Федеральное государственное автономное образовательное учреждение высшего образования
Университет ИТМО

Кафедра Вычислительной Техники

Дисциплина: Низкоуровневое программирование

Лабораторная работа №8

Выполнил: **Доморацкий Эридан Алексеевич**

Группа: **P33113**

Преподаватель: **Логинов Иван Павлович**

2020г

Задание

16.4.1 Assignment: Sepia Filter

In this assignment, we will create a program to perform a sepia filter on an image. A sepia filter makes an image with vivid colors look like an old, aged photograph. Most graphical editors include a sepia filter.

Выполнение

```
; do_sepia.asm

; vim: syntax=nasm

%define c_1_1 0.393
%define c_1_2 0.769
%define c_1_3 0.189
%define c_2_1 0.349
%define c_2_2 0.686
%define c_2_3 0.168
%define c_3_1 0.272
%define c_3_2 0.543
%define c_3_3 0.131

global do_sepia

section .rodata

align 16
c_r: dd c_1_1, c_2_1, c_3_1, c_1_1, c_2_1, c_3_1, c_1_1, c_2_1, c_3_1, c_1_1, c_2_1, c_3_1
c_g: dd c_1_2, c_2_2, c_3_2, c_1_2, c_2_2, c_3_2, c_1_2, c_2_2, c_3_2, c_1_2, c_2_2, c_3_2
c_b: dd c_1_3, c_2_3, c_3_3, c_1_3, c_2_3, c_3_3, c_1_3, c_2_3, c_3_3, c_1_3, c_2_3, c_3_3

section .text

; process 4 pixels with sepia filter via SSE
; rdi - red    12-dim vector
; rsi - green  12-dim vector
; rdx - blue   12-dim vector
; rcx - result 12-dim vector
do_sepia:
   lea r9, [rel c_r]
   lea r10, [rel c_g]
   lea r11, [rel c_b]

   mov r8, 3
.loop:
   cvtdq2ps xmm0, [rdi]
   cvtdq2ps xmm1, [rsi]
   cvtdq2ps xmm2, [rdx]

   mulps xmm0, [r9]
   mulps xmm1, [r10]
   mulps xmm2, [r11]

   addps xmm0, xmm1
   addps xmm0, xmm2
   cvtps2dq xmm0, xmm0
   movdqa [rcx], xmm0

   add rdi, 16
   add rsi, 16
   add rdx, 16
   add rcx, 16
   add r9, 16
   add r10, 16
   add r11, 16

   dec r8
   test r8, r8
   jnz .loop

   ret
```

```c
// sepia.c

#include <stdio.h>

#include <value.h>
#include <image.h>

void do_sepia(const uint32_t r[12], const uint32_t g[12], const uint32_t b[12], uint32_t
result[12]);

static uint8_t sat(uint32_t a) {
   return a > 255 ? 255 : a;
}

static void process_cluster(struct pixel * cluster, uint8_t size) {
   static uint32_t r[12], g[12], b[12], result[12];
   uint8_t i, j;

   for (i = 0; i < size; ++i) {
       for (j = 0; j < 3; ++j) {
           r[i * 3 + j] = cluster[i].red;
           g[i * 3 + j] = cluster[i].green;
           b[i * 3 + j] = cluster[i].blue;
       }
   }

   do_sepia(r, g, b, result);
   for (i = 0, j = 0; i < size; ++i) {
       cluster[i].red   = sat(result[j++]);
       cluster[i].green = sat(result[j++]);
       cluster[i].blue  = sat(result[j++]);
   }
}

const char * sepia(struct image * image, uint32_t argc, const struct value * argv) {
   const uint32_t length = image->width * image->height;
   const uint32_t clusters = length / 4;
   uint32_t i;

   for (i = 0; i < clusters; ++i) {
       process_cluster(image->pixels + i * 4, 4);
   }

   if (length % 4 > 0) {
       process_cluster(image->pixels + i * 4, length % 4);
   }

   return NULL;
}


static void sepia_naive_one(struct pixel * const pixel) {
   static const float c[3][3] = {
       { .393f, .769f, .189f },
       { .349f, .686f, .168f },
       { .272f, .543f, .131f }
   };

   struct pixel const old = * pixel;

   pixel->red   = sat(old.red * c[0][0] + old.green * c[0][1] + old.blue * c[0][2]);
   pixel->green = sat(old.red * c[1][0] + old.green * c[1][1] + old.blue * c[1][2]);
   pixel->blue  = sat(old.red * c[2][0] + old.green * c[2][1] + old.blue * c[2][2]);
}

const char * sepia_naive(struct image * image, uint32_t argc, const struct value * argv) {
   const uint32_t length = image->width * image->height;
   uint32_t i;

   for (i = 0; i < length; i++) {
       sepia_naive_one(image->pixels + i);
   }

   return NULL;
}
```

Вывод

В результате выполнения лабораторной работы был реализован фильтр для изображений сепия с использованием SSE инструкций процессора для параллельной обработки чисел.