcos de pagina, Imagen: Capi, Largo mensaje: 4000
Fallos de Pagina
200
150
100
50
0
512
1024
Tamaño Pagina
4
8

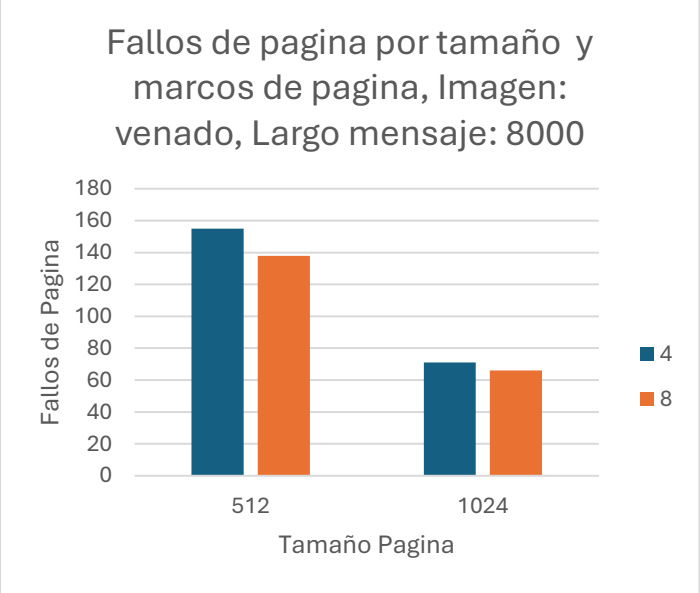

Fallos de pagina por tamaño y marcos de pagina, Imagen: Capi, Largo mensaje: 8000
Fallos de Pagina
200
150
100
50
0
512
1024
Tamaño Pagina
4
8


Fallos de pagina por tamaño y marcos de pagina, Imagen: venado, Largo mensaje: 100
Fallos de Pagina
200
150
100
50
0
512
1024
Tamaño Pagina
4
8


Fallos de pagina por tamaño y marcos de pagina, Imagen: venado, Largo mensaje: 1000
Fallos de Pagina
200
150
100
50
0
512
1024
Tamaño Pagina
4
8

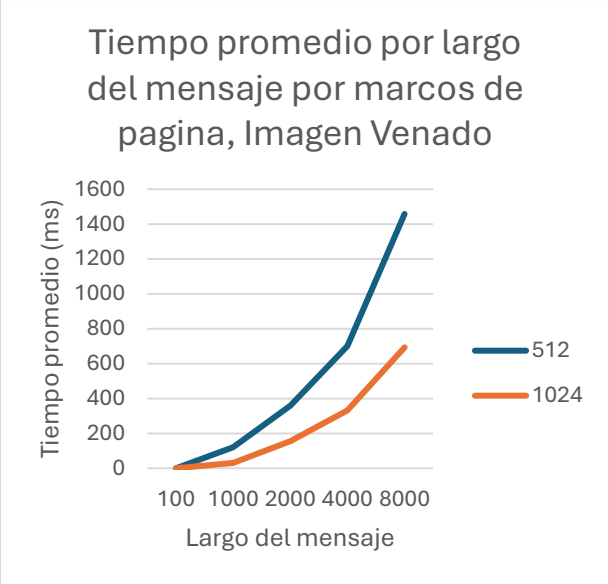


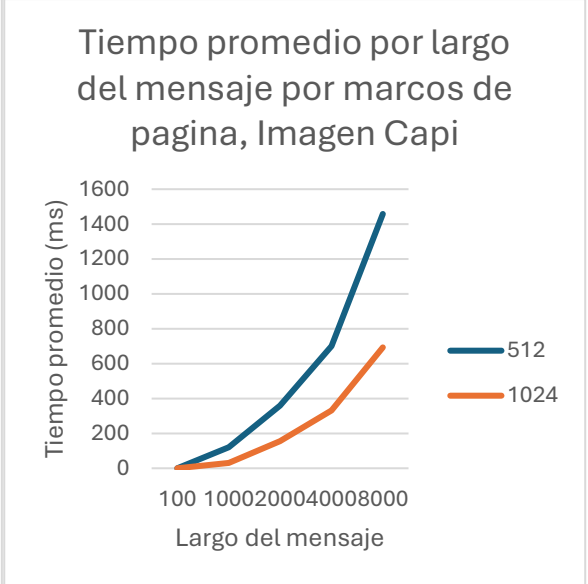


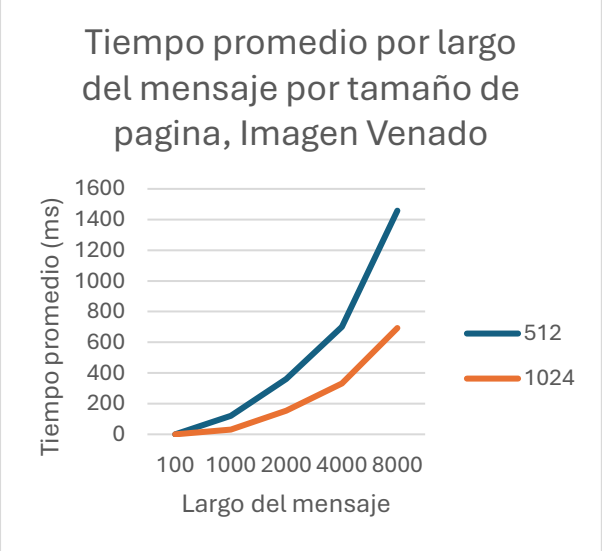


Además de los escenarios definidos, considere otras configuraciones que le permitan entender cómo afecta la memoria virtual el desempeño del programa.

Incluya las gráficas de tiempo (hits, misses, total).







Escriba su interpretación de los resultados: ¿corresponden a los resultados que esperaba, con respecto al número de marcos asignados? Explique su respuesta.

Los resultados obtenidos son consistentes con lo que se esperaba en relación al número de marcos de página asignados. Cuando se incrementa el número de marcos de página (de 4 a 8), se observa una disminución significativa en la cantidad de fallos de página, especialmente para mensajes de mayor tamaño y en imágenes más grandes. Esto se debe a que, al tener más marcos de página disponibles, se puede mantener un mayor número de páginas en la memoria RAM, lo que reduce la necesidad de cargar y descargar páginas constantemente desde la memoria virtual.

Por ejemplo, en las gráficas de **"Fallos de página por tamaño y marcos de página"**, se ve claramente que al utilizar 8 marcos de página en lugar de 4, los fallos de página disminuyen considerablemente tanto para tamaños de página de 512 como de 1024 bytes. Además, los tiempos promedio son menores con 8 marcos, ya que hay menos fallos de página que generan acceso a la memoria swap.

En cuanto a los **porcentajes de hits**, estos también aumentan cuando se asignan más marcos de página. Por ejemplo, para el mensaje de 8000 caracteres y la imagen "venado", el porcentaje de hits es más alto con 8 marcos, tanto para tamaños de página de 512 como de 1024, como se muestra en las gráficas comparativas de hits. Esto confirma que al incrementar el número de marcos, el sistema puede retener más páginas relevantes en RAM, mejorando el rendimiento y reduciendo los tiempos de acceso.

En Conclusión los resultados concuerdan con las expectativas: incrementar los marcos asignados resulta en menos fallos de página, tiempos de acceso menores, y mayores porcentajes de hits, lo que se espera de un sistema de paginación.

¿Si la localidad del problema manejado fuera diferente cómo variarían los resultados? Explique su respuesta. (considere una localidad mayor y una localidad menor).

La localidad de referencia se refiere a la tendencia de los procesos a acceder a un grupo cercano de páginas en un período determinado (localidad temporal y espacial). Si la localidad del problema fuera mayor, es decir, si se accediera repetidamente a un conjunto reducido de páginas cercanas, se esperaría un menor número de fallos de página, incluso con un número reducido de marcos de página. Esto se debe a que las páginas necesarias estarían constantemente en la memoria RAM, reduciendo la necesidad de reemplazos y accesos a la memoria virtual.

Por otro lado, si la localidad del problema fuera menor (acceso disperso a muchas páginas diferentes y no repetitivo), los fallos de página aumentarían considerablemente, ya que las páginas necesarias estarían en constante rotación entre la RAM y la memoria virtual, causando más accesos a la swap y incrementando los tiempos de respuesta. En este caso, incluso con un mayor número de marcos de página, el rendimiento sería peor, ya que las páginas relevantes no permanecerían en la RAM el tiempo suficiente para reducir los fallos de página.

Diagrama UML

Link a lucidchart: https://lucid.app/lucidchart/828ce297-a8d4-419f-94af-cadaf0828d17/edit?viewport_loc=-1594%2C-2192%2C2758%2C3076%2C0_0&invitationId=inv_b08b2951-af2f-4113-954c-96eb8a5eb2f4