Grupo 8 - Caso 2

Juan Pérez - 202210323
William Pollock - 202221321
Santiago Cruz – 202316267

Contrato de Trabajo del Grupo

1). Procedimiento del equipo

Dia, hora y lugar de reuniones periódicas

- Dia: Dentro de lo posible fines de semana
- Hora: En horas de la tarde(>6pm)
- Lugar: Virtual

Método de comunicación preferido

- Grupo de WhatsApp para avisos generales.
- Google meet para reuniones.
- Correo electrónico para un acercamiento inicial.

Política de toma de decisiones

- Tendremos en cuenta un consenso siempre que sea posible.
- Si no hay un acuerdo, se decidirá por mayoría de votos.

Método para establecer y seguir agendas de reuniones

- La agenda deberá ser establecida por los integrantes al menos (dentro de lo posible) un día antes de la reunión.
- Toda la información será enviada al grupo de WhatsApp.

Método de manteniendo de registros:

- No se consideró necesario un seguimiento de registros, pues, las reuniones son más bien puntuales, y no largas ni complejas.

Cuidado de lo social

Las reuniones no son formales, por lo tanto, se tiene un ambiente de compañerismo que ayuda a fortalecer la confianza y el trabajo en equipo.

2). Expectativas del equipo

Calidad del trabajo

Estándares de proyecto:

- Los documentos deben estar bien estructurados y con ciertos estándares.
- El informe debe seguir un estilo uniforme y revisarse antes de la entrega.
- Cada miembro debe leer y comprobar los entregables finales antes del envío.

Estrategias para cumplir con estos estándares:

- Se revisarán los avances en cada reunión.
- Se establecerán plazos internos para revisión y corrección.

Participación del equipo

Estrategias para asegurar la cooperación y distribución equitativa de tareas:

- Cada miembro elegirá su tarea según sus fortalezas y preferencias, pero todos los miembros participaran activamente en las responsabilidades de los demás.

Estrategias para fomentar la inclusión de ideas:

- Cada reunión incluirá momentos donde cada miembro podrá proponer ideas.
- Se hará una votación para seleccionar las mejores propuestas/ideas.

Estrategias para mantenerse en la tarea:

- Se mantendrá una comunicación constante donde comunicaremos avances y demás información.

Preferencias de liderazgo:

- No contamos con un “líder” como tal, y ninguno de los integrantes asumió nunca un papel de liderazgo, más bien todos nos complementamos entre sí.

Responsabilidad personal

Asistencia, puntualidad y participación esperada:

- Se espera que todos asistan a las reuniones salvo por una causa de fuerza mayor.
- Esencialmente se espera una participación regular en el grupo de WhatsApp.

Nivel esperado de responsabilidad en tareas y plazos:

- Cada miembro debe cumplir con su tarea antes de la fecha límite acordada.
- Si alguien no puede completar su tarea, debe avisar con antelación para encontrar una solución.

Nivel esperado de comunicación:

- Responder mensajes dentro de lo posible a un tiempo menor de un día.
- Notificar cualquier retraso o problema con las tareas.

Nivel esperado de compromiso con el equipo:

- Respetar las decisiones grupales.
- Contribuir de manera equitativa en el trabajo y discusiones.

3). Consecuencias por incumplimiento

Manejo de infracciones

- La primera falta se manejará con un aviso verbal en privado.
- Si hay una segunda falta, se discutirá en grupo y se redistribuirán las tareas si es necesario.

Acciones en caso de reincidencia

- Si un miembro sigue sin cumplir, se informará al profesor para definir una solución.
- En casos extremos, se puede pedir su exclusión del equipo o reasignación en caso de que un miembro no se encuentre satisfecho con los demás.

4). Acciones para fortalecer al equipo

Reconocimiento de aportes y logros

- Se destacarán los logros individuales en reuniones y mensajes grupales.
- Se hará una evaluación(coevaluación) anónima al final del proyecto para reconocer el esfuerzo de cada uno (y determinar la nota correspondiente).
- **Celebración de metas alcanzadas**
- Al finalizar el proyecto, nos felicitaremos mutuamente y nos desearemos lo mejor

Compromiso de los miembros

Yo, como miembro del equipo, confirmo que:
a) Participé en la formulación de este acuerdo.

b) Me comprometo a cumplir con estos términos.
c) Acepto las consecuencias establecidas en caso de incumplimiento.

Nombre y fecha:

1. Santiago Cruz – 31/03/2025
2. William Pollock – 29/03/2025
3. Juan Pérez – 01/04/2025

Descripción del Algoritmo para Generar Referencias de Página (Opción Uno)

Este algoritmo se encarga de generar un archivo con las referencias de memoria virtual utilizadas al aplicar un filtro de Sobel (*applySobel*) obre una imagen BMP de 24 bits (de bit depth) en color.

1). Cálculo de desplazamiento en memoria

Clase GeneradorReferencias.java

Primero, se determina la ubicación en memoria virtual de las matrices involucradas.

```
int bytesImagen = alto * ancho * 3;       //bytes(RGB)
int bytesFiltro = 3 * 3 * 4;              //bytes(enteros)
int bytesRespuesta = alto * ancho * 3;    //bytes
```

Después, se organizan los desplazamientos en memoria.

```
int offsetImagen = 0;
int offsetFiltroX = offsetImagen + bytesImagen;
int offsetFiltroY = offsetFiltroX + bytesFiltro;
int offsetRespuesta = offsetFiltroY + bytesFiltro;
```

2). Generación de referencias

Acceso a la imagen (Referencias de lectura “R”)
Se recorren los pixeles de la imagen, evitando los bordes (**i, j** en el rango [1, alto -1] y [1, ancho-1]), y para cada píxel (**i, j**), se acceden sus 9 pixeles vecinos en los kernels 3x3.

```
for (int i = 1; i<alto -1;i++){
    for (int j = 1;j<ancho-1;j++){
        for (int ki = -1; ki <=1; ki++){
            for (int kj = -1; kj <=1;kj++){
                int fila = i + ki;
                int col = j + kj;

                for (int c=0; c<3; c++){
                    int pos = (fila*ancho + col) * 3 + c;
                    int dir = offsetImagen + pos;
                    referencias.add(crearReferencia( nombre: "Imagen[" + fila + "][" + col + "]." + canal(c), dir, tamPagina, tipo: 'R'));
                }
```

Cada referencia incluye una dirección de memoria calculada (dir = offsetImagen + pos), canal de color (R, G, B) y un tipo de acceso ("R" para lectura).

Acceso a los filtros SobelX y SobelY (Referencias de lectura "R")

Para cada posición (**ki, kj**) del kernel 3x3, se generan referencias a los filtros SobelX y SobelY.

```
int idx = (ki+1) * 3 + (kj +1);
int dirX = offsetFiltroX + idx*4;
int dirY = offsetFiltroY +idx*4;

for (int c = 0; c < 3;c++){
    referencias.add(crearReferencia( nombre: "SOBEL_X[" + (ki+1) + "][" + (kj+1) + "]", dirX, tamPagina, tipo: 'R'));
    referencias.add(crearReferencia( nombre: "SOBEL_Y[" + (ki+1) + "][" + (kj+1) + "]", dirY, tamPagina, tipo: 'R'));
```

Cada referencia apunta a una celda del filtro y se marca como lectura ("R").

Escritura en la matriz de respuesta (Referencias de escritura "W")

Una vez procesado el píxel (**i, j**), se almacenan los resultados en la matriz **respuesta.**

```
int pixelIndex = (i*ancho +j)*3;D
for (int c = 0; c < 3; c++){
    int dir = offsetRespuesta + pixelIndex + c;
    referencias.add(crearReferencia( nombre: "Rta[" + i + "][" + j + "]." + canal(c), dir, tamPagina, tipo: 'W'));
}
```

Cada referencia indica una escritura ("W") en la dirección correspondiente.

3). Cálculo de páginas y generación del archivo

Calculo de cantidad de páginas virtuales.

```
int totalReferencias = referencias.size();
int totalPaginas = (offsetRespuesta + bytesRespuesta + tamPagina -1)/tamPagina;
```

NR = referencias.size(): Total de accesos a memoria.

NP = (offsetRespuesta + bytesRespuesta + tamPagina - 1) / tamPagina: Total de páginas necesarias.

Escritura del archivo de referencias

```
    try (PrintWriter writer = new PrintWriter(new FileWriter(archivoSalida))){
        writer.println("TP=" + tamPagina);
        writer.println("NF=" + alto);
        writer.println("NC=" +ancho);
        writer.println("NR=" + totalReferencias);
        writer.println("NP=" + totalPaginas);
        for (Referencia r: referencias){
            writer.println(r.toString());
        }
        System.out.println("Archivo generado: " + archivoSalida);
    }catch (IOException e){
        e.printStackTrace();
    }
}
```

Se escribe un archivo con los datos generados, incluyendo las referencias de memoria virtual.

Función para crear una referencia de memoria

```
private static Referencia crearReferencia(String nombre, int direccion, int tamPagina, char tipo){  4 usages
    int pagina = direccion / tamPagina;
    int offset = direccion % tamPagina;
    return new Referencia(nombre, pagina, offset, tipo);
}
```

Calcula la página virtual (dirección/tamPagina) y el desplazamiento (dirección % tamPagina).

Función para identificar el canal del color

```
    private static String canal(int c){  2 usages
        if (c == 0) return "r";
        else if (c == 1) return "g";
        else if (c == 2) return "b";
        else return "?";
    }
}
```

Devuelve “r”, “g” o “b” según el índice del canal de color.

Clase Referencia.java

```java
public class Referencia {  10 usages
    public String nombre;  2 usages
    public int pagina;  4 usages
    public int offset;  2 usages
    public char tipoAcceso;  3 usages

    public Referencia(String nombre, int pagina, int offset, char tipoAcceso){  2 usages
        this.nombre = nombre;
        this.pagina = pagina;
        this.offset = offset;
        this.tipoAcceso = tipoAcceso;
    }

    @Override
    public String toString() { return nombre + "," + pagina + "," + offset + "," + tipoAcceso; }
}
```

Esta clase tiene como función organizar cómo los datos de las referencias serán imprimidos dentro del archivo de salida. Entre los datos que serán ordenados están el *nombre* de la imagen, la *página*, el desplazamiento (*offset*) y el tipo de acceso (lectura o escritura). Lo siguiente es un ejemplo de cómo se vería esto en el archivo de salida (gracias a *toString*):

Imagen[0][0].r, 0, 0, R

Estructuras de Datos Usadas Para Sistema de Paginación (Opción dos)

PaginaVirtual.java

```java
public class PaginaVirtual {
    public int numero;
    public boolean enRAM;
    public boolean bitR;
    public boolean bitM;
    public long ultAcc;

    public PaginaVirtual(int numero){
        this.numero = numero;
        this.enRAM = false;
        this.bitR = false;
        this.bitM = false;
        this.ultAcc = System.nanoTime();
    }
}
```

Se encarga de guardar los datos de las páginas virtuales, tales como si está en memoria principal (*enRAM*), si fue referenciada recientemente (*bitR*), si fue modificada (*bitM*) y la última vez que fue accedida (*ultAcc*). Esta clase también tendrá un rol importante en la siguiente:

ActualizadorBits (clase interna de *SimuladorMemoria*)

```
static class ActualizadorBits extends Thread{
    private final Map<Integer, PaginaVirtual> tablaPaginas;

    public ActualizadorBits(Map<Integer, PaginaVirtual> tablaPaginas){
        this.tablaPaginas = tablaPaginas;
    }

    public void run(){
        while (!isInterrupted()){
            try{
                Thread.sleep(millis:1);
                synchronized (tablaPaginas){
                    for (PaginaVirtual p: tablaPaginas.values()){
                        p.bitR = false;
                    }
                }
            }catch (InterruptedException e){
                break;
            }
        }
    }
}
```

Esta clase se encarga de reiniciar los bits R de todas las páginas cada 1 milisegundo.

ProcesadorReferencias (clase interna de *SimuladorMemoria*)

```
static class ProcesadorReferencias extends Thread{
    private final List<Referencia> referencias;
    private final SimuladorMemoria simulador;

    public ProcesadorReferencias(List<Referencia> referencias, SimuladorMemoria simulador){
        this.referencias = referencias;
        this.simulador = simulador;
    }

    public void run(){
        try{
            int contador = 0;
            for (Referencia ref: referencias){
                simulador.accederPagina(ref);
                contador++;
                if (contador % 10000 == 0){
                    Thread.sleep(millis:1);
                }
            }
        }catch (InterruptedException e){
            System.out.println(x:"Procesador interrumpido");
        }
    }
}
```

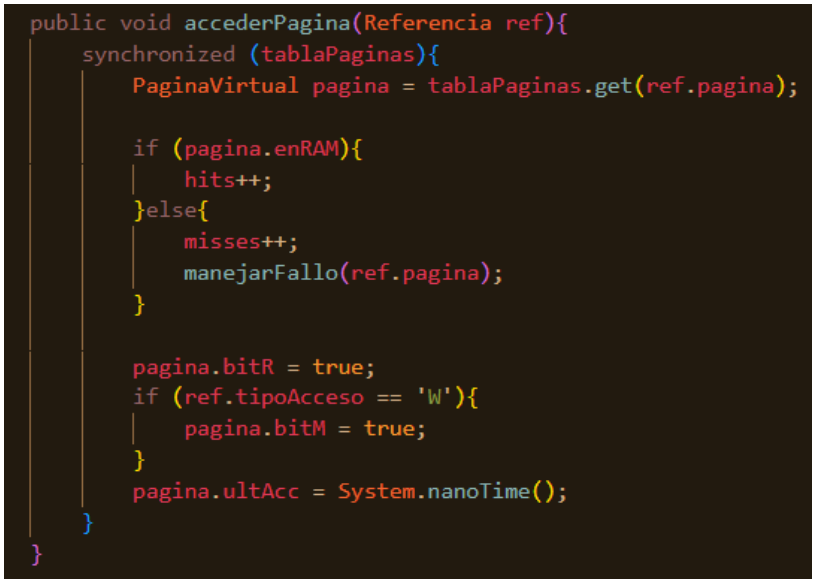
Esta clase es un hilo que simula el proceso del usuario leyendo el archivo de referencias. Decidimos que cada 10 mil accesos se duerme 1 milisegundo para simular el paso del tiempo.

SimuladorMemoria.java

```
public void simular(String archivoReferencias){
    List<Referencia> referencias = new ArrayList<>();

    try (BufferedReader br = new BufferedReader(new FileReader(archivoReferencias))){
        String linea;
        while (!(linea = br.readLine()).startsWith(prefix:"Imagen[")&& linea != null);

        do{
            String[] partes = linea.split(regex:",");
            String nombre = partes[0];
            int pagina = Integer.parseInt(partes[1]);
            int offset = Integer.parseInt(partes[2]);
            char tipo = partes[3].charAt(index:0);
            referencias.add(new Referencia(nombre, pagina, offset, tipo));

            synchronized (tablaPaginas){
                tablaPaginas.putIfAbsent(pagina, new PaginaVirtual(pagina));
            }
        }while ((linea = br.readLine()) != null);
    }catch (IOException e){
        e.printStackTrace();
        return;
    }

    ActualizadorBits actualizador = new ActualizadorBits(tablaPaginas);
    ProcesadorReferencias procesador = new ProcesadorReferencias(referencias, this);
    actualizador.start();
    procesador.start();

    try{
        procesador.join();
        actualizador.interrupt();
        actualizador.join();
    }catch (InterruptedException e){
        e.printStackTrace();
    }

    imprimirResultados(referencias.size());
}
```

El primer método (*simular*) se encarga de cargar todas las referencias desde el archivo de salida .txt generado en la opción uno. Se crea una lista para guardar las referencias y se saltan las líneas del encabezado del archivo hasta llegar a las líneas que contienen referencias. Después, se divide cada línea y se crea un objeto de tipo Referencia que se agrega a la lista.

Cuando ya se guardan las referencias, se inician los hilos de simulación con ayuda de *ActualizadorBits* (reinicia los R bits periódicamente) y *ProcesadorReferencias* (simula al proceso accediendo memoria). Cuando ya todos los procesos hayan terminado, se imprime el resultado.



```
public void accederPagina(Referencia ref){
    synchronized (tablaPaginas){
        PaginaVirtual pagina = tablaPaginas.get(ref.pagina);

        if (pagina.enRAM){
            hits++;
        }else{
            misses++;
            manejarFallo(ref.pagina);
        }

        pagina.bitR = true;
        if (ref.tipoAcceso == 'W'){
            pagina.bitM = true;
        }
        pagina.ultAcc = System.nanoTime();
    }
}
```

En cuanto al siguiente método (*accederPagina*), se recupera la información de la página accedida. Si está dentro del RAM se suma un hit, mientras que si no lo está se suma una miss y se invoca *manejarFallo* (el cual será mostrado después). Finalmente, se marca que la página que fue referenciada (si es de escritura también M = true) y se actualiza la marca de tiempo.

```
private void manejarFallo(int paginaFaltante){
    synchronized(marcosRAM){
        if (marcosRAM.size() < numMarcos){
            marcosRAM.add(paginaFaltante);
            synchronized(tablaPaginas){
                tablaPaginas.get(paginaFaltante).enRAM = true;
            }
        }else{
            List<Integer>[] clases = new List[4];
            for (int i = 0; i < 4; i++) clases[i] = new ArrayList<>();
            synchronized (tablaPaginas){
                for (int p: marcosRAM){
                    PaginaVirtual pv = tablaPaginas.get(p);
                    int clase = (pv.bitR ? 2:0) + (pv.bitM ? 1:0);
                    clases[clase].add(p);
                }
            }
            int paginaAReemp = -1;
            for (List<Integer> clase: clases){
                if (!clase.isEmpty()){
                    paginaAReemp = clase.get(index:0);
                    break;
                }
            }

            if (paginaAReemp != -1){
                synchronized (tablaPaginas){
                    tablaPaginas.get(paginaAReemp).enRAM = false;
                    tablaPaginas.get(paginaFaltante).enRAM = true;
                }
                marcosRAM.remove(paginaAReemp);
                marcosRAM.add(paginaFaltante);
            }
        }
    }
}
```

Este método (*manejarFallo*) aplica el algoritmo de reemplazo NRU si la página no está dentro del RAM. Lo primero que hace es verificar si hay espacio en el RAM. Si sí hay, se agrega la página dentro del RAM; si no, se aplica NRU (se crean 4 listas, una por cada clase del algoritmo):

Clase 0: R = 0, M = 0
Clase 1: R = 0, M = 1
Clase 2: R = 1, M = 0
Clase 3: R = 1, M = 1

Después, cada página es clasificada según sus bits y se elige la primera página de la clase más baja, ya que es más favorable para ser reemplazada. El método se acaba realizando el reemplazo.

```java
private void imprimirResultados(int totalReferencias){
    System.out.println("Total referencias: " + totalReferencias);
    System.out.println("Hits: " + hits);
    System.out.println("Fallas de página: " + misses);
    System.out.printf(format:"Porcentaje de hits: %.2f %%\n", (100.0 * hits) / totalReferencias);
    long tiempoHit = 50;
    long tiempoMiss = 10000000;
    long tiempoTotal = hits * tiempoHit + misses * tiempoMiss;
    System.out.printf(format:"Tiempo real simulado: %.2f ms\n", tiempoTotal / 1000000.0);

}
```

Finalmente, se imprimen los resultados en consola. En este caso, el tiempo real es una simulación de cuánto habría tardado el proceso según la cantidad de hits y misses y los tiempos estimados por acceso. Como estimados aplicamos 50 ns para cada hit y 10 ms (10,000,000 ns) para cada miss. Cuando ya se tienen estos valores, se calcula el tiempo total multiplicando los hits y misses por sus respectivos valores de tiempo y convirtiéndolos en milisegundos.

Esquema de Sincronización

Durante la simulación, se utilizan dos hilos que acceden simultáneamente a estructuras compartidas: ***ProcesadorReferencias*** y ***ActualizadorBits***; ambos threads leen y modifican estructuras como *marcosRAM* y *tablaPaginas.* Por esta razón, se usa bloqueo (a través de *synchronized*) para evitar condiciones de carrera y garantizar que los datos se mantengan consistentes.

En cuanto a *tablaPaginas*, *ProcesadorReferencias* marca los bits R y M mientras que *ActualizadorBits* pone todos los bits R igual a falso. En las imágenes anteriores del código se puede ver que dentro de algunos de los métodos está la línea *synchronized (tablaPaginas).* Agregar estas áreas críticas de sincronización previene que los dos hilos accedan y/o modifiquen la misma página al mismo tiempo, lo que pudiera haber causado bits R o M con valores incorrectos o errores por acceso concurrente.

Ahora yendo hacia *marcosRAM*, este sólo es modificado por *manejarFallo*, pero hay que tener en cuenta que este método contiene a *tablaPaginas*. Agregar la sincronización en este caso previene corrupciones de la cola (que se modifique mientras se recorre/reemplaza una página).

Tabla de Datos Recopilados

Animal.bmp (500x300)

Página de 64B, animal.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	12465936	5147706	7318230	41,29%

2	12465936	5330282	7135654	42,76%
4	12465936	7612509	4853427	61,07%
6	12465936	10248495	2217441	82,21%
8	12465936	11748824	717112	94,25%
10	12465936	12360399	105537	99,15%
13	12465936	12433209	32727	99,74%
15	12465936	12434498	31438	99,75%
17	12465936	12435537	30399	99,76%
20	12465936	12436061	29875	99,76%
Pagina de 128B, animal.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	12465936	5170995	7294941	41,48%
2	12465936	5362260	7103676	43,02%
4	12465936	9213888	3252048	73,91%
6	12465936	11748230	717706	94,24%
8	12465936	12368460	97476	99,22%
10	12465936	12449222	16714	99,87%
13	12465936	12450070	15866	99,87%
15	12465936	12451184	14752	99,88%
17	12465936	12451612	14324	99,89%
20	12465936	12451939	13997	99,89%
Pagina de 256B, animal.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	12465936	5182734	7283202	41,58%
2	12465936	5370355	7095581	43,08%
4	12465936	10636530	1829406	85,32%
6	12465936	12044143	421793	96,62%
8	12465936	12454436	11500	99,91%
10	12465936	12458251	7685	99,94%
13	12465936	12458809	7127	99,94%
15	12465936	12458934	7002	99,94%
17	12465936	12458927	7009	99,94%
20	12465936	12458940	6996	99,94%
Pagina de 512B, animal.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	12465936	5188794	7277142	41,62%
2	12465936	5371774	7094162	43,09%
4	12465936	11205526	1260410	89,89%
6	12465936	12278819	187117	98,50%

8	12465936	12461623	4313	99,97%
10	12465936	12462432	3504	99,97%
13	12465936	12462436	3500	99,97%
15	12465936	12462440	3496	99,97%
17	12465936	12462443	3493	99,97%
20	12465936	12462452	3484	99,97%
Pagina de 1024B, animal.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	12465936	9644277	2821659	77,37%
2	12465936	9803752	2662184	78,64%
4	12465936	11857700	608236	95,12%
6	12465936	12461377	4559	99,96%
8	12465936	12465058	878	99,99%
10	12465936	12465058	878	99,99%
13	12465936	12465058	878	99,99%
15	12465936	12465058	878	99,99%
17	12465936	12465058	878	99,99%
20	12465936	12465058	878	99,99%
Pagina de 2048B, animal.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	12465936	9647777	2818159	77,39%
2	12465936	9811364	2654572	78,71%
4	12465936	12180945	284991	97,714%
6	12465936	12465495	441	99,996%
8	12465936	12465495	441	99,996%
10	12465936	12465495	441	99,996%
13	12465936	12465495	441	99,996%
15	12465936	12465495	441	99,996%
17	12465936	12465495	441	99,996%
20	12465936	12465495	441	99,996%
Pagina de 4096B, animal.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	12465936	9657517	2808419	77,47%
2	12465936	9819324	2646612	78,77%
4	12465936	12452947	12989	99,896%
6	12465936	12465716	220	99,998%
8	12465936	12465716	220	99,998%
10	12465936	12465716	220	99,998%
13	12465936	12465716	220	99,998%

15	12465936	12465716	220	99,998%
17	12465936	12465716	220	99,998%
20	12465936	12465716	220	99,998%

Apaisaje.bmp (356x100)

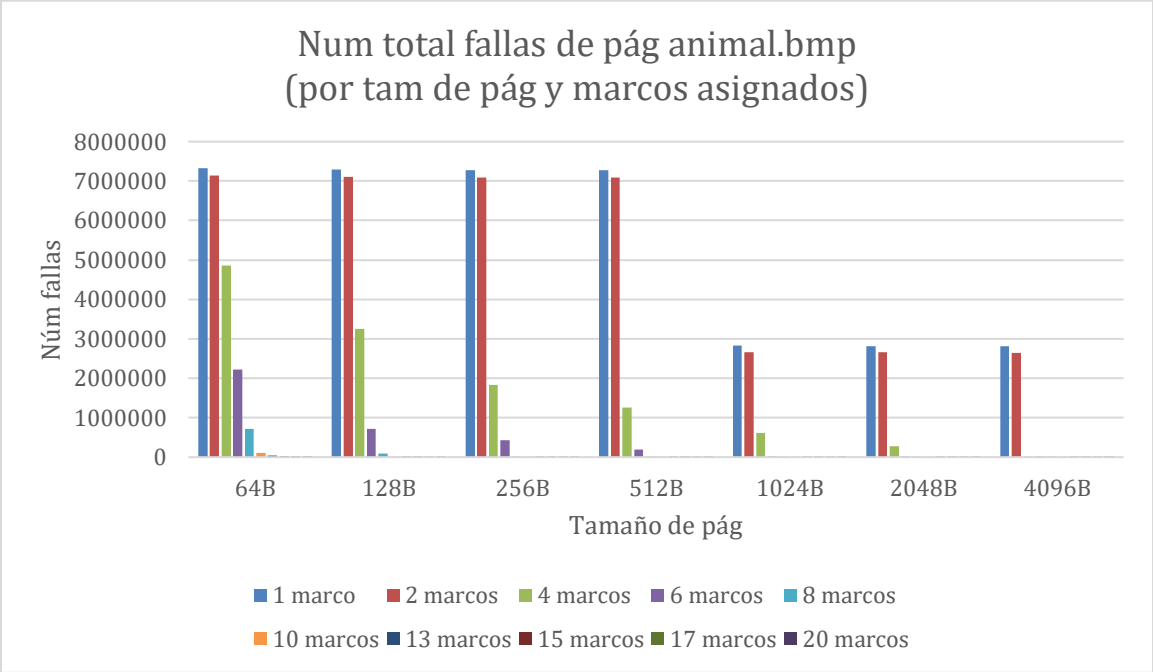
Página de 64B, apaisaje.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	2914128	1550275	1363853	53,20%
2	2914128	1588228	1325900	54,50%
4	2914128	1920505	993623	65,90%
6	2914128	2395656	518472	82,21%
8	2914128	2738544	175584	93,97%
10	2914128	2871474	42654	98,54%
13	2914128	2906414	7714	99,74%
15	2914128	2906610	7518	99,74%
17	2914128	2906989	7139	99,76%
20	2914128	2907041	7087	99,76%
Pagina de 128B, apaisaje.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	2914128	2249640	664488	77,20%
2	2914128	2286273	627855	78,45%
4	2914128	2575590	338538	88,38%
6	2914128	2844769	69359	97,62%
8	2914128	2909539	4589	99,84%
10	2914128	2910570	3558	99,88%
13	2914128	2910660	3468	99,88%
15	2914128	2910743	3385	99,88%
17	2914128	2910853	3275	99,89%
20	2914128	2910852	3276	99,89%
Pagina de 256B, apaisaje.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	2914128	2252357	661771	77,29%
2	2914128	2289294	624834	78,56%
4	2914128	2741569	172559	94,08%
6	2914128	2904645	9483	99,67%
8	2914128	2912318	1810	99,94%

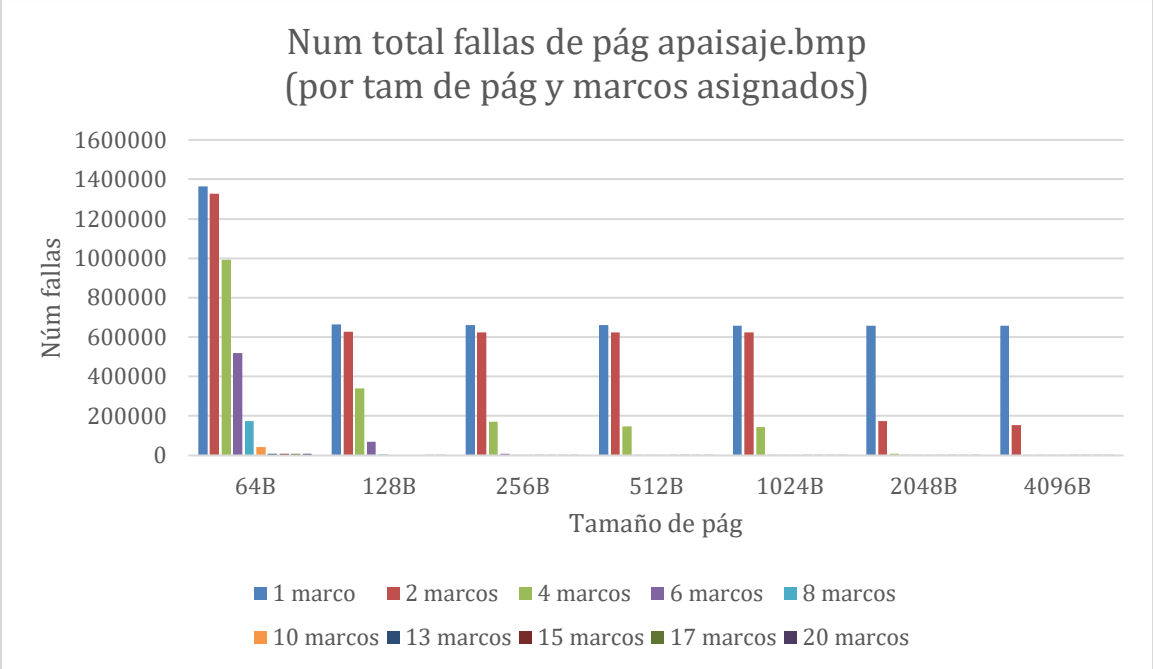
10	2914128	2912394	1734	99,94%
13	2914128	2912488	1640	99,94%
15	2914128	2912488	1640	99,94%
17	2914128	2912489	1639	99,94%
20	2914128	2912498	1630	99,94%
Pagina de 512B, apaisaje.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	2914128	2254228	659900	77,36%
2	2914128	2291145	622983	78,62%
4	2914128	2768133	145995	94,99%
6	2914128	2913086	1042	99,96%
8	2914128	2913230	898	99,97%
10	2914128	2913310	818	99,97%
13	2914128	2913316	812	99,97%
15	2914128	2913323	805	99,97%
17	2914128	2913331	797	99,97%
20	2914128	2913333	795	99,97%
Pagina de 1024B, apaisaje.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	2914128	2254906	659222	77,38%
2	2914128	2291700	622428	78,64%
4	2914128	2771790	142338	95,12%
6	2914128	2913858	270	99,99%
8	2914128	2913920	208	99,99%
10	2914128	2913920	208	99,99%
13	2914128	2913920	208	99,99%
15	2914128	2913920	208	99,99%
17	2914128	2913920	208	99,99%
20	2914128	2913920	208	99,99%
Pagina de 2048B, apaisaje.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	2914128	2255444	658684	77,40%
2	2914128	2739753	174375	94,02%
4	2914128	2904431	9697	99,667%
6	2914128	2914024	104	99,996%
8	2914128	2914024	104	99,996%
10	2914128	2914024	104	99,996%
13	2914128	2914024	104	99,996%
15	2914128	2914024	104	99,996%

17	2914128	2914024	104	99,996%
20	2914128	2914024	104	99,996%
Pagina de 4096B, apaisaje.bmp				
Marcos Asignados	Total referencias	Hits	Fallas	% hits
1	2914128	2256290	657838	77,43%
2	2914128	2760891	153237	94,74%
4	2914128	2911247	2881	99,901%
6	2914128	2914076	52	99,998%
8	2914128	2914076	52	99,998%
10	2914128	2914076	52	99,998%
13	2914128	2914076	52	99,998%
15	2914128	2914076	52	99,998%
17	2914128	2914076	52	99,998%
20	2914128	2914076	52	99,998%

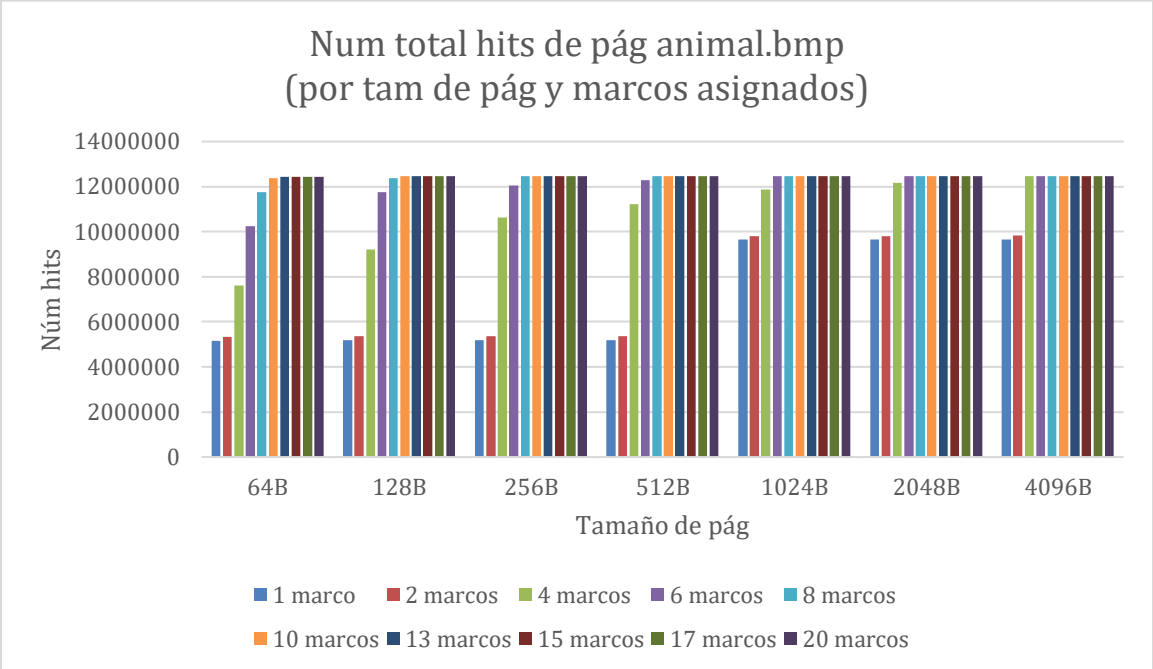
Gráficas del Comportamiento del Sistema

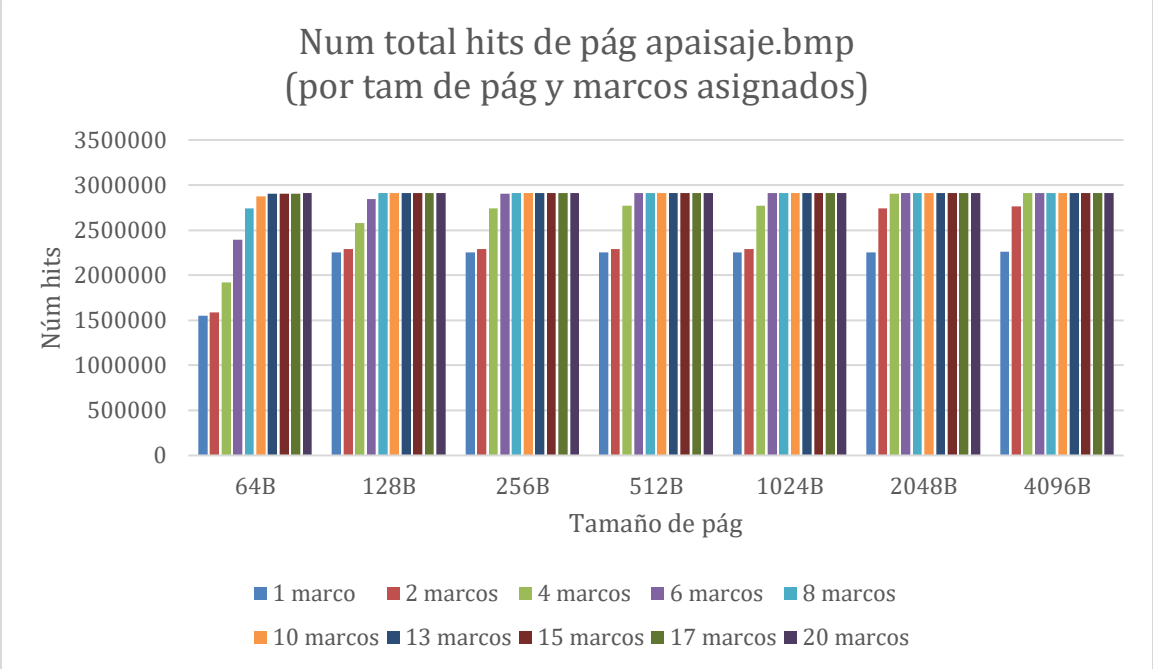
Marcos asignados vs. número de misses





Marcos asignados vs. número de hits





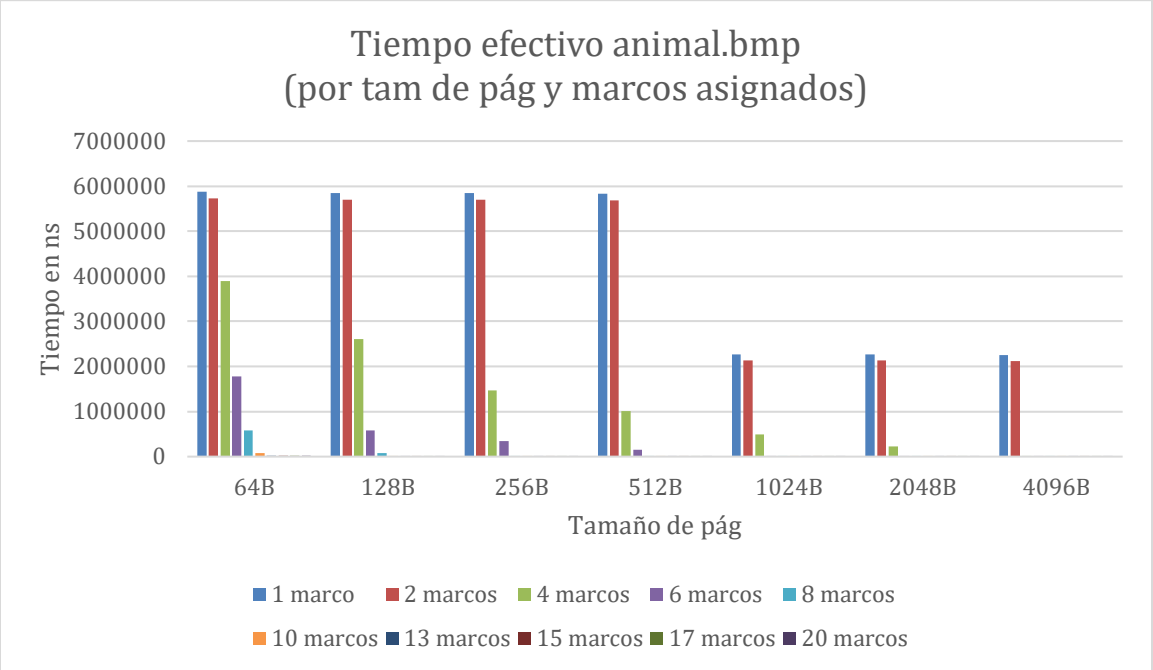
Escenarios Adicionales Considerados

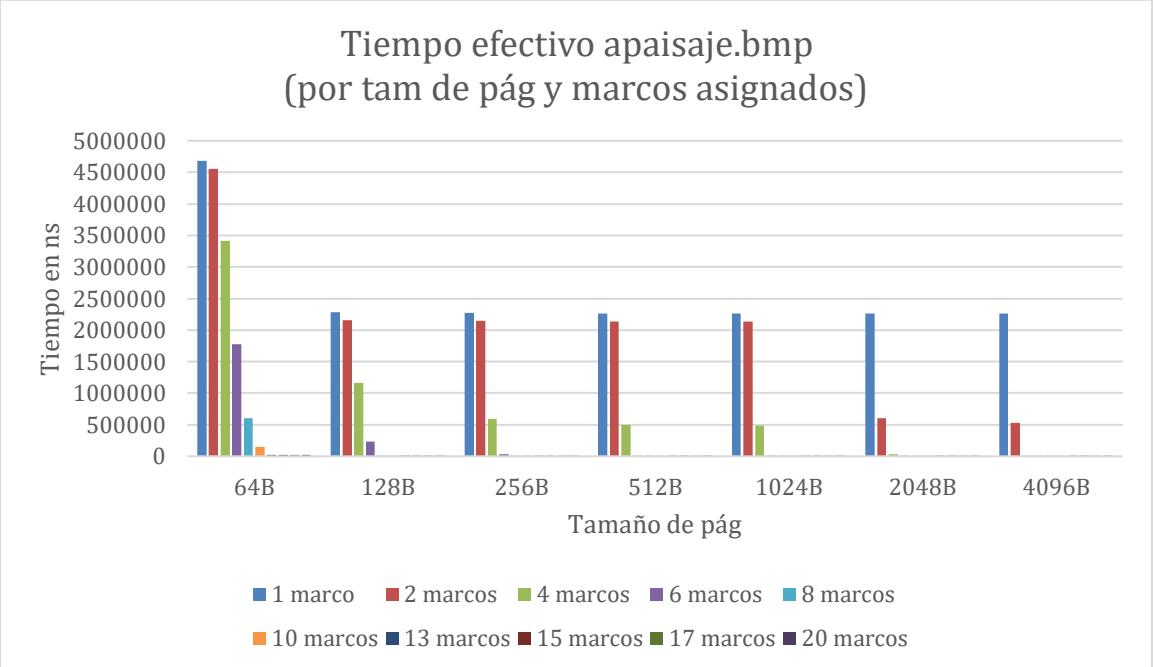
Otras configuraciones que nos permitieron entender cómo afecta la memoria virtual el desempeño del programa fueron la modificación de variables como el número de marcos, el tamaño de página y el tipo de imagen. Basado en los resultados anteriores, caímos en cuenta que:

- Asignar una menor cantidad de marcos genera más misses; esto disminuye el porcentaje de hits y aumenta el tiempo total de ejecución
- Aumentar los marcos mejora el rendimiento hasta que casi todas las páginas activas caben en el RAM
- Las páginas pequeñas generan más divisiones y potencialmente más fallos, mientras que las páginas más grandes reducen el número de páginas total (esto puede desperdiciar memoria si no se usan a su totalidad)

En conclusión, la asignación de memoria virtual tiene un impacto directo en el desempeño del programa y puede perjudicar altamente la ejecución de este si no se le

presta atención.





Interpretación de los Resultados

De acuerdo con los resultados obtenidos podemos hacer las siguientes tendencias con el programa:

Mas marcos, menos fallos, con una cantidad menor de marcos, podemos ver que las fallas de página son altas. Mientras que, a partir de cierto punto (aproximadamente 10-15 marcos), las fallas se suelen reducir significativamente.

Esto sucede debido a que, a medida que aumentamos los marcos, se pueden retener más páginas en memoria, lo que reduce la necesidad de reemplazo.

Al aumentar el tamaño de página, se requiere menos número de marcos para alcanzar un porcentaje (%) de hits alto, ya que se agrupan más referencias en una sola página (cada una contiene más datos, una referencia cubre mayor proporción de imagen).

Por último, se puede observar una reducción de fallas esperada, con mejoras a medida que aumentaban los marcos, llegando a una estabilización en los valores altos (de marcos asignados).

Análisis del Filtro Sobel y Localidad

Aplicar el filtro Sobel representa un problema de localidad alta. Esto se debe a que, para calcular el valor de cada píxel en el archivo .txt de salida, el algoritmo accede no solo al píxel actual sino a todos los 8 píxeles que lo rodean (en el kernel 3x3); esto también ocurre repetidamente para todo píxel adyacente. Como consecuencia, muchas páginas que ya están dentro del RAM se reutilizan muchas veces antes de ser reemplazadas, lo que causa un alto porcentaje de hits. Esto permite que incluso con una baja cantidad de marcos, el rendimiento se mantenga alto.