#include <stdio.h>

#include <stdlib.h>

#include <emmintrin.h>

#include <opencv/cv.h>

#include <opencv/highgui.h>

int main(int argc, char\*\* argv) {

IplImage\* Img1 = cvLoadImage(argv[1], CV\_LOAD\_IMAGE\_UNCHANGED);

IplImage\* ImgA = cvCreateImage(cvSize(Img1->width, Img1->height), 8, 4);

// a visualization window is created with title 'image'

cvNamedWindow("original", CV\_WINDOW\_AUTOSIZE);

cvShowImage("original", Img1);

cvWaitKey(0);

cvNamedWindow("alfa", CV\_WINDOW\_AUTOSIZE);

\_\_m128i mask\_alfa = \_mm\_set1\_epi32(0xFF000000);

\_\_m128i valor\_Canal\_alfa;

\_\_m128i R, G, B, aux;

for (int fila = 0; fila < Img1->height; fila++) {

\_\_m128i \*pOriginal = (\_\_m128i \*) (Img1->imageData + fila \* Img1->widthStep);

\_\_m128i \*pAlfa = (\_\_m128i \*) (ImgA->imageData + fila \* ImgA->widthStep);

for (int columna = 0; columna < Img1->widthStep; columna = columna + 16) {

valor\_Canal\_alfa = \_mm\_and\_si128(\*pOriginal, mask\_alfa);

R = \_mm\_srli\_epi32(valor\_Canal\_alfa, 8);

G = \_mm\_srli\_epi32(valor\_Canal\_alfa, 16);

B = \_mm\_srli\_epi32(valor\_Canal\_alfa, 24);

aux = \_mm\_or\_si128(R, G);

aux = \_mm\_or\_si128(aux, B);

\*pAlfa = aux;

pOriginal++;

pAlfa++;

}

}

cvShowImage("alfa", ImgA);

cvWaitKey(0);

// memory release for img before exiting the application (funcion liberar memoria)

cvReleaseImage(&Img1);

// Self-explanatory (funcion distruir ventana)

cvDestroyWindow("original");

return EXIT\_SUCCESS;

}