**РАСЧЕТНОЕ ЗАДАНИЕ**

**Отображение монохромного (1 бит на точку) файла BMP на экране (рекомендуемый видеорежим - 256 цветов)**  
**Курс: Системное программное обеспечение**

|  |  |
| --- | --- |
| Выполнил |  |
| Студент | Винокуров Родион Николаевич |
| Группа | А-12-19 |
| Вариант | 6 |

|  |  |
| --- | --- |
| ПринялПреподаватель | Гольцов Александр Геннадьевич |
| Дата |  |
| Подпись |  |

**Содержание**

**Введение3**

1. **Анализ задания3**
2. **Проектирование пользовательского интерфейса4**
3. **Форматы представления данных4**
4. **Работа с файлом5**
5. **Работа с видеорежимом6**
6. **Работа с вводом/выводом**7
7. **Вывод изображения на экран7**
8. **Тестирование и отладка10**
9. **Анализ возможностей разработанной программы14**
10. **Заключение14**

**Список литературы14**

**Приложение А. Листинг программы14**

# **Введение**

Актуальность темы: формат BMP все еще достаточно часто используется для хранения изображений, хоть и по большей части замещен растровыми форматами со сжатием, например, JPEG.

Необходимые навыки для написания данной программы:

* 1. Навыки написания кода для работы с файлами в ассемблере, а именно, открытия, закрытия, и чтения файлов.
  2. Навыки работы с процедурами в ассемблере: написание, вызов, передача аргументов в процедуру разными способами.
  3. Умение работы с файлами BMP (знание формата и его особенностей).
  4. Базовые знания языка ассемблер.

**1. Анализ задания**

**Техническое задание:** спроектировать и отладить программу

для вывода на экран монохромного BMP файла в режиме 256

цветов.

**Конкретизация задания:** программа должна запрашивать у

пользователя имя файла для чтения. Указанный файл будет

открыт программой, прочитан, и выведен на экран. После

этого программа будет ожидать от пользователя ввода

любого символа с клавиатуры. Когда это произойдет,

программа должна очистить экран и завершить свою работу.

Разбиение общей задачи на подзадачи:

-Сохранение в памяти введенного пользователем имени

файла.

-Открытие указанного пользователем файла.

-Чтение и анализ заголовка файла.

-Установка нужного графического режима для отображения

данных из файла на экране.

-Чтение графических данных в буфер из файла.

-Отображение считанных в буфер графических данных на

экране

-Возвращение старого графического режима.

-Закрытие файла после завершения работы программы.

**Ограничения:**

1. Размер входного файла для отображения– меньше 4 Гб.

Данное ограничение устанавливается ограничением адреса указателя файла в 32 бита. 2^32 = 4 Гб. А еще нужно учитывать размер заголовка отображаемого файла.

**2. Проектирование пользовательского интерфейса**

Т.к. для работы программы необходим только адрес

открываемого файла интерфейс можно организовать в виде

приглашения ввода адреса открываемого файла.

Формат приглашения:

Enter file name: ***имя\_файла***

Здесь: ***имя\_файла*** это имя файла в директории, из которой

запускается программа, либо его адрес.

**3. Форматы представления данных**

Сообщения, выдаваемые на экран:

Ввод имени файла:

entmsg db 'Enter file name: $'

Ошибка открытия файла:

errmsg db 10,13,'File opening error!',10,13,'$'

Ошибка закрытия файла:

clerr db 10,13,'File closing error!',10,13,'$'

Ошибка формата файла:

forerr db 10,13,'The opened file is not a BMP file!',10,13,'$'

Ошибка формата BMP файла:

monerr db 10,13,'The opened file is not monochrome!',10,13,'$'

Строка для хранения имени файла:

string db 255,255,255 dup (0)

Длина строки с именем файла:

len db 0

Дескриптор файла:

descr dw ?

Буфер для одной строки изображения (320 пикс=320 бит=40 байт):

buff db 40 dup (0)

Хранит номер изначального видеорежима:

oldMode db ?

Первые два байта открытого файла (4d42 указывает на BMP файл):

filetype dw ?

Смещение графических данных файла:

bfOffBits dd ?

Ширина изображения:

biWidth dd ?

Высота изображения:

biHeight dd ?

Количество бит на точку (должен быть равен 1!):

biBitCount dw ?

Смещение в байтах для перехода к следующей строке изображения:

bytesinstr dw ?

Количество байт, которые будут прочитаны в буфер и отображены на экране:

bytesdrawing dw ?

Координата по вертикали самой верхней строки изображения (отсчет идет с 200 в

сторону уменьшения):

yres dw ?

**4. Работа с файлом**

Используются прерывания DOS int 21h.

*Открытие файла:*

AH=3Dh

DS:DX=адрес строки ASCIIZ с именем файла

AL = режим открытия

AL = 0 чтобы открыть для чтения

AL = 1 чтобы открыть для записи

AL = 2 чтобы открыть для чтения и записи

AX = код ошибки если CF установлен и описатель файла если

нет ошибки.

*Установка файлового указателя:*

AH = 42h

BX = описатель файла

CX:DX = на сколько передвинуть указатель: (CX \* 65536) + DX

AL = 0 переместить к началу файла + CX:DX

AL = 1 переместить к текущей позиции + CX:DX

AL = 2 переместить к концу файла + CX:DX

AX = код ошибки если CF установлен

DX:AX = новая позиция указателя файла (если нет ошибки)

*Чтение из файла:*

AH=3Fh

BX = описатель файла

DS:DX = адрес буфера для чтения данных

CX = число считываемых байт

AX = код ошибки если CF установлен или число действительно

прочитанных байт

*Закрытие файла:*

AH = 3Eh

BX = описатель файла

AX = код ошибки если CF установлен

**5. Работа с видеорежимом**

Используются прерывания DOS int 10h.

*Читать текущий видеорежим:*

AH=0Fh

AL=текущий режим

AH = количество столбцов символов

BH = активная страница

*Установить новый видеорежим:*

AH = 00h

AL = видео режим

AL = флаг режима видео / байт режима контроллера ЭЛТ

**6. Работа с вводом/выводом**

Используются прерывания DOS int 21h.

*Вывод строки на экран:*

AH = 09h

DS:DX = адрес строки, заканчивающейся символом '$' (ASCII

24H)

*Буферизованный ввод строки:*

AH = 0Ah

DS:DX = адрес входного буфера

*Консольный ввод без эха:*

AH = 08h

AL = символ, полученный через стандартный ввод

**7. Вывод изображения на экран**

Для отображения изображений в данной работе будут использоваться стандартные графические видеорежимы, т.к. они поддерживаются всеми видеоадаптерами. С помощью функции 00 прерывания 10h, в частности, можно установить следующие VGA режимы:

*Таблица 1*

|  |  |  |
| --- | --- | --- |
| Номер режима | Разрешение | Число цветов |
| 11h | 640x480 | 2 |
| 12h | 640x480 | 16 |
| 13h | 320x200 | 256 |

Т.к. видеорежим 13h поддерживает рекомендуемое в этой работе число цветов (256), мы будем работать с ним. BIOS также предоставляет следующую функцию для записи точки на экране в данных графических режимах:

*Вывести точку на экран (int 10h):*

АН = 0Ch

ВН = номер видеостраницы (игнорируется для режима 13h, поддерживающего только одну страницу)

DX = номер строки

СХ = номер столбца

AL = номер цвета (для режимов 10h и llh, если старший бит 1, номер цвета точки на экране будет результатом операции «исключающее ИЛИ»)

Однако из-за того, что данная функция слишком медленно исполняется, вместо нее мы будем напрямую работать с видеобуфером. В видеорежиме 13h каждый байт в области памяти, начинающейся с адреса A000h:0000h, соответствует одной точке на экране, а значение, которое может принимать этот байт (0 – 255), соответствует номеру цвета этой точки. Таким образом, если сегментный регистр es настроен на адрес A000h, регистр di содержит смещение пикселя в данном сегменте, а регистр al содержит цвет пикселя, вывод точки на экран осуществляется вот так:

mov byte ptr es:[di],al

Редактирование палитры осуществляется с помощью портов 3c8h и 3c9h. Сначала в первый порт поступает одно число, обозначающее индекс редактируемого цвета в палитре, затем во второй – три числа, обозначающие новые интенсивности каналов красного, зеленого, и синего цветов. Т.к. программа работает всего с двумя цветами – черным и белым, удобно будет использовать всего два цвета палитры: с индексом 0 и индексом 1. В первый запишем черный цвет (значение 0 во всех трех каналах), во второй – белый (значение 63 во всех трех каналах).

После установки палитры осталось лишь спроектировать две процедуры: первая будет выводить строку пикселей из буфера buff на экран (drawstr), вторая – окрашивать нужный пиксель экрана в белый цвет (drawpix). Начнем со второй процедуры.

Удобно передавать этой процедуре в качестве аргументов индексы столбца и строки, по которым находится нужный пиксель. Тогда в этой процедуре кроме вышеописанной работы с видеобуфером нужно рассчитывать индекс в памяти окрашиваемого пикселя. Пусть ***row*** – номер строки пикселя, а ***col*** – номер его столбца. Т.к. разрешение по горизонтали равно 320, а отсчет обеих величин начинается с нуля, смещение окрашиваемого пикселя равно 320\*(***row***-1)+***col***-1. Таким образом, перед тем, как редактировать видеобуфер, необходимо всего лишь рассчитать смещение нужного пикселя по вышеприведенной формуле.

Роль первой процедуры – отображать пиксельные данные из буфера buff на экране. Т.к. минимально адресуемый объем памяти в Ассемблере Intel 8086 – 1 байт, лучшим вариантом будет поступить так: помещать по одному байту из buff в регистровую память, а дальше с помощью маски выделять из этого байта каждый его бит, и если рассматриваемый пиксель нужно зажечь (т.е. рассматриваемый бит равен 1), вызывать процедуру drawpix. Собственно, по такому принципу и работает разработанная процедура drawstr.

И последнее – пара слов о том, как программа читает исходный BMP файл. Т.к. все используемые в этом процессе прерывания описаны выше, остановимся на самом алгоритме. Во-первых, т.к. разрешение изображения по горизонтали может быть как больше так и меньше стандартных для данного видеорежима, разумно сделать так, чтобы при большем или меньшем разрешении программа работала корректно. Для этого введем величины bytesdrawing и yres. В них записано реально отображаемое разрешение картинки, которое всегда меньше или равно 320\*200. При этом вторая величина как бы перевернута (yres=0 соответствует разрешение 200 пикселей, yres=100 – 100 пикселей и т.д.), т.к. строки пикселей в файле BMP обычно идут от нижней к верхней. Кроме того, необходимо учитывать выравнивание в BMP файле. Благодаря нему, размер памяти, который занимает в файле одна строка пикселей всегда кратен 4 байтам. Тогда длина строки изображения в байтах равна ***ceil(Width/32)\*4***, где ***Width*** –разрешение картинки по горизонтали. Но нам нужна даже не эта величина, а величина ***ceil(Width/32)\*4-bytesdrawing***. Т.к. наиболее оптимальный вариант чтения файла – последовательно, нам нужно, чтобы сдвиг указателя происходил относительно текущего его местоположения в файле и переносил нас на следующую строку. Но для этого нужно учитывать уже произошедший сдвиг указателя на ***bytesdrawing*** байт в результате чтения. Отсюда и получается вышеприведенная величина, хранящаяся в поле ***bytesinstr.*** Таким образом, алгоритм чтения файла нашей программой таков: читаем из файла ***bytesdrawing*** байт, затем сдвигаемся на ***ceil(Width/32)\*4-bytesdrawing*** байт вперед относительно текущего положения файлового указателя, после чего цикл повторяется.

**8. Тестирование и отладка**

Отладка проводилась с помощью отладчика Turbo Debugger и набора исходных файлов.

Для тестирования программы были использованы следующие наборы вводных данных и исходных файлов.

*Таблица 2*

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | Входные данные | Результат | Смысл теста |
| 1 | Монохромный файл BMP 320\*200, на ввод поступает его имя/адрес | (выведен исходный файл в его оригинальном виде) | Стандартный для программы формат файла и ввода |
| 2 | Монохромный файл BMP разрешения 500\*500, на ввод поступает его имя/адрес | (выведен нижний левый угол изображения в его оригинальном виде) | Файл больше максимально поддерживаемого программой разрешения |
| 3 | Монохромный файл BMP разрешения 100\*100, на ввод поступает его имя/адрес | (изображение выведено в оригинальном виде в левом нижнем углу экрана) | Файл меньше максимально поддерживаемого программой разрешения |
| 4 | Цветной файл BMP, на вход поступает его имя/адрес | (программа выводит сообщение об ошибке формата файла) | Входной файл –цветной BMP. |

*Продолжение Таблицы 2*

|  |  |  |  |
| --- | --- | --- | --- |
| 5 | Файл с расширением BMP, содержащий мусор, на вход поступает его имя/адрес | (программа выводит сообщение об ошибке формата файла) | Входной файл не соответствует формату BMP. |
| 6 | На вход поступает имя/адрес несуществующего файла | (программа выводит сообщение об ошибке открытия файла) | Файла с введенным именем не существует. |

*Таблица 3*

|  |  |
| --- | --- |
| № теста | Изображение |
| 1 |  |

*Продолжение Таблицы 3*

|  |  |
| --- | --- |
| 2 |  |
| 3 |  |
| 4 |  |

**9. Анализ возможностей разработанной программы**

Отказоустойчивость достигнута тестированием различных файлов с различными параметрами. Возможные пути улучшения – дальнейшая низкоуровневая оптимизация программы и добавление обработки программой более сложных исключительных ситуаций (к примеру, несоответствие графических данных внутри файла его заголовку).

**10. Заключение**

В данной работе было разработано приложение для отображения монохромного BMP изображения в разрешении 320\*200 в видеорежиме 256 цветов.

**Список литературы**

1. Абель П. Язык Ассемблера для IBM PC и программирования / Пер. с англ. Ю.В.Сальникова. – М.:Высш. шк., 1992. – 447 с.:ил.
2. Белецкий Я. Энциклопедия языка Си: Пер. с польск. – М.: Мир, 1992. – 687 с.,ил.
3. BMP. — Текст : электронный // Википедия : [сайт]. — URL: https://ru.wikipedia.org/wiki/BMP (дата обращения: 05.09.2022).

**Приложение А. Листинг программы**

;Курсовой проект по СПО

;Вариант: 6

;Выполнил: Винокуров Родион

;Группа: А-12-19

;Отображение монохромного (1 бит на точку) файла BMP на экране (рекомедуемый видеорежим - 256 цветов)

.model small

.stack 100h ;256 байт на стек

.data

;сообщения, выдаваемые на экран

entmsg db 'Enter file name: $' ;ввод имени файла

errmsg db 10,13,'File opening error!',10,13,'$' ;ошибка открытия файла

clerr db 10,13,'File closing error!',10,13,'$' ;ошибка закрытия файла

forerr db 10,13,'The opened file is not a BMP file!',10,13,'$' ;ошибка формата файла

monerr db 10,13,'The opened file is not monochrome!',10,13,'$' ;ошибка формата BMP файла

;параметры для ввода имени входного файла

string db 255,255,255 dup (0) ;строка для хранения имени файла

len db 0 ;длина строки с именем файла

;дескриптор файла

descr dw ?

;служебные переменные

buff db 40 dup (0) ;буфер для одной строки изображения (320 пикс=320 бит=40 байт)

oldMode db ? ;хранит номер изначального видеорежима

filetype dw ? ;первые два байта открытого файла (4d42 указывает на BMP файл)

bfOffBits dd ? ;смещение графических данных файла

biWidth dd ? ;ширина изображения

biHeight dd ? ;высота изображения

biBitCount dw ? ;количество бит на точку (должен быть равен 1!)

bytesinstr dw ? ;смещение в байтах для перехода к следующей строке изображения

bytesdrawing dw ? ;количество байт, которые будут прочитаны в буфер и отображены на экране

yres dw ? ;координата по вертикали самой верхней строки изображения (отсчет идет с 200 в сторону уменьшения)

.code

;процедура для рисования строки из буфера buff

;row - номер строки

drawstr proc near

ARG row

push bp

mov bp,sp

mov ax,row

mov bx,0 ;столбец, в котором находится текущий пиксель

mov dh,80h ;начальное значение маски (единица в самой левой позиции)

mov bp,0 ;смещение читаемого байта внутри буфера

;пусть dl - очередной байт буфера, dh - маска

@@cycle:

inc bx

lea si,buff ;загружаем в si адрес начала буфера

add si,bp ;прибавляем смещение

mov dl,[si]

and dl,dh ;накладываем маску на прочитанный байт

cmp dl,0

jz @@sk1\_ ;если в рассматриваемом бите 1 - вызываем drawpix

push dx

push di

push bx

push ax

call drawpix

pop ax

pop bx

pop di

pop dx

@@sk1\_:

ror dh,1 ;сдвигаем единицу в маске на одну позицию вправо

jnc @@cycle ;перенос=>байт прочитан, переход к следующему байту

inc bp

cmp bp,40

jnz @@cycle

pop bp

retn

drawstr endp

;нарисовать точку

;row - строка, col - столбец

drawpix proc near

ARG row,col

push bp

mov bp,sp

;di=320\*(row-1)+col-1 - смещение пикселя от начала видеобуфера

mov ax,row

mov bx,col

dec ax

mov dx,320

mul dx

add ax,bx

mov di,ax

dec di

mov byte ptr es:[di],1 ;помещаем единицу в видеобуфер по нужному смещению

pop bp

retn

drawpix endp

start:

;инициализация сегментных регистров

mov ax,@data

mov ds,ax

mov es,ax

;вывод сообщения о вводе имени файла

mov dx,offset entmsg

mov ah,09h

int 21h

;ожидание ввода строки

mov ah,0Ah

mov dx,offset string

int 21h

;сохраняем в len текущую длину строки

mov si,offset string

inc si ;помещаем в si смещение второго байта строки (текущая длина)

mov al,[si]

mov len,al

;обнуляем последний символ строки

mov si,offset string

mov ax,si

add al,len

mov si,ax

add si,2 ;первые два символа: первоначальная и текущая длина строки

mov byte ptr [si],0

;открытие файла

add dx,2 ;пропускаем два первых символа

xor al,al ;режим только для чтения

mov ah,3dh

int 21h

jnc fexist\_

;вывод сообщения об ошибке при открытии файла

mov dx,offset errmsg

mov ah,09h

int 21h

;завершение работы программы

mov ax, 4C00h

int 21h

fexist\_:

;читаем первые два байта из файла

mov descr,ax ;сохраняем дескриптор файла в descr

mov ah,3fh

mov bx,descr

mov cx,2

mov dx,offset filetype ;читаем тип файла в filetype

int 21h

;файла формата BMP?

mov ax,[filetype]

cmp ax,4d42h

jz fisbmp\_

;выводим сообщение об ошибке формата

mov dx,offset forerr

mov ah,09h

int 21h

;пытаемся закрыть файл

mov ah,3eh

mov bx,descr

int 21h

;не удалось закрыть файл - сообщение об ошибке

jnc noclerr

mov dx,offset clerr

mov ah,09h

int 21h

;завершение работы программы

noclerr:

mov ax, 4C00h

int 21h

fisbmp\_:

;переходим к позиции в файле 0Ah (bfOffBits)

mov ax,4200h

mov bx,descr

xor cx,cx

mov dx,0Ah

int 21h

;читаем значение из файла в bfOffBits

mov ah,3fh

mov cx,4

mov dx,offset bfOffBits

int 21h

;переходим к позиции в файле 12h (biWidth)

mov ax,4200h

xor cx,cx

mov dx,12h

int 21h

;читаем значение из файла в biWidth

mov ah,3fh

mov cx,4

mov dx,offset biWidth

int 21h

;переходим к позиции в файле 16h (biHeight)

mov ax,4200h

xor cx,cx

mov dx,16h

int 21h

;читаем значение из файла в biHeight

mov ah,3fh

mov cx,4

mov dx,offset biHeight

int 21h

;находим yres

lea si,biHeight

mov ax,[si]

mov yres,ax

cmp ax,200 ;сравниваем biHeight с 200

jc stheight

mov yres,200 ;если biHeight>=200, то yres=200, иначе yres=biHeight

stheight:

mov ax,200

sub ax,yres ;yres:=200-yres (т.к. строки в файле BMP расположены снизу вверх)

mov yres,ax

;переходим к позиции в файле 1Ch (biBitCount)

mov ax,4200h

xor cx,cx

mov dx,1Ch

int 21h

;читаем значение из файла в biBitCount

mov ah,3fh

mov cx,2

mov dx,offset biBitCount

int 21h

;файл монохромный? (biBitCount=1?)

mov ax,biBitCount

cmp ax,1

jz fismon\_

;файл не монохромный=>сообщение об ошибке

mov dx,offset monerr

mov ah,09h

int 21h

;пытаемся закрыть файл

mov ah,3eh

mov bx,descr

int 21h

;не удалось закрыть файл=>выводим сообщение об ошибке

jnc noclerr\_\_

mov dx,offset clerr

mov ah,09h

int 21h

;завершение работы программы

noclerr\_\_:

mov ax, 4C00h

int 21h

fismon\_:

;устанавливаем граф.режим

mov ah,0Fh

int 10h ;определим текущий граф. режим

mov oldMode,al ;сохраним его в oldMode

mov ah,00h

mov al,13h ;видеорежим 13h (320\*200,256 цветов)

int 10h

push 0A000h ;сегментный адрес видеобуфера

pop es ;es=0A000h

;выбираем индекс 0 в порте выбора палитры (3c8h)

mov dx,3c8h

mov al,0

out dx,al

;значение красного цвета

mov dx,3c9h ;установка цвета производится через порт 3c9h

mov al,0

out dx,al

;значение зеленого цвета

mov al,0

out dx,al

;значение синего цвета

mov al,0

out dx,al

;выбираем индекс 1 в порте выбора палитры (3c8h)

mov dx,3c8h

mov al,1

out dx,al

;значение красного цвета

mov dx,3c9h ;установка цвета производится через порт 3c9h

mov al,63

out dx,al

;значение зеленого цвета

mov al,63

out dx,al

;значение синего цвета

mov al,63

out dx,al

;расчет длины строки изображения в байтах

lea si,biWidth

mov ax,[si]

mov dx,[si+2]

mov bx,32

div bx ;ax=biWidth div 32,dx=biWidth mod 32

cmp dx,0

jz noalign\_

inc ax ;biWidth не кратно 32=>прибавляем 1 к ax (длина округляется в большую сторону)

noalign\_:

mov bx,4

mul bx ;переводим величину в байты

mov bytesinstr,ax ;bytesinstr - длина строки в байтах с учетом выравнивания

;находим число выводимых на экран в одной строке байт

mov bytesdrawing,ax

cmp ax,40

jc lwidth ;если bytesinstr>=40, то bytesdrawing=40,иначе bytesdrawing=bytesinstr

mov bytesdrawing,40

lwidth:

;bytesinstr:=bytesinstr-bytesdrawing (при чтении указатель будет смещаться на bytesdrawing байт)

mov ax,bytesdrawing

sub bytesinstr,ax

;пусть cx - координата строки изображения

mov cx,200

push cx ;сохраняем координату строки в стеке

;переходим к графическим данным изображения

mov ax,4200h

mov bx,descr

lea si,bfOffBits

mov cx,[si+2]

mov dx,[si]

int 21h

;читаем строку в буфер buff

mov ah,3fh

mov cx,bytesdrawing

mov dx,offset buff

int 21h

call drawstr ;вывод строки изображения на экран

pop cx ;восстанавливаем cx

dec cx ;переходим к следующей строке изображения

DRAWSTRING:

push cx ;сохраняем координату строки в стеке

;сдвигаемся на bytesinstr байт от текущей позиции

mov ax,4201h

mov bx,descr

mov cx,0

mov dx,bytesinstr

int 21h

;читаем строку в буфер buff

mov ah,3fh

mov cx,bytesdrawing

mov dx,offset buff

int 21h

call drawstr ;вывод строки изображения на экран

pop cx ;восстанавливаем cx

dec cx ;переходим к следующей строке изображения

cmp cx,yres ;достигнута последняя строка изображения?

jnz DRAWSTRING

;ожидание ввода символа для задержки изображения на экране

mov ah,08h

int 21h

;восстановим старый видеорежим

mov ah,0

mov al,oldMode

int 10h

;пытаемся закрыть файл

mov ah,3eh

mov bx,descr

int 21h

;не удалось закрыть файл - выводим сообщение об ошибке

jnc noclerr\_

mov dx,offset clerr

mov ah,09h

int 21h

;завершение работы программы

noclerr\_:

mov ax, 4C00h

int 21h

end start