

**Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
Национальный исследовательский университет «МЭИ»**

Расчетное задание

Вариант 4

На тему: “Преобразование файла из формата РСХ в формат BMP.”

Курс: “Системное программное обеспечение”

Выполнил

Студент: Балашов Савва Арсеньевич

Группа: А-08-19

Дата: 20.05.2022

Проверил

Преподаватель: Челышев Эдуард Артурович

Оценка: _____

Дата: _____

Москва, 2022

Содержание

Введение.....	3
Анализ задания.....	4
Проектирование пользовательского интерфейса.....	5
Представления данных.....	6
Формат файла РСХ.....	10
Файл формата ВМР	14
Прерывания	17
Алгоритм работы программы	19
Описание процедур и макросов	20
Тестирование и отладка.....	21
Анализ возможностей разработанной программы.....	24
Заключение	24
Список литературы	25
Приложение А. Листинг программы	26

Введение

Данное задание в общем случае актуально, поскольку очень часто требуется преобразовать файл из одного формата в другой, однако конкретный пример с форматированием .rsx файла в .bmp файл может быть немного устаревшим, поскольку расширение .rsx уже практически не используется. Однако, благодаря тому, что оно было одним из первых широко принятых форматов изображений, его структуру приняли за основу другие более современные расширения, такие как .bmp, .jpg, .png.

Перевод из одного формата в другой требует анализа заголовка файла и прочтения тела файла с последующей перезаписью его в другом виде.

Для такой задачи потребуются навыки:

- Знание регистровой модели процессора
- Умение работать с типами данных
- Умение работать с несколькими частями сегмента
- Умение работать с файлами, а именно, загружать и сохранять
- Умение работать с процедурами
- Умение работать с прерываниями процессора

Анализ задания

Техническое задание: Преобразование файла из формата РСХ в формат BMP (256-цветов на точку).

Конкретизация: Загрузка из корневой директории файла формата .rsx, преобразование его в формат .bmp и сохранение полученного файла в корневую директорию.

Подзадачи: начальная инициализация сегментных регистров, создание нескольких сегментов данных для отдельного хранения rsx и bmp файлов, вывод сообщений, процедуры очистки экрана, вывода меню, сохранения названия файла, закрытия файла, сравнения цвета с палитрой, сохранения цвета в палитру. Основная процедура: проверки ошибок и соответствия формату, расшифровка раstra, обработка раstra, сохранение в bmp, вывод меню.

Проектирование пользовательского интерфейса

Вывод сообщений будет осуществляться при помощи прерывания int 21h (ah = 9, dx = <сообщение>) (не имеет подходящей альтернативы). В начале работы очистится экран с помощью прерывания int 10h (ah = 00h, al = 3) (можно использовать функцию прокручивания видимой части вниз, в данном случае равноценно). Затем выведется сообщение с просьбой ввести название .rsx файла. После ввода аналогичное сообщение для выбора названия .bmp файла. После открытия файла, будет производиться конвертация в несколько шагов. Сначала, будет скопирован заголовок входного файла и переформатирован в заголовок выходного файла путем присвоения нужных данных. Далее, в буфер будет помещен зашифрованный изначально растр изображения. Его нужно будет расшифровать. Расшифровка будет проходить путем поиска эталонных байтов. Затем следует записать строки изображения в растр выходного файла в обратном порядке, поскольку BMP хранит информацию таким образом. Последним шагом сохраним файл. После успешного завершения работы, выведется сообщение об успехе и появится меню для повторной конвертации или выхода из программы.

Представления данных

Буферы для хранения имён файлов (байт максимальная длина 255, место для помещения строки)

PCXName db 255,255 dup (?)

BMPName db 255,255 dup (?)

Строки сообщений, которые будут выводиться на экран (байт, 0ah и 0dh - перенос строки, "<сообщение>", перенос строки, \$ окончание строки)

MenuNewMsg db 0ah,0dh, '*****Press R to choose a new file*****',0ah,0dh,'\$'

MenuExitMsg db 0ah,0dh, '*****Press E to exit*****',0ah,0dh,'\$'

Prompt1 db 'PCX File: \$'

Prompt2 db 0ah,0dh,'BMP File: \$'

OFE db 0ah,0dh,'Open File Error ',0ah,0dh,'\$'

NotPCX db 0ah,0dh,'File Is Not PCX-Image',0ah,0dh,'\$'

Not256 db 0ah,0dh,'File Is Not 256-colors PCX',0ah,0dh,'\$'

RE db 0ah,0dh,'Read error',0ah,0dh,'\$'

CNCFMsg db 0ah,0dh,'Could Not Create File',0ah,0dh,'\$'

DoneMsg db 0ah, 0dh, 'File converted successfully', 0ah, 0dh, '\$'

Заголовки файлов (Размеры байт, слово, двойное слово. Заполняются динамически)

Manufacturer db ?

Version db ?

Encoding db ?

BitsPerPixel db ?

WindowXmin dw ?

WindowYmin dw ?

WindowXmax dw ?
WindowYmax dw ?
Hres dw ?
Vres dw ?
Colormap db 48 dup (?)
Reserved db ?
NPlanes db ?
BytesPerLine dw ?
PaletteInfo dw ?
Filler db 58 dup(?)
PCXHeaderSize equ \$-Manufacturer

bfType db 'BM'
bfSize dd ?
bfReserved1 dw 0
bfReserved2 dw 0
bfOffBits dd ?
BMPHeaderSize equ \$-bfType

biSize dd BMPInfoHeaderSize
biWidth dd ?
biHeight dd ?
biPlanes dw 1
biBitCount dw 8
biCompression dd 0
biSizeImage dd ?
biXPelsPerMeter dd ?
biYPelsPerMeter dd ?
biClrUsed dd 256

biClrImportant dd ?
BMPInfoHeaderSize equ \$-biSize

Палитра BMP
BMPPallete db 1024 dup(0)
BMPPalleteSize equ 1024
ColorsCount dw 0

Байты для выравнивания РСХ
AdditiveBytes db 0

Дескрипторы
PCXDescr dw ?
BMPDescr dw ?

Количество байт в запакованном растре РСХ и номер байта в растра
BMP
ArraySize dw ?
CurByte dw 0

Переменные для хранения текущих значений составляющих R, G и
B
CurR db ?
CurG db ?
CurB db ?

Буфер для запакованного растра РСХ
PackedArray db 60000 dup (?)

Дополнительные сегменты данных для хранения растров РСХ и ВМР (para - от параграф (16 байт), public - этот сегмент будет объединен с другими с той же областью в один сегмент, 'класс' сегмента)

MemoryBuf segment para public 'data'

db 0FFFFh dup (?)

MemoryBuf ends

Bitmap segment para public 'data'

db 0FFFFh dup (0)

Bitmap ends

Формат файла PCX

Файл PCX - это растровое изображение в формате программы PC Paintbrush, который был разработан корпорацией ZSoft в 1985 году. Изображения PCX могут быть представлены в следующих цветовых режимах: черно-белый, 16 цветов, оттенки серого (8 бит), на основе палитры (8 бит) или RGB (24 бита). В настоящее время формат PCX используют отдельные приложения для факса и сканирования, а также поддерживают некоторые графические редакторы, среди которых CorelDRAW, Adobe Photoshop, GIMP.

Используется сжатие без потерь. При сохранении изображения подряд идущие пиксели одинакового цвета объединяются и вместо указания цвета для каждого пикселя указывается цвет группы пикселей и их количество.

Файлы изображения PCX начинаются с заголовка длиной 128 байт (табл. 1). Затем идут закодированные графические данные. При кодировании используется простой алгоритм, основанный на методе длинных серий. Если в файле запоминается несколько цветовых слоев, каждая строка изображения запоминается по цветовым слоям.

Таблица 1.

Заголовок файла PCX

Смещение	Обозначение	Длина	Описание / комментарий
0	Manufacturer	1	Постоянный флаг 10 = ZSoft .PCX
1	Version	1	Информация о версии: 0 = Версия 2.5 2 = Версия 2.8 с информацией о палитре 3 = Версия 2.8 без информации о палитре 5 = Версия 3.0
2	Encoding	1	1 = PCX кодирование длинными сериями
3	Bits per pixel	1	Число бит на пиксел в слое

4	Window	8	Размеры изображения (Xmin, Ymin) – (Xmax, Ymax) в пикселах включительно
12	Hres	2	Горизонтальное разрешение создающего устройства
14	Vres	2	Вертикальное разрешение создающего устройства
16	Colormap	48	Набор цветовой палитры (см. далее текст)
64	Reserved	1	
65	NPlanes	1	Число цветовых слоев
66	Bytes per Line	2	Число байт на строку в цветовом слое (для PCX-файлов всегда должно быть четным)
68	Palette Info	2	Как интерпретировать палитру: 1 = цветная/черно-белая, 2 = градации серого
70	Filler	58	Заполняется нулями до конца заголовка

Метод кодирования состоит в следующем:

Для каждого байта X, прочитанного из файла. Если оба старших бита X равны 1, то <повторитель> = значению, хранящемуся в 6 младших битах X <данные> = находятся в следующем байте за X. Иначе <повторитель> = 1 <данные> = X

Поскольку для насыщения данного алгоритма требуется в среднем 25% неповторяющихся данных и, по меньшей мере, наличие смещения между повторяющимися данными, то размер получаемого файла, как правило, оказывается приемлемым.

Декодирование файлов в формате PCX

Сначала определяется размер изображения, для этого вычисляют $[XSIZE = Xmax - Xmin + 1]$ и $[YSIZE = Ymax - Ymin + 1]$.

Затем вычисляют, сколько байтов требуется для сохранения одной несжатой строки развертки изображения:

$TotalBytes = NPlanes * BytesPerLine$

Т.к. всегда используется целое число байтов, возможно существование неиспользуемых данных в конце каждой строки развертки. TotalBytes показывает, сколько памяти должно быть доступно для

декодирования каждой строки развертки, включая неиспользуемую информацию на правом конце каждой строки.

Далее выполняется собственно декодирование, читается первый байт данных из файла. Если два старших бита этого байта равны 1, оставшиеся шесть битов показывают, сколько раз следует повторить следующий байт из файла. Если это не так, то этот байт сам является данными с повторителем равным 1. Далее продолжается декодирование до конца строки, ведя подсчет количества байтов, переданных в буфер вывода. В конце каждой строки развёртки имеет место остановка алгоритма кодирования, но ее не существует при переходе от одного слоя к другому. Когда строка сформирована полностью, в конце каждого слоя внутри строки возможно наличие лишних данных. Для нахождения этого остатка используются значения XSIZE и YSIZE. Если данные являются многослойными, то BytesPerLine показывает, где заканчивается каждый слой внутри строки развёртки.

Продолжается декодирование оставшихся строк. В файле возможно наличие лишних строк с округлением на 8 или 16 строк.

Для доступа к 256-цветной палитре следует прочитать в заголовке поле Version. Если оно равно 5, палитра должна быть. Или прочитать в заголовке поле Bits per pixel. Если оно равно 8, 256-цветная палитра должна быть.

В пакете программ PCX Programmer's Toolkit фирмы Genus Microprogramming принят другой способ хранения 256-цветной палитры.

Дескриптор изображения определяет действительное расположение и размеры последующего изображения внутри пространства, определенного в дескрипторе экрана. Также определяются флаги, указывающие на присутствие локальной таблицы для поиска цветов и определения последовательности высвечивания пикселей. Каждый дескриптор изображения начинается с символа-разделителя изображений.

Роль разделителя изображений состоит просто в синхронизации при входе в дескриптор изображения.

Файл формата BMP

BMP - стандартный формат графических файлов Windows. BMP-файлы состоят из трех основных частей:

- заголовок;
- палитра;
- графические данные (значения пикселей).

Заголовок содержит информацию о файле и находящемся в нем графическом изображении. Здесь хранятся параметры изображения (ширина, высота, глубина пикселей), а также количество цветов в нем.

Палитра присутствует только в BMP-файлах, содержащих палитровые изображения (с глубиной пикселей 8 бит и менее). К 8-битным изображениям прикладывается палитра, состоящая из не более чем 256 элементов.

Графические данные - это и есть само изображение. Формат этих данных зависит от глубины пикселей.

Данные заголовка BMP-файла хранятся в двух структурах: `BITMAPFILEHEADER` и `BITMAPINFOHEADER`. Структура `BITMAPFILEHEADER` присутствует в начале любого BMP-файла и содержит информацию о самом файле. Для нас в этой структуре представляет интерес лишь одно поле — `bfType`, сигнатура BMP-файла. В BMP-файлах это поле содержит буквы `BM`. По содержимому этого поля мы будем убеждаться в том, что выбранные файлы действительно имеют формат BMP.

В таблице 2 представлена структура файла BMP.

Таблица 2.

Структура файла BMP

Имя	Длина	Смещение	Описание
Заголовок файла (BitMapFileHeader)			
Type	2	0	Сигнатура "BM"
Size	4	2	Размер файла
Reserved 1	2	6	Зарезервировано
Reserved 2	2	8	Зарезервировано
OffsetBits	4	10	Смещение изображения от начала файла
Информационный заголовок (BitMapInfoHeader)			
Size	4	14	Длина заголовка
Width	4	18	Ширина изображения, точки
Height	4	22	Высота изображения, точки
Planes	2	26	Число плоскостей
BitCount	2	28	Глубина цвета, бит на точку
Compression	4	30	Тип компрессии (0 - несжатое изображение)
SizeImage	4	34	Размер изображения, байт
XpelsPerMeter	4	38	Горизонтальное разрешение, точки на метр
YpelsPerMeter	4	42	Вертикальное разрешение, точки на метр
ColorsUsed	4	46	Число используемых цветов (0 - максимально возможное для данной глубины цвета)
ColorsImportant	4	50	Число основных цветов
Таблица цветов (палитра) (ColorTable)			
ColorTable	1024	54	256 элементов по 4 байта
Данные изображения (BitMap Array)			
Image	Size	1078	Изображение, записанное по строкам слева направо и снизу вверх

Первое поле, biSize, определяет размер структуры BITMAPINFOHEADER в байтах.

Поля biWidth, biHeight и biBitCount определяют размеры изображения. Содержимое поля biCompression позволяет узнать, хранится ли изображение в сжатом виде.

В поле biSizeImage хранится размер графических данных (в пикселях). Однако учтите, что это поле часто оказывается незаполненным (содержит нулевое значение).

Наконец, поле biClrUsed определяют количество цветов в палитре (для палитровых изображений). Как и поле biSizeImage, оно часто может быть равно нулю. Это означает, что палитра содержит максимальное количество элементов (256 для 8-битных изображений).

Палитра в BMP-файлах хранится в виде списка структур RGBQUAD, где каждый элемент представляет отдельный цвет.

Графические данные в основном представляют собой список пикселей, из которых состоит изображение. Однако каждая горизонтальная строка пикселей должна занимать блок памяти, выровненный по границе параграфа. Следовательно, если количество байт, необходимых для хранения строки пикселей, не кратно четырем, в каждую строку включается от одного до трех дополняющих байт.

Изображения хранятся в BMP-файлах в перевернутом виде, так что первая строка пикселей файла на самом деле является нижней строкой настоящего изображения. Чтобы восстановить нормальное изображение, нужно начать чтение файла с последней строки пикселей и двигаться к началу.

Прерывания

Используются прерывания DOS int 21h, int16h и int 10h

Int 21:

- AH = 9 - вывод строки на экран. Вход: AH = 9, DS:DX = <адрес строки>. Возвращаемые значения отсутствуют.
- AH = 3dh - открыть файл с условием. Вход: AL = условие: 0 - чтение, 1 - запись, 2 - чтение/запись. DS:DX = <адрес имени файла>. Возвращаемые значения: CF = 1 при ошибке, AX = код ошибки; CF = 0 при успехе, AX = дескриптор файла
- AH = 3fh - прочитать из файла с условием. Вход: BX = условие, CX = количество байтов для чтения, DS:DX = <адрес буфера>. Возвращаемые значения: CF = 1 при ошибке, AX = код ошибки; CF = 0 при успехе, AX = количество прочитанных байтов
- AH = 3ch - создать файл с условием. Вход: CX = условие, DS:DX = <адрес имени файла>. Возвращаемые значения: CF = 1 при ошибке, AX = код ошибки; CF = 0 при успехе, AX = дескриптор файла
- AH = 40h - запись в файл с условием. Вход: BX = условие, CX = количество байтов для записи, DS:DX = <адрес буфера>. Возвращаемые значения: CF = 1 при ошибке, AX = код ошибки; CF = 0 при успехе, AX = количество записанных байтов
- AH = 42h - перенос указателя чтения/записи в файл. Вход: AL = условие сдвига (0 - начало файла, 1 - текущая позиция, 2 - конец файла), BX = условие (чтение/запись), CX:DX = смещение в байтах. Возвращаемые значения: CF = 1 при ошибке, AX = код ошибки; CF = 0 при успехе, DX:AX = новое смещение
- AH = 4ch - выход из программы. Вход: AL = 00. Возвращаемые значения отсутствуют.
- AH = 0ah
- AH = 3eh

Int 16h:

- AH = 0 - считывание символа с клавиатуры. Возвращаемые значения: AH = сканкод, AL = символ.

Int 10h:

- AH = 0 - задание режима отображения. Вход: AL = 03 - режим отображения.

Алгоритм работы программы

1. Инициализация сегментов
2. Очистка экрана, вывод сообщений с просьбой ввести названия файлов.
3. Открытие файла с проверкой на ошибку открытия, соответствия формату и данных заголовка
4. Присвоение данных из заголовка файла РСХ данным заголовка ВМР и создание файла ВМР. Сохранение информации из заголовка в созданный файл.
5. Считывание зашифрованного растра РСХ файла в буфер. Дешифрование полученного растра в буферный сегмент в цикле побайтово. Вычисление количества байтов выравнивания и построчный перебор с записью пикселей с проверкой на количество цветов (256).
6. Запись строк изображения в ВМР в цикле в обратном РСХ порядке. Перезапись заголовка файла с новым размером. Закрытие файлов. Вывод сообщения. Запуск меню.

Описание процедур и макросов

Названия и описания работы процедур представлены в Таблице 3.

Таблица 3.

Процедуры

Название процедуры	Назначение процедуры
GetFileName	Получение дескриптора файла с названием, введенным с клавиатуры
Close	Закрытие файла
FindColorIndex	Поиск цветов в палитре
SetPallete	Запись цветов в палитру
DisplayMenu	Отображение строк меню
ClearScreen	Очистка экрана
main	Основная программа

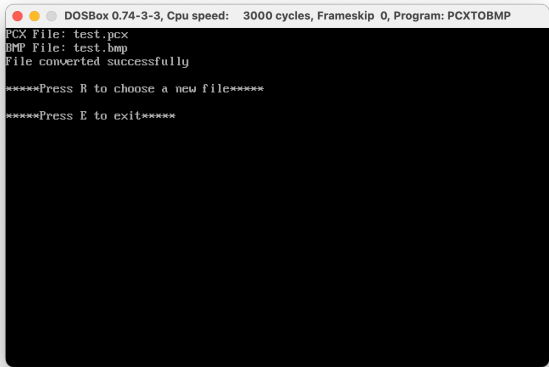
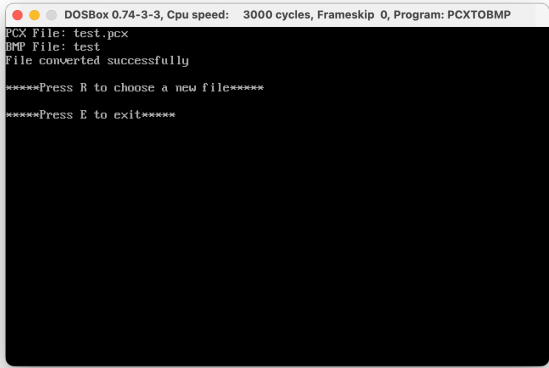
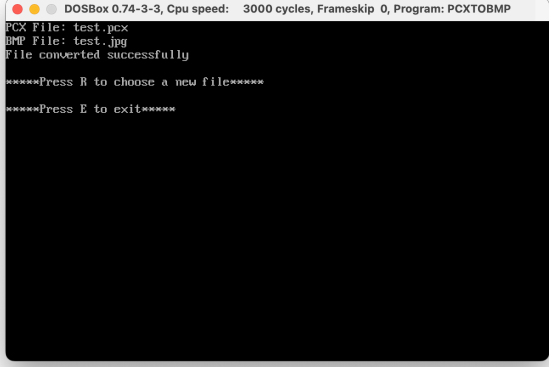
Весь код программы представлен в Приложении А.

Тестирование и отладка

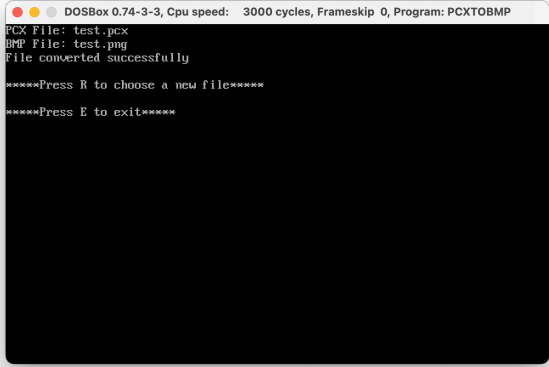
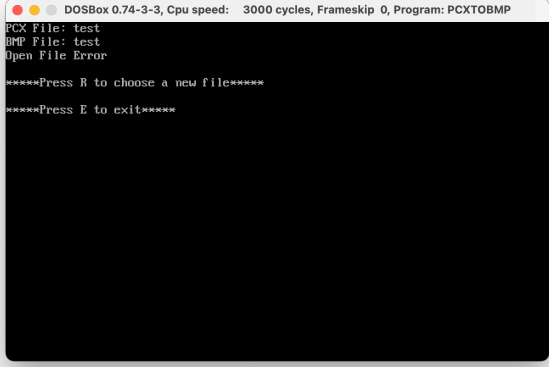
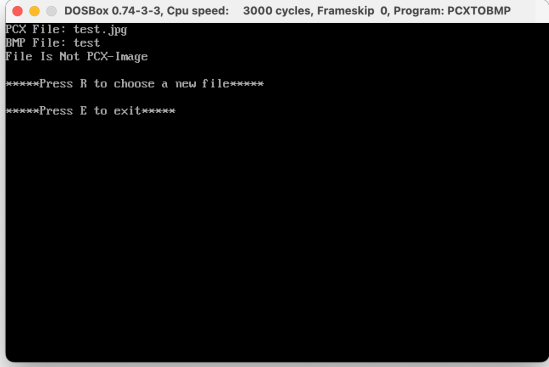
В таблице 4 представлены результаты тестирования программы.

Таблица 4.

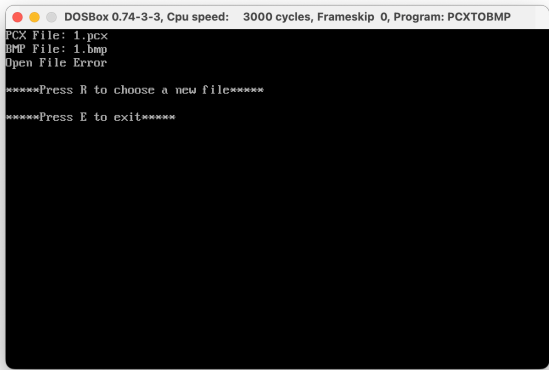
Тестирование и отладка

Тест №	Окно программы	Итог
1. Стандартная работа программы		Успешная конвертация
2. Установим название выходного файла без расширения		Успешная конвертация
3. Установим название выходного файла с расширением .jpg		Успешная конвертация

Продолжение таблицы 4

<p>4. Установим название выходного файла с расширением .png</p>		<p>Успешная конвертация</p>
<p>5. Установим название входного файла без расширения</p>		<p>Ошибка открытия файла</p>
<p>6. Установим название входного файла с расширением .jpg</p>		<p>Ошибка - неподходящий формат</p>

Продолжение таблицы 4

<p>7. Установим название несуществующего входного файла</p>		<p>Ошибка открытия файла</p>
-------------------------------------------------------------	------------------------------------------------------------------------------------	------------------------------

На рис. 8 представлен результат успешных конвертаций.

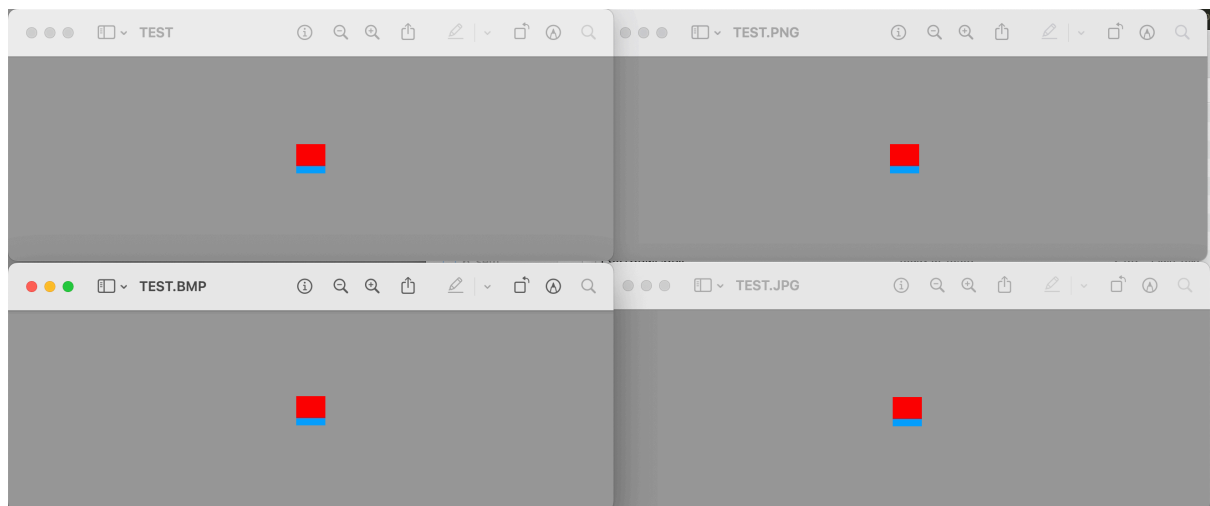


Рис. 8. Успешные конвертации

Анализ возможностей разработанной программы

Программа прошла тестирование. Возможно улучшение посредством более строгого ограничения расширения выходного файла и повышения уровня взаимодействия с пользователем.

Заключение

В процессе выполнения расчетного задания были изучены способы работы с файлами на языке ассемблера i8086, методы создания буферных сегментов данных и структура файлов форматов .rsx и .bmp. Была создана программа для конвертации изображений с расширением .rsx в изображения с расширением .bmp.

Список литературы

1. Абель П. Язык ассемблера для IBM PC и программирования. – М.: "Высшая школа", 1992. – 447 с.
2. www.codenet.ru: Формат файла изображений PCX. - URL: <http://www.codenet.ru/progr/formt/pcx1.php> (дата обращения (15.05.2022)). - Режим доступа: для всех пользователей.
3. pascal.sources.ru: Формат BMP файла. - URL: <https://pascal.sources.ru/articles/099.htm> (дата обращения (15.05.2022)). - Режим доступа: для всех пользователей.
4. kre.hww.ru: Работа с файлами в программах на ассемблере. – URL: <https://kre.hww.ru/ASM/Book/Charter7/1.htm#2> (дата обращения 14.05.2022). – Режим доступа: для всех пользователей.

Приложение А. Листинг программы

```
; Расчётное задание
; Балашов Савва, А-08-19
; Преобразование файла из формата PCX в формат BMP (256-
цветов на точку).

; сегмент стека
stk segment stack
db 100h dup ('?')
stk ends

; сегмент данных
data segment para public 'data'

PCXName db 255,255 dup (?)
; буфер для строки, хранящей имя файла pcx
BMPName db 255,255 dup (?)
; буфер для строки, хранящей имя файла bmp

; строки меню, которые будут выводиться на экран
MenuNewMsg db 0ah,0dh, '*****Press R to choose a new
file*****',0ah,0dh,'$'
MenuExitMsg db 0ah,0dh, '*****Press E to
exit*****',0ah,0dh,'$'

; строки сообщений, которые будут выводиться на экран
Prompt1 db 'PCX File: $'
Prompt2 db 0ah,0dh,'BMP File: $'
OFE db 0ah,0dh,'Open File Error ',0ah,0dh,'$'
NotPCX db 0ah,0dh,'File Is Not PCX-Image',0ah,0dh,'$'
Not256 db 0ah,0dh,'File Is Not 256-colors PCX',0ah,0dh,'$'
RE db 0ah,0dh,'Read error',0ah,0dh,'$'
CNCFMsg db 0ah,0dh,'Could Not Create File',0ah,0dh,'$'
DoneMsg db 0ah, 0dh, 'File converted successfully', 0ah,
0dh, '$'

; заголовок PCX
Manufacturer db ?
; Постоянный флаг 10 = ZSoft .PCX
Version db ?
; Информация о версии (должна быть 5)
Encoding db ?
; 1 = PCX кодирование длинными сериями
BitsPerPixel db ?
; кол-во бит на пиксель
WindowXmin dw ?
; верхняя левая граница изображения
WindowYmin dw ?
WindowXmax dw ?
; правая нижняя граница изображения
WindowYmax dw ?
Hres dw ?
; горизонтальное разрешение
Vres dw ?
; вертикальное разрешение
Colormap db 48 dup (?)
; информация о палитре
Reserved db ?
; зарезервировано
```

```

NPlanes db ?
; кол-во цветовых слоев
BytesPerLine dw ?
; кол-во байт на 1 цветовой слой ( ширина + байты для
выравнивания)
PaletteInfo dw ?
; как интерпретировать палитру
Filler db 58 dup(?)
; нули
PCXHeaderSize equ $-Manufacturer
; размер заголовка

; заголовок BMP
bfType db 'BM'
; информация о типе файла
bfSize dd ?
; размер самого файла в байтах
bfReserved1 dw 0
; нули
bfReserved2 dw 0
; нули
bfOffBits dd ?
; смещение относительно начала файла на битовый массив
растра
BMPHeaderSize equ $-bfType
; размер заголовка

; заголовок BMPInfo
biSize dd BMPInfoHeaderSize
; размер структуры BITMAPINFOHEADER
biWidth dd ?
; ширина изображения
biHeight dd ?
; высота изображения
biPlanes dw 1
; количество плоскостей
biBitCount dw 8
; кол-во бит на пиксель
biCompression dd 0
; тип сжатия
biSizeImage dd ?
; размер изображения в байтах
biXPelsPerMeter dd ?
; горизонтальное разрешение
biYPelsPerMeter dd ?
; вертикальное разрешение
biClrUsed dd 256
; текущее число цветов графического движка
biClrImportant dd ?
; кол-во важных цветов
BMPInfoHeaderSize equ $-biSize
; размер структуры BITMAPINFOHEADER

; палитра BMP
BMPPallete db 1024 dup(0)
; буфер для палитры BMP
BMPPalleteSize equ 1024
; фиксированный размер палитры
ColorsCount dw 0
; переменная для хранения кол-ва цветов в документе

```

```

AdditiveBytes db 0
; кол-во байт для выравнивания в PCX

PCXDescr dw ?
; дескриптор файла PCX
BMPDescr dw ?
; дескриптор файла BMP

ArraySize dw ?
; переменная для хранения кол-ва байт в запакованном растре
PCX
CurByte dw 0
; переменная, хранит номер следующего байта в растре bmp

; переменные для хранения текущих значений составляющих R, G
и B
CurR db ?
CurG db ?
CurB db ?
PackedArray db 60000 dup (?)
; буфер, в который считывается запакованный растр PCX
data ends

; сегмент для хранения распакованного растра PCX
MemoryBuf segment para public 'data'
    db 0FFFFh dup (?)
MemoryBuf ends
; сегмент для хранения растра BMP
Bitmap segment para public 'data'
    db 0FFFFh dup (0)
Bitmap ends

; сегмент кода
code segment para public 'code'

; основная программа
main proc
    assume cs:code,ds:data,ss:stk,es:MemoryBuf
    mov ax,data
    mov ds,ax
    mov ax,MemoryBuf
    mov es,ax

    jmp Start    ; переходим на начало программы

; сообщения об ошибках
FileNotFound:    ; если файл не найден
    mov ah,9
    mov dx, offset OFE
    int 21h
    jmp MainMenu

ReadErr:         ; если ошибка чтения из файла
    mov ah,9
    lea dx,RE
    int 21h
    jmp MainMenu

Start:
    call ClearScreen    ; очистка экрана

    ; ввод строки имени исходного PCX-файла

```

```

    push offset PCXName
    push offset Prompt1
    call GetFileName      ; введем строку и приведем ее к
требуемому виду

    ; ввод строки имени выходного BMP-файла
    push offset BMPName
    push offset Prompt2
    call GetFileName      ; введем строку и приведем ее к
требуемому виду

    ; Откроем файл
    lea dx,PCXName
    mov ah,3dh            ; ф-ия для открытия файла
    xor al,al             ; режим - чтение
    add dx,2              ; перейдем на начало строки
    int 21h               ; открыть файл
    jc FileNotFound       ; если ошибка, то сообщим об этом

    mov PCXDescr,ax       ; сохраним дескриптор открытого
файла
    mov bx,ax

    ; Читаем заголовок
    mov ah,3fh            ; ф-ия для чтения из файла
    lea dx,Manufacturer   ; адрес на начала буфера
    mov cx,PCXHeaderSize  ; размер заголовка
    int 21h
    jc ReadErr

    ; проверка соответствия файла формату PCX
    CheckPCX:

        ; проверка записи о формате
        mov ah,Manufacturer
        cmp ah,10
        je PCXVersionCheck ; если все верно, то дальше
проверяем
        push word ptr PCXDescr ; если нет, то закрываем
файл, печатаем сообщение об ошибке и выходим в меню
        call Close
        mov ah,9
        lea dx,NotPCX
        int 21h
        jmp MainMenu

    ; проверка соответствия версии формата
    PCXVersionCheck:
        mov ah,Version
        cmp ah,5
        je PCXBitPixelCheck ;если все верно, то дальше
проверяем
        push word ptr PCXDescr ; если нет, то закрываем
файл, печатаем сообщение об ошибке и выходим в меню
        call Close
        mov ah,9
        lea dx,NotPCX
        int 21h
        jmp MainMenu

    ; проверка кол-ва бит на пиксель (должно быть 8 для 256
цветов)

```

```

PCXBitPixelCheck:
    mov ah,BitsPerPixel
    cmp ah,8
    je CheckNPlanes          ;если все верно, то
проверяем NPlanes
    push word ptr PCXDescr   ; если нет, то закрываем
файл, печатаем сообщение об ошибке и выходим в меню
    call Close
    mov ah,9
    lea dx,Not256
    int 21h
    jmp MainMenu
; проверка кол-ва цветовых линий
; должны быть 3 - R, G и B
CheckNPlanes:
    mov ah,NPlanes
    cmp ah,3
    je CompleteCheck        ;если все верно, то формат
проверен
    push word ptr PCXDescr   ; если нет, то закрываем
файл, печатаем сообщение об ошибке и выходим в меню
    call Close
    mov ah,9
    lea dx,Not256
    int 21h
    jmp MainMenu

CompleteCheck:
; формирование размеров изображения, которые будут
указаны в BMP
    mov ax,WindowXmax
    sub ax,WindowXmin
    inc ax
    mov word ptr biWidth,ax ; сохраним в biWidth
    mov ax, WindowYmax
    sub ax,WindowYmin
    inc ax
    mov word ptr biHeight,ax ; в ax - высота изображения
    ; сформируем biSizeImage = biHeight * biWidth
    mov dx,word ptr biWidth
    mul dx
    mov word ptr biSizeImage,ax
    mov word ptr biSizeImage+2,dx

; запишем в структуру BMP (в памяти) информацию о
разрешении изображения
    mov ax,Hres
    mov word ptr biXPelsPerMeter,ax
    mov ax,Vres
    mov word ptr biYPelsPerMeter,ax

; сформируем BfOffBits - смещение на начало растра
    mov ax, BMPHeaderSize      ; размер заголовка
    add ax,BMPInfoHeaderSize   ; + размер
информационной структуры
    add ax,1024                 ; + размер палитры
    mov word ptr BfOffBits,ax   ; сохраним

; вычислим кол-во дополняющих байт для строки BMP
    mov ax,word ptr biWidth ; в ax - ширина изображения

```

```

; умножим ее на 3 и разделим на 4 (см. пояснительную
записку)
mov bl,4
mov cl,3
mul cl
div bl
mov AdditiveBytes,ah ; сохраним остаток

; Создадим / откроем выходной BMP-файл
lea dx,BMPName
mov ah,3ch ; ф-ия для создания
файла
mov cx,0 ; режим записи
add dx,2 ; перейдем на начало
строки
int 21h ; открыли файл
jnc SkipNotCreateMessage ; если нет ошибок, то
обрабатываем файл
push word ptr PCXDescr ; если ошибка открытия
то закрываем файл, печатаем сообщение об ошибке и выходим в
меню
call Close
mov ah, 9
lea dx,CNCFMsg
int 21h
jmp MainMenu

SkipNotCreateMessage:

mov BMPDescr,ah ; сохраним дескриптор файла
в переменную
mov bx,ah
; запишем заголовок BMP
lea dx,bfType
mov ah,40h ; ф-ия DOS для записи в файл
mov cx,BMPHeaderSize
int 21h

; запишем также информационную структуру
mov cx,BMPInfoHeaderSize
lea dx,biSize
mov ah,40h
mov bx,BMPDescr
int 21h

; считаем запакованный растр из PCX файла в буфер
PackedArray
mov ah,3fh
mov bx,PCXDescr
lea dx,PackedArray
mov cx,60000
int 21h
mov cx,ax ; количество считанных байт
кладем в cx (для организации цикла)
mov ArraySize,cx ; и записываем в переменную

;----- дешифрование растра PCX
-----

lea si,PackedArray ; si указывает на считанный растр
xor di,di ; di=0 - нулевое смещение в сегменте
MemoryBuf

```

```

UnPack:
    mov al,[si]          ; загружаем в al байт раstra
    cmp al,10111111b    ; является ли данный байт
эталонным ? (не счетчиком)
    jb Ethalon          ; если да, то на Ethalon
    ; если это счетчик, то вычленим количество
повторений с помощью сдвигов
    shl al,2
    shr al,2
    inc si              ; переходим к следующему байту

    dec cx              ; cx сразу уменьшим
    mov ah,[si]         ; в ah загрузим этот байт раstra
    ; цикл повторения байта ah раз
CycleWrite:
    mov es:[di],ah      ; запишем его в сегмент,
выделенный для расшифровки
    inc di              ; увеличим индекс

    dec al              ; уменьшим счетчик
повторений
    jnz CycleWrite      ; если еще не нужное
количество раз повторили байт, то пишем еще

    inc si              ; выставляем указатель на
след. байт
    jmp Next            ; переход к следующей
итерации
    ; Для эталонного байта
Ethalon:
    mov es:[di],al      ; пишем его в буферный сегмент
    inc di              ; увеличим индекс

    inc si              ; переходим к следующему байту

Next:

loop Unpack

;----- обработка расшифрованного раstra
-----

    ; вычислим кол-во байт, дополненных к строке развертки
до выравнивания.
    mov cx,word ptr biWidth          ; в cx - ширина
изображения
    mov ax,word ptr BytesPerLine     ; в ax - кол-во байт на
линию
    sub ax,cx                        ; найдем разницу
    mov di,ax                        ; в di кол-во лишних
байт
    mov cx,word ptr biHeight          ; в cx - высота
    xor si,si                         ; si=0
    ; будем в цикле перебирать все строки
ForHeight:
    push cx                          ; сохраним счетчик
строк, чтоб не испортить
    mov cx,word ptr biWidth          ; в cx - снова
ширина
    mov ax,cx                        ; запишем ее в ax
    add ax,di                        ; добавим число байт
для выравнивания

```



```

xor bx,bx                                ; bx=0

; в цикле будем перебирать все пиксели по ширине.
раздельно по разверткам R, G и B.
ForWidth:

    ; строки развертки хранятся в порядке R,G,B
    mov dl,es:[si+bx]                    ; в dl пишем
значение R текущего пикселя
    mov CurR,dl                          ; сохраняем
    add si,ax                            ; переходим к
линии G
    mov dl,es:[si+bx]                    ; в dl пишем
значение R текущего пикселя
    mov CurG,dl                          ; сохраняем
    add si,ax                            ; переходим к
линии B
    mov dl,es:[si+bx]                    ; в dl пишем
значение R текущего пикселя
    mov CurB,dl                          ; сохраняем
    sub si,ax                            ; вернули si на
начало линии R
    inc bx                               ; увеличили
счетчик обработанных пикселей
    call SetPallette                     ; вызываем
процедуру обработки пикселя
    loop ForWidth

    ; переход к следующей строке
    add si,ax
    add si,ax
    add si,bx
    add si,di
    mov al,AdditiveBytes                 ; в al - кол-во
дополняющих строку байт
    xor ah,ah                             ; ah=0
    ; увеличиваем счетчик байтов в BMP-растре
    mov cx,CurByte
    add cx,ax
    mov CurByte,cx
    pop cx                               ; восстанавливаем
счетчик строк
    loop ForHeight

    ; проверка адекватности кол-ва цветов
    mov ax,ColorsCount
    cmp ax,256                           ; если цветов меньше чем
256,
    jng SkipColorError                   ; то работаем дальше,
иначе обрабатываем ошибку
    ; закроем файлы, выведем сообщение и выйдем в меню
    push word ptr PCXDescr
    call Close
    push word ptr BMPDescr
    call Close
    mov ah,9
    lea dx,Not256
    int 21h
    jmp MainMenu

SkipColorError:

```

```

        ; запишем в BMP палитру
        mov cx,BMPPalleteSize
        lea dx,BMPPallete
        mov ah,40h
        mov bx,BMPDescr
        int 21h
        ; строки изображения в сегменте Bitmap хранятся в
        ; обычном порядке, а в BMP - в перевернутом, поэтому
        ; необходимо
        ; записать строки изображения в обратном порядке

        mov cx,word ptr biHeight                ; в cx - высота
        (кол-во строк фактически)
        mov bx,BMPDescr                        ; работаем с BMP-
        файлом
        WriteStrings:
            push cx                            ; сохраним cx
            mov cx,word ptr biWidth            ; теперь в cx -
        ширина изображения
            add cl,AdditiveBytes                ; в cl - кол-во
        байт для выравнивания
            mov ax,CurByte                      ; в ax - индекс
        текущего байта в сегменте
            sub ax,cx                          ; вычитаем из
        него размер строки. теперь указывает индекс на начало строки
            mov CurByte,ax                    ; сохраним
            mov dx,CurByte                    ; запишем в dx
        для записи в файл
            mov ax,Bitmap                     ; в ax - адрес
        сегмента Bitmap
            mov ds,ax                          ; теперь сегмент
        данных по умолчанию - Bitmap, поскольку dx - это смещение
        относительно ds
            mov ah,40h
            int 21h                            ; записали
        строку
            ; восстанавливаем сегментный регистр ds
            mov ax,data
            mov ds,ax
            pop cx                            ;
        восстанавливаем cx
        loop WriteStrings

        ; для BMP файла, укажем размер bfSize в заголовке
        ; вычислим его на основе уже сформированного файла
        mov ah,42h                            ; ф-ия для перемещения указателя по
        файлу
        mov al,2                              ; относительно конца файла
        xor cx,cx
        xor dx,dx
        int 21h
        ; теперь в dx:ax -длина файла
        mov word ptr bfSize,ax
        mov word ptr bfSize+2,dx
        ; теперь перейдем в начало файла
        mov ah,42h                            ; ф-ия для перемещения указателя по
        файлу
        mov al,0                              ; относительно начала файла
        xor cx,cx
        xor dx,dx
        int 21h
        ; и заново запишем заголовок

```

```

    lea dx,bfType
    mov ah,40h          ; ф-ия DOS для записи в файл
    mov cx,BMPHeaderSize
    int 21h
    ; закрываем входной и выходной файлы

    push BMPDescr
    call Close

    push PCXDescr
    call Close

    ; выводим сообщение об успехе
    mov ah,9
    lea dx, DoneMsg
    int 21h

    ; меню программы
MainMenu:
    call DisplayMenu

    ; ждём нажатие на клавиатуру
    mov ah, 00h
    int 16h

    ; сравниваем с нужными клавишами
    cmp al, 'R'
    je ReturnStart
    cmp al, 'r'
    je ReturnStart
    cmp al, 'E'
    je ExitProg
    cmp al, 'e'
    je ExitProg
    call ClearScreen
    jmp MainMenu

ReturnStart:
    jmp Start

ExitProg:
    mov ax,4C00h ; выходим
    int 21h

main endp

; в стеке - адрес сообщения, буфер строки
; выход: в bx дескриптор файла
GetFileName proc
    pop di          ; в di - адрес возврата
    pop dx          ; в dx - адрес сообщения
    ; вывод строки для ввода имени файла
    mov ah,9
    int 21h        ; выводим строку
    pop dx          ; в dx - адрес (смещение) на буфер для ввода
строки
    ; ввод имени
    mov ah,0ah      ; 0ah - ф-ия для буферизированного ввода
строки с клавиатуры
    int 21h

```

```

; формирование строки имени файла
mov bx,dx      ; в [bx] лежит размер буфера для ввода
inc bx         ; в [bx] - количество байт в строке
mov al,[bx]    ; в al - кол-во записанных байт
add bl,al      ; теперь bx указывает на последний записанный
байт в строке
inc bl         ; bx указывает на след. после последнего
байт
mov [bx],0     ; и помещаем туда ноль (необходимо для DOS)
push di        ; кладем в стек адрес возврата
ret
GetFileName endp

; Функция для закрытия файла
; ВХОД: на вершине стека - дескриптор файла
Close proc

mov ah,3eh     ; 3eh - ф-ия для закрытия файла
pop cx         ; в cx - адрес возврата
pop bx         ; берем из стека дескриптор файла
int 21h
push cx        ; снова в на вершине стека адрес возврата
ret
Close endp

; Ф-ия для поиска цвета в палитре
; Вход: -
; выход: в dx индекс цвета или 0ffffh если цвета
CurR, CurG, CurB в палитре нет
FindColorIndex proc
push cx ; сохраним cx и si в стеке
push si
lea bx,BMPPallete ; bx указывает на палитру
mov cx,ColorsCount ; в cx - кол-во цветов в палитре
xor si,si ; si=0
xor di,di ; di=0
mov dx,0ffffh ; по умолчанию - цвет в палитре не найден,
если не докажется обратное
; будем в цикле перебирать все цвета палитры
Find:
mov al,[bx+si] ; в al - значение состоявляющей B
текущего цвета
cmp al,CurB ; сравниваем с искомым
jne NextColor ; если цвета не идентичны то переходим
к проверке следующего цвета
mov al,[bx+si+1]; в al - значение состоявляющей G
текущего цвета
cmp al,CurG ; сравниваем с искомым
jne NextColor ; если цвета не идентичны то переходим
к проверке следующего цвета
mov al,[bx+si+2]; в al - значение состоявляющей R
текущего цвета
cmp al,CurR ; сравниваем с искомым
jne NextColor ; если цвета не идентичны то переходим
к проверке следующего цвета
mov dx,di ; если все проверки прошли успешно, то
искомый цвет найдем и в dx помещаем его индекс

NextColor:
inc di ; переходим к следующему цвету
add si,4 ; в si - смещение на начала следующей
записи RGBQ

```

```

        loop Find
        pop si ; восстановим регистры
        pop cx
ret
FindColorIndex endp

; Ф-ия для обработки пикселя.
SetPallete proc
    ; сохраним регистры, которые не стоит портить
    push ax
    push bx
    push si
    push di
    push dx

    lea si,BMPPallete ; si указывает на палитру
    mov bx,ColorsCount ; в bx - кол-во цветов
    test bx,bx ; если bx=0
    jz NullColors ; то цветов нет и это будет первый
    call FindColorIndex ; если не первый, то ищем не было ли
данного цвета CurR, CurB, CurG в палитре
    cmp dx,0ffffh ; если уже был,
    jne SkipAddColor ; то не добавляем
AddColor:
    ; добавление цвета в палитру
    mov ax,ColorsCount ; в ax - кол-во цветов в палите
    mov bl,4
    mul bl ; умножаем на 4, чтобы получить
смещение относительно начала палитры (каждый цвет 4мя
байтами кодируется)
    mov bx,ax ; в bx будет лежать это смещение
NullColors: ; если первый цвет, то все что было
выше нам не нужно
    ; в BMP данные хранятся в порядке BGRQ, где Q - 00
всегда. в таком порядке их и записываем:
    mov al,CurB ; в al составляющая B
    mov [si+bx],al
    mov al,CurG ; в al составляющая G
    mov [si+bx+1],al
    mov al,CurR ; в al составляющая Q
    mov [si+bx+2],al
    mov [si+bx+3],0
    mov dx,ColorsCount ; в dx индекс данного цвета в
палитре
    inc ColorsCount ; увеличиваем кол-во цветов

    ; запишем теперь пиксел (а именно порядковый номер цвета
в палитре) в растр
SkipAddColor:
    push es ; сохраним зн-ие сегментного
регистра es
    mov ax,Bitmap ; в ax - адрес сегмента Bitmap
    mov es,ax ; теперь es на него указывает
    mov si,CurByte ; si указывает на текущий байт
растра
    mov es:[si],dl ; записываем зн-ие индекса
палитры
    inc CurByte ; и увеличиваем указатель

    ; восстановим es и необходимые регистры из стека
    pop es
    pop dx

```

```

        pop di
        pop si
        pop bx
        pop ax
ret
SetPallette endp

; ф-ия вывода строк меню
DisplayMenu proc
; смещение и вывод
    lea dx, MenuNewMsg
    mov ah,9
    int 21h
    lea dx, MenuExitMsg
    mov ah,9
    int 21h
    ret
DisplayMenu endp

; ф-ия очистки экрана
ClearScreen proc
    mov ah,0
    mov al,3
    int 10H
    ret
ClearScreen endp

code ends

end main

```