Trabajo final Arquitectura de Computadores/ Interfaces y Arquitectura Hardware

> ***Student Outcome 06**: An ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions.*

Los experimentos desempeñan un papel muy importante en ingeniería porque sirve de apoyo en las fases de validación de un servicio, modelo o desarrollo de un componente software o hardware de un sistema en condiciones más cercanas a los escenarios de trabajo, o cuando se quiere estudiar el comportamiento de un sistema.

A lo largo del curso se estudió la estructura y arquitectura de los computadores, donde el objetivo en sí mismo del curso es evaluar cómo las estructuras hardware y componentes software de un computador afectan el rendimiento y la utilidad de los programas de aplicación, tanto en el procesamiento, acceso y almacenamiento de los datos de los programas que los manipulan. Por tal razón la evaluación y análisis del desempeño es el fin último de un trabajo final.

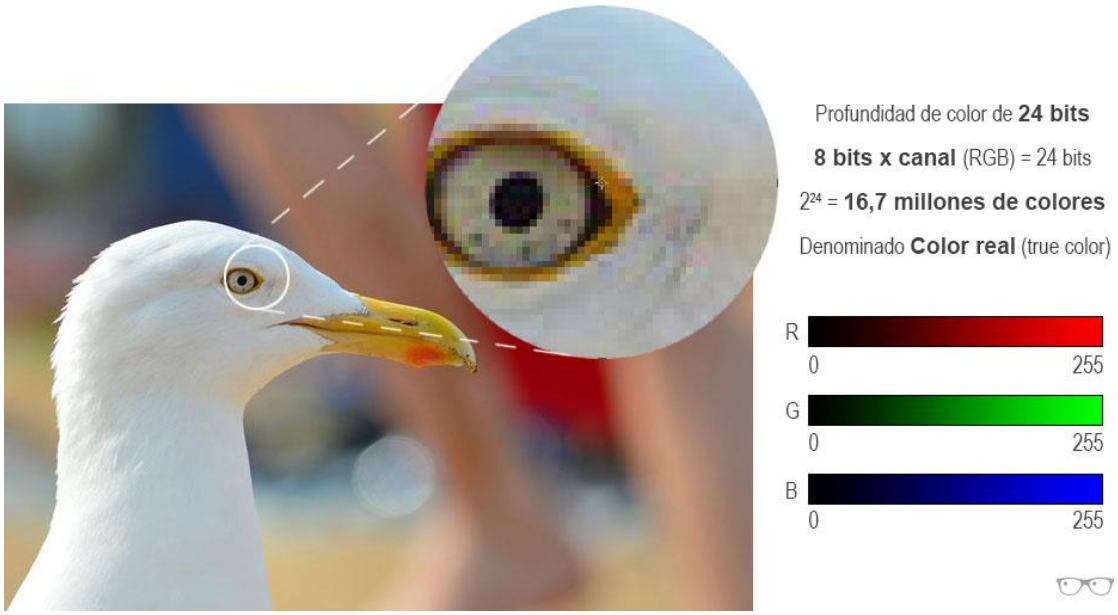
Un componente importante en la arquitectura de un computador es el sistema de memoria, ya que normalmente se convierte en el cuello de botella de los tiempos de ejecución. Entonces la determinación y medición del tiempo de acceso es crucial para tomar decisiones de configuraciones correctas o adecuadas y determinar y explicar desde la perspectiva del programador, como diferentes programas o algoritmos con la misma complejidad algorítmica pero con diferente grado de aprovechamiento del principio de localidad, podrían tener diferencias significativas a la hora de su ejecución en un computador con una configuración determinada y especialmente en su interacción de con el sistema de memoria.

Para comparar y analizar la eficiencia de los algoritmos y que son escritos en lenguaje de alto nivel, establecer una medida precisa de la eficiencia de un algoritmo no es fácil, ya que, si compara con el tiempo de ejecución, ésta respuesta tiene una gran variabilidad. La complejidad de un algoritmo deberá estar relacionada con el número de operaciones elementales necesarias (asignaciones, comparaciones, sumas, restas, multiplicaciones, divisiones, etc.) para resolver el problema.

El estudio de la estructura y arquitectura hardware de un equipo de cómputo permite comprender cuales son los componentes hardware y elementos software que impactan en el desempeño. Dichos factores van desde el tipo de carga de trabajo o exigencia computacional, tipo de CPU, configuración de la jerarquía de memoria del computador, el sistema operativo, compiladores, lenguajes de programación, los tipos de datos utilizados en el programa, entre otros. De allí la importancia de realizar un diseño experimental bien hecho para establecer los niveles de interacción de los factores y el impacto en la variable respuesta en el que se esté interesado medir.

Marco teórico: imágenes de mapa de bits

Un mapa de bits es una matriz rectangular de píxeles donde cada elemento de dicha matriz es un conjunto de bits que especifica el color de cada píxel. El número de bits dedicado a un píxel individual determina el número de colores que se pueden asignar a ese píxel.



Tomado de: https://www.elvisualista.com/2016/05/05/que-es-un-mapa-de-bits-1/

Por ejemplo, si cada píxel se representa mediante 4 bits, a un píxel determinado se le puede asignar uno de 16 colores diferentes (2 ^ 4 = 16). En la tabla siguiente se muestran algunos ejemplos de diferentes profundidades de color que se pueden asignar a un píxel representado por un número determinado de bits y en diferentes formatos.

Profundidad de color	Tipo de imagen	Colores
1bit	Blanco y negro	2 colores
8bits	Escala de grises	256 tonos de gris
4 bits	Color	16 colores
8 bits	Color	256 colores
16 bits	Color	65.000 colores
16 bits	Escala de grises	65.000 tonos de grises
24 bits	Color, RGB	16,7 millones de colores
32 bits	Color, CMYK	16,7 millones de colores
48 bits	Color	281.000 millones de colores

Los archivos de disco que almacenan mapas de bits normalmente contienen uno o más bloques de información que almacenan información como el número de bits por píxel, el número de píxeles de cada fila y el número de filas de la matriz. Este tipo de archivo también puede contener una tabla de colores (a veces denominada paleta de colores). (https://docs.microsoft.com/, s.f.)

De acuerdo a la cantidad de píxeles incluida en el mapa de bits, determina la resolución de la imagen. Cuando hablamos de 1280 x 720, o 1920 x 1080, y no es más que el número de puntos expresado de forma que definan el ancho por el alto. Los mapas de bits, por otra parte, pueden diferenciarse según la cantidad de colores que puede presentar cada uno de los píxeles. Esta información se expresa en bits en potencia de 2. Hoy en día, el mínimo aceptable es 16 bits, siendo 24 y 32 más comunes. Por otro lado, se definen como tipo RGB y RGBA, don éste último acepta un cuarto valor, para producir imágenes traslúcidas además del componente Red, Green y Blue.

Debe tener en cuenta que algunos mapas de bits que se denominan indexados por una paleta, almacena índices en una tabla de colores. Algunos mapas de bits no tienen necesidad de una tabla de colores. Por ejemplo, si un mapa de bits usa 24 bits por píxel, dicho mapa de bits puede almacenar los propios colores en lugar de los índices en una tabla de colores. En la ilustración siguiente se muestra un mapa de bits que almacena los colores directamente (24 bits por píxel) en lugar de usar una tabla de colores. La imagen también muestra una vista ampliada de la imagen correspondiente. En el mapa de bits, FFFFFF representa el blanco, FF0000 representa rojo, 00FF00 representa el verde y 0000FF representa el azul. (https://docs.microsoft.com/, s.f.)

```
0000FF 0000FF 0000FF 0000FF 0000FF 0000FF 0000FF 0000FF
00FF00 FF0000 FFFFFF FF0000 FFFFFF FF0000 FFFFFF 00FF00
00FF00 FFFFFF FF0000 FFFFFF FF0000 FFFFFF FF0000 00FF00
00FF00 FF0000 FFFFFF FF0000 FFFFFF FF0000 FFFFFF 00FF00
00FF00 FFFFFF FF0000 FFFFFF FF0000 FFFFFF FF0000 00FF00
00FF00 FF0000 FFFFFF FF0000 FFFFFF FF0000 FFFFFF 00FF00
00FF00 FFFFFF FF0000 FFFFFF FF0000 FFFFFF FF0000 00FF00
0000FF 0000FF 0000FF 0000FF 0000FF 0000FF 0000FF 0000FF
```

Ejemplo de mapa de bits

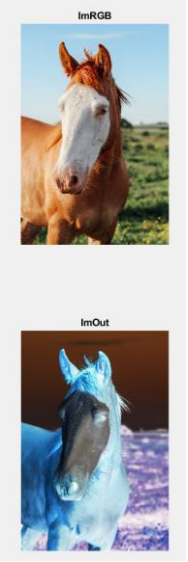
Las imágenes RGB se componen de tres canales de color. Una imagen RGB con 8 bits por píxel cuenta con 256 posibles valores para cada canal, lo que significa más de 16 millones de posibles valores de color. Las imágenes RGB con 8 bits por canal se denominan imágenes de 24 bits (8 bits x 3 canales = 24 bits de datos por píxel). Además de las imágenes de 8 bits por canal, se trabajará con imágenes que contienen 16 o 32 bits por canal (48 y 96 bits por pixel) (Adobe, 2018)

Versiones de algoritmos

El propósito de un algoritmo en el trabajo final es leer uno a uno todos los pixeles de un arreglo de mapas de bits e invertir su color y almacenar de nuevo en los pixeles origen.
El siguiente ejemplo escrito en Matlab ilustra la situación donde se invierte uno a uno los pixeles y se guarda en una segunda matriz, en éste caso se trata de una imagen RGB de 24 bits de profundidad (8 bits por cada canal):

```
ImRGB0 = imread('Horse.jpeg');
ImRGB=ImRGB0;
Rows=size(ImRGB,1);
Columns=size(ImRGB,2);
ImOut=uint8(zeros(Rows,Columns,2));

for R=1: Rows
    for C=1:Columns
    ImRGB(R,C,1)=255-ImRGB(R,C,1);  % Image[i,j].R
    ImRGB(R,C,2)=255-ImRGB(R,C,2); % Image[i,j].G
    ImRGB(R,C,3)=255-ImRGB(R,C,3);  % Image[i,j].B
    end
end
```



Uno de los factores de estudio para el diseño del experimento será la forma u orden en que se recorre la matriz. Las 5 versiones propuestas son las siguientes:

```
%Versión 1
for R=1: Rows
    for C=1:Columns
    ImRGBO(R,C,1)=255-ImRGB(R,C,1);  % Image[i,j].R
    ImRGBO(R,C,2)=255-ImRGB(R,C,2); % Image[i,j].G
    ImRGBO(R,C,3)=255-ImRGB(R,C,3);  % Image[i,j].B
    count = count + 3;
    end
end
%Versión 2
for R=1: Rows
    for C=1:Columns
    ImRGBO(R,C,1)=255-ImRGB(R,C,1);  % Image[i,j].R
    count = count + 1;
    end
end
for R=1: Rows
    for C=1:Columns
    ImRGBO(R,C,2)=255-ImRGB(R,C,2); % Image[i,j].G
    count = count + 1;
    end
end
for R=1: Rows
    for C=1:Columns
    ImRGBO(R,C,3)=255-ImRGB(R,C,3);  % Image[i,j].B
    count = count + 1;
    end
end
```

```
%Versión 3
for C=1:Columns
    for R=1: Rows
    ImRGBO(R,C,1)=255-ImRGB(R,C,1);  % Image[i,j].R
    ImRGBO(R,C,2)=255-ImRGB(R,C,2); % Image[i,j].G
    ImRGBO(R,C,3)=255-ImRGB(R,C,3);  % Image[i,j].B
    count = count + 3;
    end
end

%Versión 4
for R=1: Rows
    for C=1:Columns
    ImRGBO(R,C,1)=255-ImRGB(R,C,1);  % Image[i,j].R
    count = count + 1;
    end
end
for R=Rows :-1:1
    for C=Columns:-1:1
    ImRGBO(R,C,2)=255-ImRGB(R,C,2); % Image[i,j].G
    ImRGBO(R,C,3)=255-ImRGB(R,C,3);  % Image[i,j].B
    count = count + 2;
    end
end

for R=1:2: Rows -1
    for C=1:2:Columns-1

    ImRGBO(R,C,1)=255-ImRGB(R,C,1);  % Image[i,j].R
    ImRGBO(R,C,2)=255-ImRGB(R,C,2); % Image[i,j].G
    ImRGBO(R,C,3)=255-ImRGB(R,C,3);  % Image[i,j].B

    ImRGBO(R,C+1,1)=255-ImRGB(R,C+1,1);  % Image[i,j+1].R
    ImRGBO(R,C+1,2)=255-ImRGB(R,C+1,2);  % Image[i,j+1].G
    ImRGBO(R,C+1,3)=255-ImRGB(R,C+1,3);  % Image[i,j+1].B

    ImRGBO(R+1,C,1)=255-ImRGB(R+1,C,1);  % Image[i+1,j].R
    ImRGBO(R+1,C,2)=255-ImRGB(R+1,C,2);  % Image[i+1,j].G
    ImRGBO(R+1,C,3)=255-ImRGB(R+1,C,2);  % Image[i+1,j].B

    ImRGBO(R+1,C+1,1)=255-ImRGB(R+1,C+1,1);  % Image[i+1,j+1].R
    ImRGBO(R+1,C+1,2)=255-ImRGB(R+1,C+1,2);  % Image[i+1,j+1].G
    ImRGBO(R+1,C+1,3)=255-ImRGB(R+1,C+1,3);  % Image[i+1,j+1].B
    count = count + 12;
    end
end
```

La variable count no debe ir en el experimento, es solo un paso previo para verificar que todas las versiones tienen la misma complejidad algorítmica, para éste caso:
count v1= 4500000
count v2= 4500000
count v3= 4500000
count v4= 4500000
count v5= 4500000

Objetivo y condiciones trabajo final

El trabajo final tiene como objetivo de analizar y emitir un juicio sobre aspectos del comportamiento o desempeño de un sistema de cómputo aplicando la metodología de conducción de experimentos y la teoría de la arquitectura y organización de computadores

Específicamente, cuantificar y explicar el impacto en el desempeño que tiene 5 versiones equivalentes de algoritmos en varían en la localidad espacial al recorrer sus pixeles de una imagen. También se desea estudiar el impacto del tamaño de las imágenes en el tiempo de respuesta normalizado en la ejecución de la operación de invertir el color y el impacto que tiene el sistema de memoria del sistema. También sería interesante comparar dos lenguajes de programación distintos.

Condiciones del estudio experimental:

a) Implementar los 5 algoritmos propuestos en una aplicación de consola usando visual Studio y empleando dos lenguajes de programación diferentes. Estas 5 versiones deben invertir el color de pixeles de una imagen RGB de mapa de bits o matriz equivalente.

b) Un factor de estudio o de tratamiento que se está interesado en estudiar es el tamaño de la imagen, por eso el tamaño de las imágenes debe variar y tener al menos 8 niveles. Debe ser muy cuidadoso en escoger los niveles del factor de acuerdo a las condiciones y objetivo del experimento

c) Para ésta fase, se está interesado en medir y estudiar el desempeño en el tiempo de ejecución de los algoritmos.

d) Se debe emplear dos unidades experimentales diferentes (equipos de cómputo)

Debe incluir una sección donde explique en detalle y a la luz de los temas del curso, la relación del comportamiento de la memoria y el tiempo de respuesta que describió los resultados del experimento. Tenga en cuenta entre otros conceptos, los principios de localidad, la jerarquía de

memoria de cada tipo de procesador y otros componentes y técnicas usadas en los computadortes.

Para efectos de evaluación y calificación se aplicará una rúbrica la cual considera los siguientes aspectos que sigue la mayoría de los pasos de la metodología general de diseño de experimentos:

Aspecto a evaluar	**Factor**
Diseñar un experimento al seleccionar el tipo más adecuado, de acuerdo a una hipótesis dada, los recursos disponibles y los factores que deben ser medidos y controlados.	Identificación de los factores primarios, niveles y variable respuesta y definición del procedimiento de recolección de datos.
	Identificación de factores secundarios y de ruido y la manera de controlar sus efectos
	Selección del tipo de experimento de acuerdo con la hipótesis / cuestiones por resolver, los factores y las limitaciones existentes
Llevar a cabo un experimento diseñado siguiendo los procedimientos definidos para adquirir datos sobre las variables apropiadas y reportar y consignar adecuadamente los resultados.	La ejecución del experimento siguiendo el procedimiento.
	Reporte de los resultados
Analizar e interpretar los resultados de un experimento para exponer conclusiones que permitan contestadores hipótesis o preguntas del experimento.	Análisis e interpretación de las observaciones y relacionarlas con la teoría del curso.
	Conclusiones apoyadas por los datos reportados

Grupos de trabajo **2 personas**.
Fecha límite para la sustentación: entre el jueves 10 al sábado 12 de diciembre (los horarios serán concertados por el profesor y se reservará una hora por grupo)

Bibliografía

Adobe. (14 de maayo de 2018). *Acerca de las imágenes de mapa de bits*. Obtenido de https://helpx.adobe.com/es/photoshop/using/image-essentials.html

https://docs.microsoft.com/. (s.f.). *Tipos de mapas de bits*. Obtenido de https://docs.microsoft.com/es-es/dotnet/framework/winforms/advanced/types-of-bitmaps