Министерство науки и высшего образования РФ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет автоматики и вычислительной техники

Кафедра радиоэлектронных средств

Отчёт по лабораторной работе №2

**Обработка изображений**

дисциплина «Проектирование устройств и систем на цифровых сигнальных процессорах»

|  |  |  |
| --- | --- | --- |
| Выполнил студент гр.ИНБс-5301: |  | /Шикалов Д. С./ |
| Проверил: старший преподаватель кафедры систем автоматизации управления |  | /Земцов М. А./ |

Киров 2023

**Цель:** изучение дискретного косинусного преобразования изображения; получение навыков обработки изображений на DSK.

**Ход работы**

Программный код приведён в <https://github.com/Argento-prg/SignalProcessors>/tree/lab2/code

**1 Формирование файла-пробника**

Исходное изображение по варианту (вариант №1) представлено на рисунке 1.

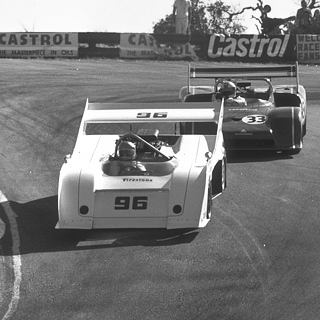


Рисунок 1 – Исходное изображение

Входные данные представлены в виде одномерного массива, где каждый элемент является значением яркости пикселя черно-белого изображения (8 бит), где 0 – черный цвет, 1 – белый цвет. Входные данные представлены на рисунке 2.

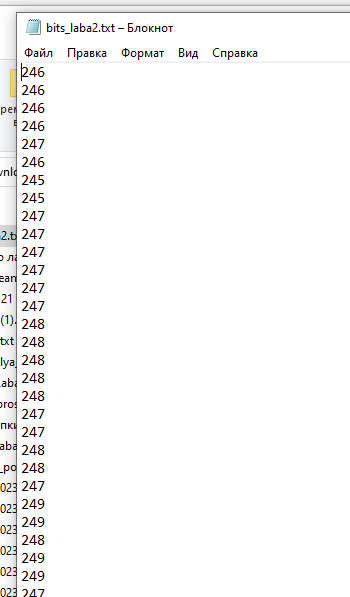


Рисунок 2 – Входные данные

**2 Обработка данных**

Для загрузки файла из которого данные изображения будут читаться в память, требуется выставить Probe Point в месте, до обработки данных. Места расположения Brake Point и Probe Point представлены на рисунке 3.

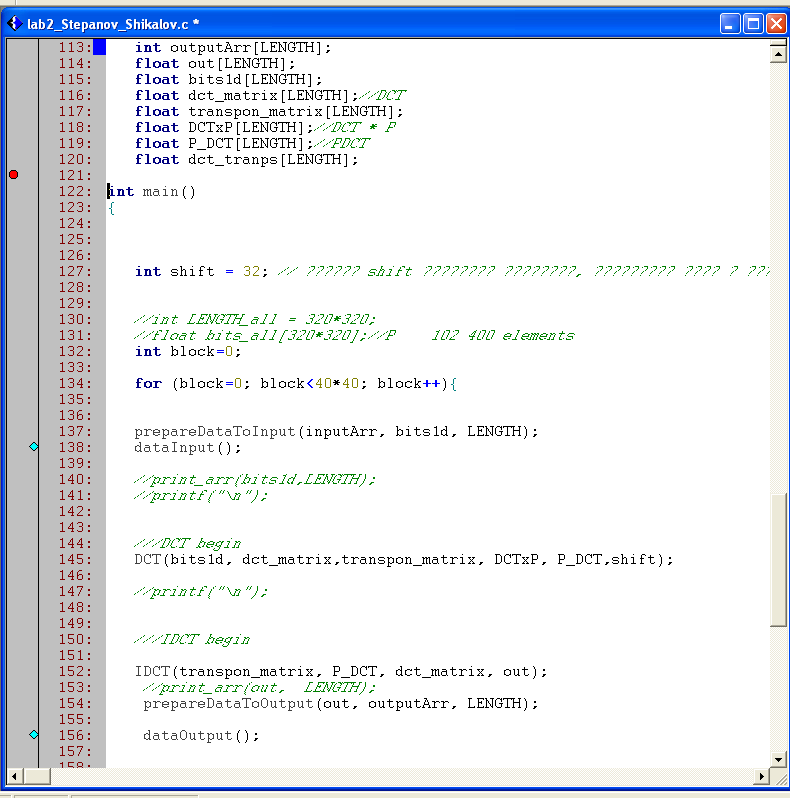


Рисунок 3 – Места расположения Brake Point и Probe Point

Brake Point в 121 строке служит для паузы в выполнении программы, в ходе которой можно будет установить адреса входного inputArr и выходного outputArr массивов.

После установки точек, показанных на рисунке 3, требуется обеспечить доступ к точкам Brake Point и Probe Point. Для этого требуется пересобрать проект (Rebuild All). После этого требуется зайти в пункт меню File I/O. Пункт меню File I/O представлено на рисунке 4.

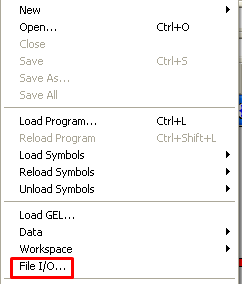


Рисунок 4 – Пункт меню File IO

Настройки для подключения входного файла представлены на рисунке 5.

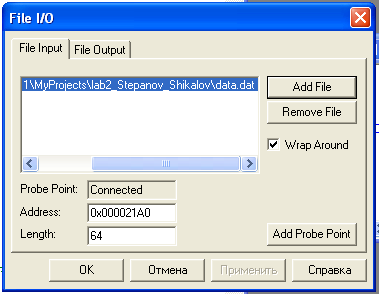


Рисунок 5 – Настройки для входного файла

Настройки для подключения выходного файла представлены на рисунке 6.

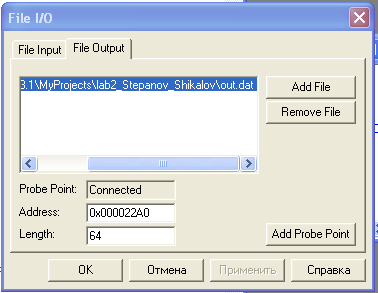


Рисунок 6 – Настройки для подключения выходного файла

Настройки для подключения Probe point представлены на рисунке 7.

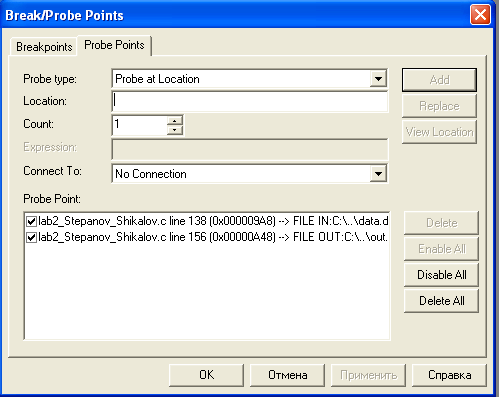


Рисунок 7 – Настройки для подключения Probe point

После данных действий, можно продолжать выполнение программы с точки останова, о которой говорилось ранее. Функции dataInput и dataOutput служат заглушками, на которых ставятся Probe Point.

**3 Результаты работы**

Результат работы программы представлен на рисунке

В результате получились следующие выходные данные (рисунок 9).

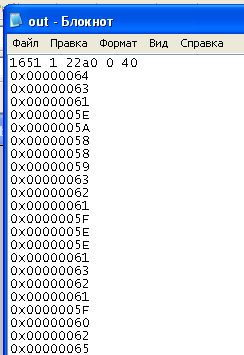


Рисунок 9 – Пример выходных данных

Полученные выходные данные преобразуются в десятичный вид, затем преобразуются в значения яркости пикселей, после чего из пикселей формируется изображение.

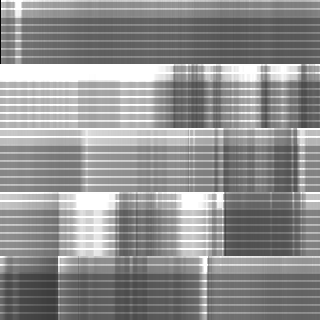


Рисунок 10 – Сжатие до 10 значащих элементов из 64



Рисунок 11 – Сжатие до 32 значащих элементов из 64

**4. Код программ**

**4.1 Код преобразования изображения**

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <stdint.h>

#define PI 3.141592653589793

#define LENGTH 64

void dataInput() {

return;

}

void dataOutput() {

return;

}

void prepareDataToInput(int\* in, float\* out, int LENGTHBuffer) {

int i;

for(i = 0; i < LENGTHBuffer; i++) {

out[i] = 0 + in[i];

}

}

void prepareDataToOutput(float\* in, int\* out, int LENGTHBuffer) {

int i;

for(i = 0; i < LENGTHBuffer; i++) {

out[i] = (int)in[i];

}

}

void transpon(float \*arr, float\* transpon\_matrix)

{

int m = LENGTH/8;

int n = m;

int i = 0;

int j = 0;

for (i = 0; i < m; ++i )

{

for (j = 0; j < n; ++j )

{

// Index in the original matrix.

int index1 = i\*n+j;

// Index in the transpose matrix.

int index2 = j\*m+i;

transpon\_matrix[index2] = arr[index1];

}

}

}

void multiply(float \*arr1, float \*arr2, float \*after\_multyply) {

int i, j, k;

int tempLENGTH = LENGTH/8;

for (i = 0; i < tempLENGTH; i++) {

for (j = 0; j < tempLENGTH; j++) {

after\_multyply[i \* tempLENGTH + j] = 0;

for (k = 0; k < tempLENGTH; k++) {

after\_multyply[i \* tempLENGTH + j] += arr1[i \* tempLENGTH + k] \* arr2[k \* tempLENGTH + j];

}

}

}

}

void dct(float \*arr){

int row;

int col;

int tempLENGTH = LENGTH/8;

for(row = 0; row < tempLENGTH; row++) {

for(col = 0; col < tempLENGTH; col++) {

if(row == 0) {

arr[row \* tempLENGTH + col] = 1 / sqrt(tempLENGTH);

arr[row \* tempLENGTH + col] \*= cos((2 \* col + 1) \* row \* PI / (2 \* tempLENGTH));

} else {

arr[row \* tempLENGTH + col] = sqrt(2.0 / tempLENGTH);

arr[row \* tempLENGTH + col] \*= cos((2 \* col + 1) \* row \* PI / (2 \* tempLENGTH));

}

}

}

}

void shakal(float\* arr, int shift){

int i = shift;

for (i = shift; i<LENGTH; i++){

arr[i]=0;

}

}

void IDCT(float \*transpon\_matrix, float \*P\_DCT, float \*dct\_matrix, float \* out){

float temp[64];

multiply(transpon\_matrix,P\_DCT, temp);// DCT^T \* P\_DCT

multiply(temp, dct\_matrix,out);

}

void DCT(float\* bits, float\* dct\_matrix,float\* transpon\_matrix, float\* DCTxP, float\* P\_DCT, int shift){

dct(dct\_matrix);//DCT

shakal(dct\_matrix, shift);//DCT shakal

multiply(dct\_matrix, bits,DCTxP);//DCT \* P

transpon(dct\_matrix,transpon\_matrix);//DCT ^ T

multiply(DCTxP,transpon\_matrix, P\_DCT);// DCT ^ T \* DCT \* P

}

int inputArr[LENGTH];

int outputArr[LENGTH];

float out[LENGTH];

float bits1d[LENGTH];

float dct\_matrix[LENGTH];//DCT

float transpon\_matrix[LENGTH];

float DCTxP[LENGTH];//DCT \* P

float P\_DCT[LENGTH];//PDCT

float dct\_tranps[LENGTH];

int main()

{

int shift = 10; // ?????? shift ???????? ????????, ????????? ???? ? ??????? dct

//int LENGTH\_all = 320\*320;

//float bits\_all[320\*320];//P 102 400 elements

int block=0;

for (block=0; block<40\*40; block++){

prepareDataToInput(inputArr, bits1d, LENGTH);

dataInput();

//print\_arr(bits1d,LENGTH);

//printf("\n");

///DCT begin

DCT(bits1d, dct\_matrix,transpon\_matrix, DCTxP, P\_DCT,shift);

//printf("\n");

///IDCT begin

IDCT(transpon\_matrix, P\_DCT, dct\_matrix, out);

//print\_arr(out, LENGTH);

prepareDataToOutput(out, outputArr, LENGTH);

dataOutput();

}

return 0;

}

**4.2 Код преобразования выходного файла со значениями пикселей итогового изображения**

import numpy as np

from PIL import Image

kek=[]

count=0

with open ('out.dat','r') as outdat:

for i in outdat.readlines():

if(count==0):

count+=1

continue

else:

kek.append(int(i,16))

image\_arr1 = np.zeros((320,320))

count = 0

for i in range(len(image\_arr1)):

for j in range(len(image\_arr1[0])):

image\_arr1[i][j] = kek[count]

count+=1

image\_arr1 = np.transpose(image\_arr1)

new\_image = Image.fromarray(image\_arr1)

new\_image = new\_image.convert('RGB')

new\_image.save('shift.png')

**Выводы:**

Был изучен и применён на практике алгоритм дискретного косинусного преобразования, а также, получены и применены на практике навыки обработки изображений на DSK.

* алгоритм дискретного косинусного преобразования основан на занулении ненужных частотных составляющих спектра изображения;
* чем меньше важных частотных составляющих спектра остается в итоговом варианте, тем менее разборчивым к идентификации получается изображение;
* алгоритм дискретного косинусного преобразования лежит в основе алгоритма сжатия в изображениях формата JPEG.