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

```
; 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

extern memcpy:function

global sepia:function

section .text

; process 4 pixels with sepia filter via SSE
; rdi - rgb    three 12-dim vector of dwords
; rsi - result 12-dim vector of bytes
do_sepia:
   ; populate registers
   cvtdq2ps xmm0, [rdi]
   cvtdq2ps xmm1, [rdi + 16]
   cvtdq2ps xmm2, [rdi + 32]
   cvtdq2ps xmm3, [rdi + 48]
   cvtdq2ps xmm4, [rdi + 64]
   cvtdq2ps xmm5, [rdi + 80]
   cvtdq2ps xmm6, [rdi + 96]
   cvtdq2ps xmm7, [rdi + 112]
   cvtdq2ps xmm8, [rdi + 128]

   ; multiply
   lea rdi, [rel .coef]
   mulps xmm0, [rdi]
   mulps xmm1, [rdi + 16]
   mulps xmm2, [rdi + 32]
   mulps xmm3, [rdi + 48]
   mulps xmm4, [rdi + 64]
   mulps xmm5, [rdi + 80]
   mulps xmm6, [rdi + 96]
   mulps xmm7, [rdi + 112]
   mulps xmm8, [rdi + 128]

   ; summarize
   addps xmm0, xmm3
   addps xmm0, xmm6
   addps xmm1, xmm4
   addps xmm1, xmm7
   addps xmm2, xmm5
   addps xmm2, xmm8

   ; convert
   cvtps2dq xmm0, xmm0
   cvtps2dq xmm1, xmm1
   cvtps2dq xmm2, xmm2
   xorps xmm3, xmm3

   packusdw xmm0, xmm1
   packusdw xmm2, xmm3

   packuswb xmm0, xmm2
```

```
   push rbp
   mov rbp, rsp

   sub rsp, 16
   and rsp, 0xfffffffffffffff0
   movdqa [rsp], xmm0

   pop qword [rsi]

   mov eax, [rsp]
   mov [rsi + 8], eax
   mov rsp, rbp
   pop rbp
   ret

section .rodata
align 16

.coef:
   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
   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
   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 cluster of four or less pixels
; rdi - cluster, pointer to array of pixels
; rsi - positive size of cluster
process_cluster:
   lea r8, [rel .r]
   lea r9, [rel .g]
   lea r10, [rel .b]
   push rdi
   push rsi

   mov rcx, rsi
.populate_loop:
   movzx eax, byte [rdi]
   mov [r8], eax
   mov [r8 + 4], eax
   mov [r8 + 8], eax
   add r8, 12
   inc rdi

   movzx eax, byte [rdi]
   mov [r9], eax
   mov [r9 + 4], eax
   mov [r9 + 8], eax
   add r9, 12
   inc rdi

   movzx eax, byte [rdi]
   mov [r10], eax
   mov [r10 + 4], eax
   mov [r10 + 8], eax
   add r10, 12
   inc rdi

   loop .populate_loop

   lea rdi, [rel .r]
   lea rsi, [rel .result]
   call do_sepia

   mov rdx, [rsp]
   shl rdx, 1
   add rdx, [rsp]
   add rsp, 8

   pop rdi
   lea rsi, [rel .result]
   call memcpy wrt ..plt
   ret
```

```
section .bss
align 16

.r:
   resd 12
.g:
   resd 12
.b:
   resd 12
.result:
   resb 12

section .text

; sepia function
; rdi - pointer to image
; rsi - count of args
; rdx - pointer to transformation args array
sepia:
   xor rax, rax
   mov eax, [rdi]      ; image->width
   xor rdx, rdx
   mov edx, [rdi + 4]  ; image->height
   mul rdx             ; image->width * image->height
   jz .end

   push rax    ; length = image->width * image->height

   shr rax, 2      ; image->width * image->height / 4
   mov rcx, rax    ; image->width * image->height / 4

   mov rax, [rdi + 8]  ; image->pixels
   push rax            ; current_cluster = image->pixels

.loop:
   push rcx            ; clusters

   mov rax, [rsp + 8]  ; current_cluster
   mov rdi, rax        ; current_cluster

   add rax, 4 * 3      ; current_cluster + 4
   mov [rsp + 8], rax  ; current_cluster += 4

   mov rsi, 4
   call process_cluster    ; process_cluster(current_cluster - 4, 4)

   pop rcx     ; clusters
   loop .loop  ; while (--clusters > 0)

   ; if .length % 4 != 0 then process last cluster
   mov rsi, [rsp + 8]  ; length
   and rsi, 3          ; length % 4

   jz .end                 ; if (length % 4 != 0)
   mov rdi, [rsp]
   call process_cluster    ; process_cluster(current_cluster, length % 4)

.end:
   pop rax
   pop rax
   xor rax, rax    ; return NULL
   ret
```

Вывод

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