DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Nº 2

Segundo Cuatrimestre de 2018 — Organización del Computador II

Integrante	LU	Correo electrónico
Nicolás Hertzulis	811/15	nicohertzulis@gmail.com
Cynthia Liberman	443/15	cynthia.lib@gmail.com
María Belén Ticona	143/16	ticona.belu@gmail.com

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

http://www.fcen.uba.ar

Resumen

En el presente trabajo se describirá la implementación de cuatro filtros de imágenes en lenguaje ensamblador x86_64 utilizando el modelo SIMD (del inglés *Single Instruction, Multiple Data*).

Índice

1. Introducción

1.1. Marco teórico

1.2. Objetivos

El objetivo de este Trabajo Práctico es mediante la programación en lenguaje ensamblador utlizando el modelo SIMD, generar distintos filtros a una única imagen de partida en formato BMP. La imágen original, es a color, y cada uno de sus pixeles por lo tanto esta dotado de una graduación específica de los colores B(blue), G(green), R(red) y finalmente A(alpha), que se refiere a la transparencia del pixel. Cada uno de los colores y la transparencia podrán adoptar valores positivos entre 0 y 255. Este rango de valores ocupan un byte en representación binaria y por lo tanto cada pixel tendrá un tamaño de 4 bytes. Los píxeles se almacenan de izquierda a derecha, y cada píxel en memoria se guarda en el siguiente orden: B, G, R, A. Las imágenes se encuentran almacenadas como porciones de pixeles contiguos en la memoria, y se las trata como matrices a nivel esquemático. La modificación de los pixeles originales dada una serie de pautas para cada filtro es lo que dará la imágen resultante. Dada una imagen src (source o fuente), se notara src k[i,j] al valor de la componente k {r, g, b, a} del píxel en la fila i y la columna j de la imagen. La fila 0 corresponde a la fila de más abajo de la imagen. La columna 0 a la de más a la izquierda.Se llamará dst (destiny o destino) a la imagen de salida generada por cada filtro. Los tipos de filtros y procedimientos adoptados se describirán en más detalle en la sección filtros.

1.3. Consideraciones Generales

2. Desarrollo

2.1. Filtro Tres Colores

El filtro *Tres Colores* se caracteteriza por modificar el color de un píxel en función de su brillo, trabajando sobre cada uno de forma independiente del resto de la imagen. Para poder determinar la coloración de cada píxel destino, se preocedió a efectuar una combinación lineal dada por el brillo y un nuevo color entre las siguientes opciones: rojo, verde y crema.

2.1.1. Pre-proceso de imagen

En primer lugar, el algoritmo implementado guarda en memoria las componentes de los nuevos colores así como tres constantes cuyos valores resultan necesarios para determinar qué combinación lineal aplicar en cada píxel. De esta forma, antes de proceder a procesar la imágen se resguardan estas constantes definidas en .rodata en registros xmm, de modo que ya se encuentren disponibles en cada iteración (resultando innecesario buscar sus valores a memoria).

[inserte esquema de registros y lo que tiene cada uno].

En cuanto al procesamiento de la imagen, en cada iteración se trabaja con 4 píxels dado que el tamaño de cada uno es de 4 Bytes -entrando 16 Bytes en el registro XMM1-. La finalización de la ejecución del ciclo se fija mediante una comparación entre la dirección a leer (resguadado en RDI) y la dirección inmediatamente posterior al final de la imagen. Esta última se determina sumando a la dirección fuente de la imagen, el tamaño de las filas (en Bytes) multiplicado por la altura de la imagen en pixels.

2.1.2. Iteración

Por cada ciclo el procesamiento consta de las siguientes partes:

1. Cálculo de Brillo
2. Máscaras según intensidad de Brillo
3. Obtención de Píxels Destino por Combinación Lineal
4. Inserción de Tranparencia

En primer lugar, para iniciar con el *Cálculo de Brillo* se obtienen de los 4 píxels levantados las componentes de cada color por separado, de modo que ocupen un DW cada una. Para ello se emplea un shift lógico empaquedado de DW dado que el mismo completa con ceros las posiciones redefinidas, quedando inalterado el valor de cada componente. De esta forma se obtiene un registro XMM por cada set de bytes rojo, verde y azul de los 4 píxels en proceso. Luego, se procede a sumar de a W dichos registros y a convertir los enteros obtenidos a float para su división por 3. Comos los datos en las imágenes son enteros y dado que para finalizar el cálculo de brillo resta tomar parte entera inferior del resultado obtenido, no resulta necesario una precisión mejor que un *single-precision floating-point*.

Para la obtención de las *Máscaras según intensidad de Brillo* se trabaja por comparación entre las constantes reservadas (véase Pre-proceso de imagen) y los brillos obtenidos mediante las instrucciones pcmpgtd y pcmpeqd. Las mismas setean en 1 en la DW donde se guarda cada brillo en caso de cumplirse la condición de mayor o igual según lo que corresponda en cada caso. En particular cabe resaltar que para el caso $85 < W \leq 170$, se efecúa un OR lógico entre las máscaras de bits de las condiciones $85 \geq W$ y $W > 170$, negando la obtenida y quedando así la máscara de brillos de comprendidos entre 85 y 170 inclusive.

En cuanto a la obtención de los píxels destino, primero se filtró cada color resguardado (rojo, verde y crema) según se cumpliese o no su condición de intensidad de brillo correspondiente. Como la combinación lineal resulta de tomar 3/4 de cada color y sumarle 1/4 del correspondiente brillo, se procedió efectuando la división por 4 al final para evitar pérdida de precisión. Ahora bien como al realizar la multiplicación por 3 de los colores se puede producir *rollover*, se extendió cada componente de los colores de tamaño Byte a Word. Es por ello que resulta necesario desdoblar también el procesamiento de los 4 píxels para realizarlo en partes Low y High (cada una con 2 píxels fuente). Tanto para la multiplicación por tres

de los colores como para su suma con el brillo se trabaja con la intrucción ADDW, habiendo previamente efectuado un *merge* entre las partes Low y High de los colores. Una vez realizada la división por 4 de cada Word mediante un shift lógico a la derecha, se empaquedan ambas partes volviendo a trabajar con 4 píxels en un solo registro.

Resulta importante remarcar que en todo momento la componente correspondiene a la transparencia se encuentra con valor 0. Para que su valor sea 255 se setean todos los bits en 1 mediante una comparación de un registro con sí mismo y se realiza un shift lógico hacia la izquierda para setear en 0 los 3 bytes menos significativos de cada DW. Finalmente se realiza un OR para terminar de obtener los píxels destino, combinando transparencia y colores obtenidos.

2.2. Filtro Efecto Bayer

2.3. Filtro Cambia Color

2.4. Filtro Edge Sobel

3. Resultados

3.1. Filtro Tres Colores

3.2. Filtro Efecto Bayer

3.3. Filtro Cambia Color

3.4. Filtro Edge Sobel

4. Conclusiones

5. Bibliografía