Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

ВОСТОЧНО-СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ

(ФГБОУ ВПО ВСГУТУ)

Технологический колледж

Дисциплина «Системное программирование»

Отчет по курсовой работе на тему

«Организация работы с видеопамятью в среде TASM 5.0»

Выполнил: ст. гр.74/1 Бубеев Б. Ж.

Проверила: Дамбаева С.В.

Улан-Удэ, 2015

Оглавление

[Введение 3](#_Toc437814117)

[Теоретическая часть 3](#_Toc437814118)

[Практическая часть 5](#_Toc437814119)

[Описание программы 5](#_Toc437814120)

[Словесное описание: 5](#_Toc437814121)

[Листинг программы: 8](#_Toc437814122)

[Скриншоты программы 12](#_Toc437814123)

[Заключение 14](#_Toc437814124)

[Список используемой литературы 15](#_Toc437814125)

# Введение

Видеопамять компьютера любой конфигурации расположена в адресном

пространстве оперативного запоминающего устройства (ОЗУ). Это позволяет напрямую адресовать видеопамять одним из косвенных способов адресации памяти. Видеопамять занимает адреса с A000h по BFFFh, что составляет 128 Кбайт. Для увеличения объёмов видеопамяти (до 64 Мбайт),

она делится на слои, так что по одному адресу находится несколько ячеек, которые расположены в разных слоях. Обращение к видеопамяти зависит от видеорежима, который определяет количество точек по горизонтали

и вертикали, а так же количество битов, отводимую для хранения кода

цвета каждой точки. Графическими режимами управляет видеоадаптер.

Более простым для программирования, допускающим простой доступ к

видеопамяти, является символьный режим, который мы и рассмотрим

подробнее. Для работы в символьном режиме отводится 16 Кбайт памяти,

начиная с адреса B800h. Экран делится на 80 столбцов и 25 строк. Общее

количество знакомест 80 х 25= 2000. Для каждого знакоместа в видеопамя-

ти отводится два байта: чётный байт – ASCII код символа, нечётный – байт

атрибутов. Счёт строк и колонок идёт из верхнего левого угла экрана, в

байте b800h:0000h хранится символ выводящийся в нулевой строке и ну-

левой колонке, в байте b800h:0001h хранится атрибут этого символа. В

байте b800h:0002h хранится символ выводящийся в нулевой строке и пер-

вой колонке, в байте b800h:0002h хранится атрибут этого символа и т.д.

# Теоретическая часть

Байт атрибутов имеет следующую структуру:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Фон | | | | Символ | | | |
| Атрибут | BL | R | G | B | I | R | G | B |
| Номер бита | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

BL – признак мерцания; R – красный цвет;

G – зелёный цвет; B – синий цвет;

I – Интенсивность свечения.

Для доступа к видеопамяти в текстовом режиме можно использовать

непосредственно один из сегментных регистров, например, ES:

mov ax, 0b800h ; записать в регистр

mov es, ax ; es адрес начала видеопамяти

xor bx , bx ; смещение символа от начала видеопамяти

mov dh, 00010100b ; атрибуты: на голубом фоне красный символ

mov dl, 65h ; ASCII код символа

mov word ptr es:[bx], dx ; запись в видеопамять символа и атрибута

inc bx ; смещение для

inc bx ; следующего символа

Другой способ доступа – размещение сегмента данных фиксировано, в

области видеопамяти директивой AT.

video segment AT 0b800h

CHAR\_ATRIB db 4000 dup(?)

video ends

code segment

assume cs: code, ds:video

START: mov ax, video

mov ds, ax

xor si, si

mov dl, byte ptr CHAR\_ATRIB[si] ;чтение символа

inc si

mov dh, byte ptr CHAR\_ATRIB[si] ;чтение атрибута

. . .

# Практическая часть

## 3.1 Описание программы

Задача:

Выводить ежесекундно в правом верхнем углу экрана системное время “часы:минуты:секунды”.

## 3.2 Словесное описание:

Идентификатор **old** определяет ячейку памяти размером 4 байта, которая хранит адрес старого обработчика прерываний от таймера. Эта ячейка будет нужна, когда будет вызываться старый обработчик прерываний от таймера. Идентификатор **buf** определяет шаблон для формирования строки, содержащей значение текущего времени в формате часы:минуты:секунды. Последний байт в шаблоне - нулевой - нужен для определения длины шаблона.

Процедура **decode** преобразует двоично-десятичное число в регистре AL в два ASCII символа, соответствующих значению часов, минут или секунд (это зависит от конкретного значения, записанного в регистре AL). Процедура прибавляет к значению разряда числа (младший разряд находится в первых четырёх битах регистра AL, старший - в старших четырёх битах) ASCII код символа '0', тем самым формируя ASCII код для цифры младшего или старшего разряда. Далее этот ASCII код записывается в текущую позицию шаблона. После записи в шаблон значение указателя на текущий символ шаблона увеличивается на 3 (две цифры для часов, минут или секунд, плюс символ ':').

Процедура **clock** является обработчиком прерываний от таймера. Дело в том, что номер аппаратного обработчика прерываний от таймера - 08h. Но в системном обработчике этого аппаратного прерывания есть вызов INT 1Ch. Прерывание 1Ch определяет пользовательский обработчик прерываний от таймера, который вызывается системным. Таким образом, ячейка **old** хранит адрес старого пользовательского обработчика прерываний от таймера. В начале процедуры **clock** командой PUSHF в стеке подготавливается структура для команды IRET и затем вызывается старый пользовательский обработчик прерываний от таймера. Далее в процедуре **clock** сохраняются в стеке все модифицируемые регистры (в том числе и сегментные). После этого инициализируется сегментный регистр DS для последующего обращения к ячейкам памяти резидентной части программы. Следующим шагом является получение значения текущего времени из BIOS при помощи прерывания BIOS 1Ah. После вызова команды INT 1Ah в регистре CH находится значение часов, в регистре CL - значение минут и в регистре DH - значение секунд. Каждое значение представлено в двоично-десятичном формате - младший разряд числа находится в младших четырёх битах регистра, а старший разряд числа - в старших четырёх битах. После получения значения текущего времени регистр BX настраивается на начало шаблона для записи в шаблон значений часов, минут и секунд. Далее три раза вызывается процедура **decode** для записи в шаблон соответственно часов, минут и секунд. После записи в шаблон необходимой информации происходит вывод символов шаблона в правый верхний угол экрана. Для этого регистр ES настраивается на сегмент видеопамяти, а регистр DI присваивается значение 140. Далее в цикле происходит вывод в видеопамять символов шаблона и атрибутов. После этого восстанавливаются значения всех модифицируемых процедурой регистров и командой IRET происходит возврат из обработчика.

В основной части программы при помощи функции DOS 35h происходит получение адреса старого пользовательского обработчика прерываний от таймера (прерывание 1Сh). Значения сегмента и смещения старого обработчика записываются в ячейку **old**. Далее устанавливается адрес нашего обработчика прерываний от таймера. Для этого в регистр DX записывается смещение нашего обработчика (смещение процедуры **clock**) и вызывается функция DOS 25h (регистр DS уже содержит значение сегмента нашего обработчика). После этого происходит вызов функции DOS 31h, которая завершает программу и сохраняет часть её кода в памяти.

## 3.3 Листинг программы:

code segment

assume cs:code

start: jmp load

old dd 0 ;адрес старого обработчика

buf db ' 00:00:00 ',0 ;шаблон для вывода текущего времени

decode proc ;процедура заполнения шаблона

mov ah, al ;преобразование двоично-десятичного

and al, 15 ;числа в регистре al

shr ah, 4 ;в пару ASCII символов

add al, '0'

add ah, '0'

mov buf[bx + 1], ah ;запись ASCII символов

mov buf[bx + 2], al ;в шаблон

add bx, 3

ret

decode endp

clock proc ;процедура обработчика прерываний от таймера

pushf

call cs:old ;вызов старого обработчика прерываний

push ds

push es

push ax

push bx

push cx

push dx

push di

push cs

pop ds

mov ah, 02h ;функция BIOS для получения текущего времени

int 1Ah ;прерывание BIOS

xor bx, bx ;bx на начало шаблона

mov al, ch ;часы

call decode

mov al, cl ;минуты

call decode

mov al, dh ;секунды

call decode

mov ax, 0B800h

mov es, ax

mov di, 140 ;чтобы вывести в правом верхнем углу

xor bx, bx ;bx на начало шаблона

mov ah, 00001111b ;атрибут выводимых символов

@@1: mov al, buf[bx] ;цикл для записи символов шаблона в

видеопамять

stosw ;запись слова в строку

inc bx

cmp buf[bx], 0 ;пока не конец шаблона,

jnz @@1 ;продолжать запись символов

@@5: pop di ;восстановление модифицируемых регистров

pop dx

pop cx

pop bx

pop ax

pop es

pop ds

iret

clock endp

load: mov ax, 351Ch ;получение адреса старого обработчика

int 21h ;прерываний от таймера

mov word ptr old, bx ;сохранение смещения обработчика

mov word ptr old + 2, es ;сохранение сегмента обработчика

mov ax, 251Ch ;установка адреса обработчика

mov dx, offset clock ;установка смещения обработчика

int 21h

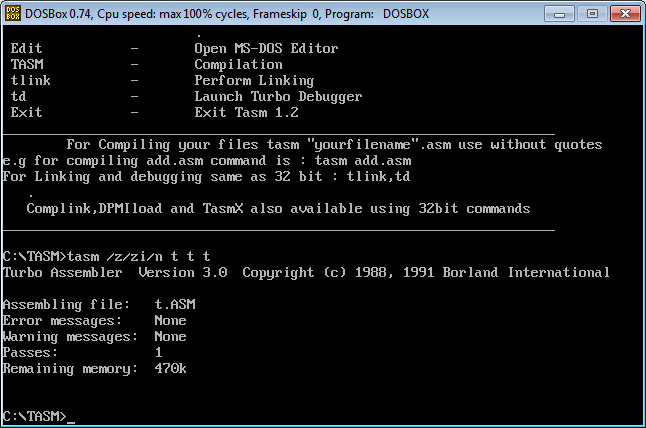
mov ah, 4ch

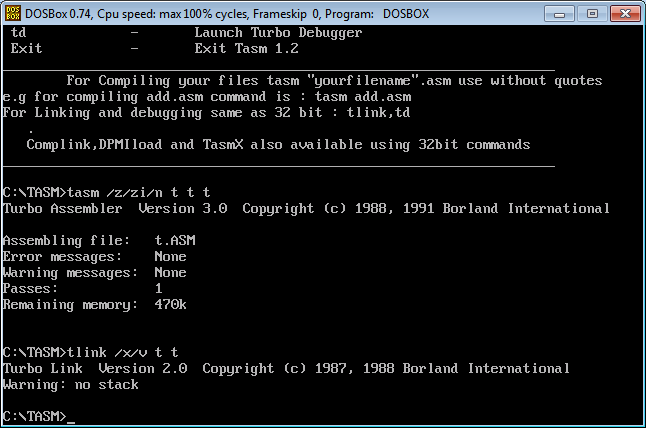
int 21h

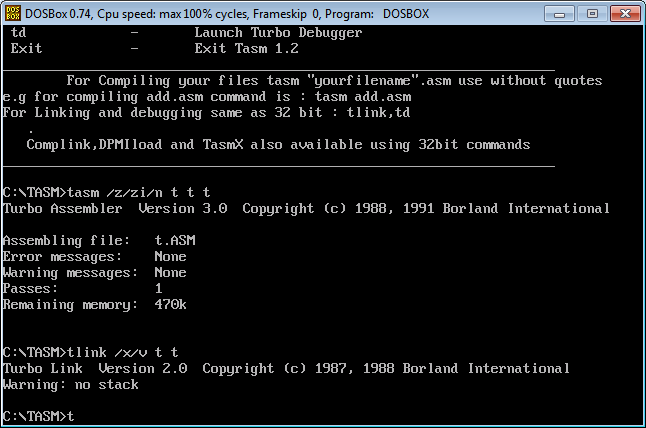
code ends

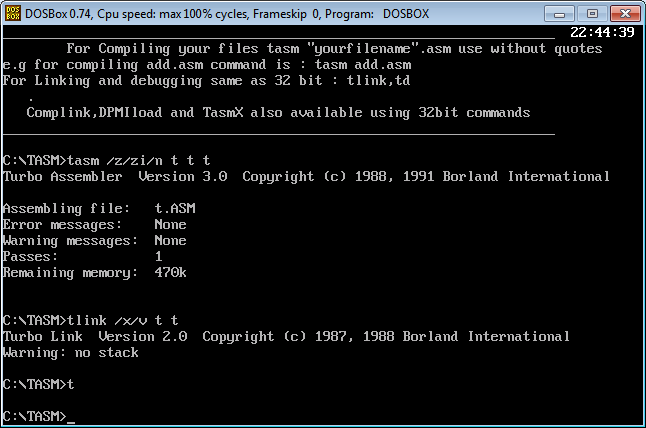
end start

# Скриншоты программы









Программа работает.

# Заключение

В заключении хочу сказать, что язык ассемблера помогает раскрыть все секреты аппаратного и программного обеспечения. С его помощью можно получить представление о том, как аппаратная часть взаимодействует с операционной системой и как прикладные программы обращаются к операционной системе. Большинство программистов работают с языками высокого уровня, где отдельное утверждение преобразовывается во множество процессорных команд. Ассемблер язык машинного уровня; каждая команда непосредственно интерпретируется в машинный код, что дает основание считать его языком низкого уровня. Наиболее часто язык ассемблера используется для написания дополнений к операционной системе или для написания программ прямого доступа к аппаратуре. Он необходим также при оптимизации критических блоков в прикладных программах с целью повышения их быстродействия.

# Список используемой литературы

1. Учебно-методическое пособие / ДГТУ / г. Ростов-на-Дону , 2001 г. / доц., к.т.н. Садовой Н.Н., доц., к.т.н. Жмайлов Б.Б.
2. <http://shackmaster.narod.ru/tsr.htm>
3. Функции BIOS - INT 1aH: ввод-вывод для времени http://www.codenet.ru/progr/dos/int\_0019.php
4. Функции DOS - INT 21H: сервис DOS <http://www.codenet.ru/progr/dos/int_0026.php>
5. Wikipedia.org