

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

ОТЧЕТ

по лабораторной работе №2

«Разработка программы ввода-вывода и обработки последовательности кодов на ассемблере»

Вариант 17

по дисциплине

«Принципы и методы организации системных программных средств»

РУКОВОДИТЕЛЬ:

(подпись)

Викулова Е.Н.

(фамилия, и.,о.)

СТУДЕНТ:

(подпись)

Сухоруков В.А.

(фамилия, и.,о.)

19-B-2

(шифр группы)

Работа защищена «__» _____

С оценкой _____

Нижний Новгород 2021

Оглавление

Цель.....	3
Вариант задания.....	3
Теоретическая часть	3
Форматы исполняемых файлов	3
Модели памяти	3
Требования, которые необходимо выполнять, при проектировании exe файла	4
Структура программы, организация ввода-вывода, используемые функции, особенности работы с видеопамятью	5
Структура программы	5
Много сегментная программа	5
Односегментная программа.....	5
Функции, используемые для ввода-вывода	6
Работа с видеопамятью	6
Тексты программ с комментариями.	7
lab2.asm.....	7
lab2com.asm	10
video.asm.....	13
Разбор содержимого файла. lst.....	15
Файл .map. Использование атрибута выравнивания.....	17
Выравнивание byte	17
Выравнивание word	17
Выравнивание para	18
Выравнивание page.....	18
Результаты выполнения программ	19
Lab2.exe	19
Lab2com.com	19
video.exe.....	19
Выводы	20

Цель

Приобретение навыков: разработки одно- и многосегментных программ на языке ассемблер, использования функций прерываний для организации ввода-вывода, управление трансляцией и компоновкой.

Вариант задания

Перестановка $a(n), a(n-1), a(n-2), \dots, a(n/2), a(1), a(2), \dots, a(n/2-1)$.

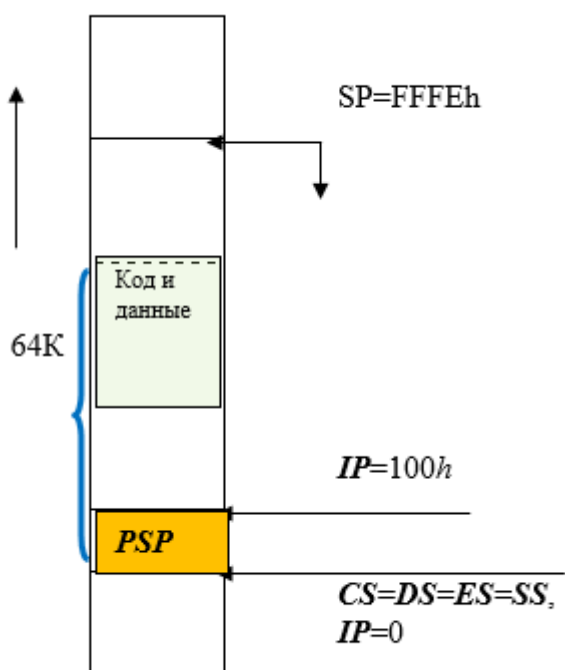
Теоретическая часть

Форматы исполняемых файлов

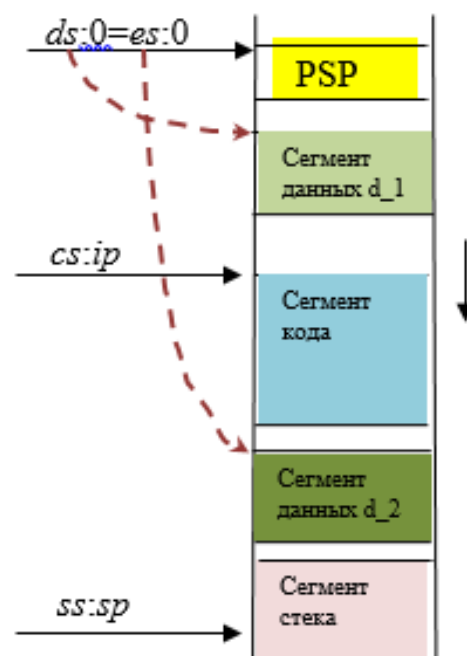
COM – полностью располагается в одном сегменте 64Кб, вся адресация - это смещения относительно одного сегментного адреса. Такой код не зависит от точки загрузки и может без настройки выполняться в любой области памяти.

EXE - программа состоит из нескольких сегментов. В EXE-файлах присутствует заголовочная часть с некоторым набором служебных таблиц для ОС (размером 512 байт или более).

Модели памяти



Структура COM файла



Структура EXE файла

Требования, которые необходимо выполнять, при проектировании exe файла

1. Определить точку входа в программу, на которую при загрузке инициализируются cs: ip, например

2. Описать сегмент стека для инициализации SS: SP, например, так:

```
st1 segment para stack 'stack'
```

```
...
```

```
st1 ends
```

или так

```
.model small
```

```
.stack 256
```

3. В начале программы явно проинициализировать регистры DS (и если нужно ES) на начало соответствующего сегмента.

Например, если есть два сегмента данных с именами d_1 и d_2, и мы хотим чтобы сегментные адреса этих сегментов были в ds и es, то в начале программы необходим следующий код:

```
mov ax,d_1
```

```
mov ds,ax
```

```
mov ax,d_2
```

```
mov es,ax
```

Структура программы, организация ввода-вывода, используемые функции, особенности работы с видеопамью

Структура программы Много сегментная программа

Сегмент *d1*

```
d1 segment para public 'data'
    mess1 db 'Input: ',10,13,'$'
    in_str db 22 dup (?)
d1 ends
```

Сегмент данных. Используется для инициализации строки сообщения и исходного массива.

Сегмент *e1*

```
e1 segment para public 'data'
    mess2 db 10,13, 'Output: ',10,13,'$'
    out_str db 20 dup ('$')
e1 ends
```

Сегмент данных. Используется для инициализации строки сообщения и нового массива.

Сегмент *c1*

Сегмент кода. Содержит код программы.

Сегмент стека

```
st1 segment para stack 'stack'
    dw 100 dup (?)
st1 ends
```

Сегмент кода

```
c1 segment para public 'code'
```

Односегментная программа

Содержит только один сегмент *“.code”*.

Функции, используемые для ввода-вывода

Для считывания массива байтов использована 10 функция 21h прерывания. Для вывода использована 9 функция 21h прерывания.

```
;Считывание исходного массива
```

```
mov dx, offset in_str
```

```
mov in_str, 20
```

```
mov ah, 10
```

```
int 21h
```

```
;Вывод нового массива
```

```
mov dx, offset out_str
```

```
mov ah, 9
```

```
int 21h
```

Работа с видеопамью

Для вывода информации из биоса были использованы константа, содержащая адреса памяти и цвета текста.

```
.data
```

```
bios equ 0FFFFh ;Адрес биоса
```

```
video equ 0B800h ;Начало видеопамью
```

```
color equ 0A3h ;Цвет текста
```

Инициализация сегментов данных:

```
mov ax, bios
```

```
mov es, ax
```

```
mov ax, video
```

```
mov ds, ax
```

Вывод информации:

```
mov al, es:[si]
```

```
mov ah, color
```

```
mov ds:[di], ax
```

Тексты программ с комментариями.

lab2.asm

```
; Лабораторная работа №2  
; Сухоруков Валерий  
; Перестановка a(n), a(n-1), a(n-2),  
; ..., a(n/2), a(1), a(2), ..., a(n/2-1).  
; Многосегментная программа. Создаётся exe файл
```

```
d1 segment para public 'data'  
mess1 db 'Input: ', 10, 13, '$'  
in_str db 22 dup (?)  
d1 ends
```

```
e1 segment para public 'data'  
mess2 db 10, 13, 'Output: ', 10, 13, '$'  
out_str db 20 dup ('$')  
e1 ends  
st1 segment para stack 'stack'  
dw 100 dup (?)  
st1 ends
```

```
c1 segment para public 'code'  
assume cs:c1, ds:d1, es:e1, ss:st1
```

```
start:
```

```
;Инициализация сегментов
```

```
mov ax, d1
```

```
mov ds, ax
```

```
mov ax, e1
```

```
mov es, ax
```

```
;Вывод строки запроса
```

```
mov dx, offset mess1
```

```
mov ah, 9
```

```
int 21h
```

```
;Считывание исходного массива
```

```
mov dx, offset in_str
```

```
mov in_str, 20
```

```
mov ah, 10
```

```
int 21h
```

```
;Инициализация регистра источника данных
```

```
; на конец исходного массива
```

```
mov cl, in_str + 1 ;Количество элементов в массиве
```

```
xor ch, ch ;Очистка регистра ch
```

```
mov ax, offset in_str+1 ;Адрес, предшествующий первому  
элементу
```

```
add ax,cx ;Запись адреса последнего  
элемента
```

```
mov si,ax ;Инициализация регистра  
источника данных
```

```
;Инициализация регистра приёмника данных
```

```
;на начало выходного массива
```

```
mov di, offset out_str
```

```
;Определения числа повторений первого цикла
```

```
;- Перестановка второй части исходного массива в начало  
нового в обратном порядке.
```

```
;Число повторений - количество элементов делить на 2.
```

```
shr cx,1 ;Деление выполняется сдвигом  
на 1 в право
```

```
m1:
```



```
mov al, [si]
mov es:[di],al
dec si
inc di
loop m1
```

```
;Инициализация регистра источника данных на начало
;исходного массива
```

```
mov si, offset in_str+2
```

```
;Определения числа повторений второго цикла - Перестановка
;первой части исходного массива в конец нового.
```

```
;Число повторений - количество элементов делить на 2.
```

```
mov cl, in_str + 1
```

```
shr cx,1
```

m2:

```
mov al,[si]
mov es:[di],al
inc si
inc di
loop m2
```

```
;Вывод строки, предшествующей выходному массиву
```

```
mov ax, es
```

```
mov ds, ax
```

```
mov dx, offset mess2
```

```
mov ah, 9
```

```
int 21h
```

```
;Вывод нового массива
```

```

    mov dx, offset out_str
    mov ah, 9
    int 21h

    mov ah, 7
    int 21h

    mov ax, 4c00h
    int 21h

c1 ends

end start

```

lab2com.asm

```

; Лабораторная работа №2
; Сухоруков Валерий
; Перестановка a(n), a(n-1), a(n-2),
; ..., a(n/2), a(1), a(2), ..., a(n/2-1).
; Односегментная программа. Создаётся com файл

.model tiny

.code
org 100h
_main:
    ;СЧИТЫВАНИЕ ИСХОДНОГО МАССИВА
    mov dx, offset in_str
    mov in_str, 20
    mov ah, 10
    int 21h

```

```

        ;Инициализация регистра источника данных
; на конец исходного массива
mov cl, in_str + 1          ;Количество элементов в массиве
xor ch, ch                  ;Очистка регистра ch
mov ax, offset in_str+1     ;Адрес, предшествующий первому
                             ;элементу
add ax,cx                   ;Запись адреса последнего
                             элемента
mov si,ax                   ;Инициализация регистра
                             ; источника данных

```

```

;Инициализация регистра приёмника данных

```

```

;на начало выходного массива

```

```

mov di, offset out_str

```

```

;Определения числа повторений первого цикла -

```

```

;Переставка второй части исходного массива в начало

```

```

;нового в обратном порядке.

```

```

;Число повторений - количество элементов делить на 2.

```

```

shr cx,1                    ;Деление выполняется сдвигом
                             ; на 1 в право

```

```

m1:

```

```

mov al, [si]
mov es:[di],al
dec si
inc di
loop m1

```

```

; Инициализация регистра источника данных

```

```

;на начало исходного массива
mov si, offset in_str+2

;Определения числа повторений второго цикла -
;Перестановка первой части исходного массива в конец нового.
;Число повторений - количество элементов делить на 2.
mov cl, in_str + 1
shr cx,1

m2:
mov al,[si]
mov es:[di],al
inc si
inc di
loop m2

;Вывод нового массива
mov dx, offset out_str
mov ah, 9
int 21h

mov ah, 7
int 21h

mov ax, 4c00h
int 21h

in_str db 22 dup (?)
out_str db 20 dup ('$')
end _main

```

video.asm

```
.model small
.stack 200h
.data
    bios equ 0FFFFh
    video equ 0B800h
    color equ 0A3h

.code
_main:
    ;Инициализация сегментов
    mov ah, 0
    mov al, 3
    int 10h
    mov ax, bios
    mov es, ax
    mov ax, video
    mov ds, ax

    ;Инициализация регистров источника и приёмника данных
    mov si, 05h
    mov di, 00h

    ;Инициализация счётчика
    mov cx, 0008h

m1:
    mov al, es:[si]
    mov ah, color
    mov ds:[di], ax
    inc di
```

```
inc di  
inc si  
loop ml
```

```
mov ah, 4ch  
mov al, 0  
int 21h
```

```
end _main
```

Разбор содержимого файла. lst

Фрагменты файла Lab2.lst

Turbo Assembler Version 3.1

14/11/21 13:13:29

Page 1

lab2.asm

В начале файла указывается версия транслятора, дата компиляции и название исходного файла.

```
6 0000 d1 segment para public 'data'
7 0000 49 6E 70 75 74 3A 20+ mess1 db 'Input: ',10,13,'$'
8 0A 0D 24
9 000A 16*(??) in_str db 22 dup (?)
10 0020 d1 ends
```

Далее следует описание сегмента данных d1. В первом столбце записано смещение относительно начала сегмента, во втором значения, которыми проинициализированы переменные. Для строки mess1 указана кодировка каждого символа в ASCII таблице. Для переменной in_str указано каким символом она проинициализирована и количество таких символов.

```
12 0000 segment para public 'data'
13 0000 4F 75 74 70 75 74+ mess2 db 'Output: ',10,13,'$'
14 3A 20 0A 0D 24
15 000B 14*(24) out_str db 20 dup ('$')
16 0021 ends
```

Описание сегмента данных e1 совпадает с описанием сегмента данных d1. Смещение относительно начала сегмента обнуляется.

```
26 0000 start:
27 ;Инициализация сегментов
28 0000 B8 0000s mov ax, d1
29 0003 8E D8 mov ds, ax
30 0005 B8 0000s mov ax, e1
31 0008 8E C0 mov es, ax
```

Сегмент кода. В первом столбце записано смещение относительно начала сегмента, во втором – КОП (код операции), в третьем – информация об операндах. Для команд, в которых используются переменные (d1 и e1) указано смещение переменных относительно начала их сегментов. Для команд, в которых используются только регистры записаны кодовые обозначения регистров.

```

86    004E  BA  0000F      mov dx, offset mess2
91    0055  BA  000DF      mov dx, offset out_str

```

В командах, использующих переменные, объявленные внутри сегмента данных в третьем столбце указано их **смещение**.

Symbol Name	Type	Value
??DATE	Text	"14/11/21"
??FILENAME	Text	"lab2 "
??TIME	Text	"13:13:29"
??VERSION	Number	030A
@CPU	Text	0101H
@CURSEG	Text	C1
@FILENAME	Text	LAB2
@WORDSIZE	Text	2
IN_STR	Byte	D1:000A
M1	Near	C1:002F
M2	Near	C1:0041
MESS1	Byte	D1:0000
MESS2	Byte	E1:0000
OUT_STR	Byte	E1:000D
START	Near	C1:0000

После окончания разбора исходного файла .asm в файле .lst формируется таблица локальных символических имён. В первом столбце указано имя переменной, во второй - тип, в третьем – значение.

Groups & Segments	Bit	Size	Align	Combine	Class
C1	16	0065	Para	Public	CODE
D1	16	0020	Para	Public	DATA
E1	16	0021	Para	Public	DATA
ST1	16	00C8	Para	Stack	STACK

Далее формируется таблица сегментов и групп. В первом столбце – название, во втором - разрядность, в третьем – размер, в четвёртом – выравнивание, в пятом – доступность (защищённость), в шестом – тип сегмента.

Файл .map. Использование атрибута выравнивания

Атрибут выравнивания сегмента сообщает компоновщику о том, что нужно обеспечить размещение начала сегмента на заданной границе.

Выравнивание byte

Выравнивание не выполняется.

<i>Start</i>	<i>Stop</i>	<i>Length</i>	<i>Name</i>	<i>Class</i>
00000H	0001FH	00020H	D1	DATA
00020H	00040H	00021H	E1	DATA
00041H	00108H	000C8H	ST1	STACK
00109H	0016DH	00065H	C1	CODE

Program entry point at 0010:0009

Выравнивание word

Сегмент начинается по адресу, кратному двум, т. е. последний (младший) значащий бит физического адреса равен 0 (выравнивание на границу слова).

<i>Start</i>	<i>Stop</i>	<i>Length</i>	<i>Name</i>	<i>Class</i>
00000H	0001FH	00020H	D1	DATA
00020H	00040H	00021H	E1	DATA
00042H	00109H	000C8H	ST1	STACK
0010AH	0016EH	00065H	C1	CODE

Program entry point at 0010:000A

Выравнивание para

Сегмент начинается по адресу, кратному 16d(10h).

Start	Stop	Length	Name	Class
00000H	0001FH	00020H	D1	DATA
00020H	00040H	00021H	E1	DATA
00050H	00117H	000C8H	ST1	STACK
00120H	00184H	00065H	C1	CODE

Program entry point at 0012:0000

Выравнивание page

Сегмент начинается по адресу, кратному 256d (100h).

<i>Start</i>	<i>Stop</i>	<i>Length</i>	<i>Name</i>	<i>Class</i>
<i>00000H</i>	<i>0001FH</i>	<i>00020H</i>	<i>D1</i>	<i>DATA</i>
<i>00100H</i>	<i>00120H</i>	<i>00021H</i>	<i>E1</i>	<i>DATA</i>
<i>00200H</i>	<i>002C7H</i>	<i>000C8H</i>	<i>ST1</i>	<i>STACK</i>
<i>00300H</i>	<i>00364H</i>	<i>00065H</i>	<i>C1</i>	<i>CODE</i>

Program entry point at 0030:0000

Результаты выполнения программ

Lab2.exe

```
Администратор: Командная строка - tlink lab2.obj
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\1>cd C:\Users\1\Desktop\assembler\LW2\lab2

C:\Users\1\Desktop\assembler\LW2\lab2>tasm lab2.asm /la
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International

Assembling file:   lab2.asm
Error messages:   None
Warning messages: None
Passes:           1
Remaining memory: 451k

C:\Users\1\Desktop\ASSEMB~1\LW2\lab2>tlink lab2.obj
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

C:\USERS\1\DESKTOP\ASSEMB~1\LW2\LAB2>lab2.exe
Input:
Sukhorukov
Output:
vokurSukho
C:\USERS\1\DESKTOP\ASSEMB~1\LW2\LAB2>
```

Lab2com.com

```
Администратор: Командная строка - tlink lab2com.obj /t - lab2com.com
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\1>cd C:\Users\1\Desktop\assembler\LW2\lab2com

C:\Users\1\Desktop\assembler\LW2\lab2com>tasm lab2com.asm /la
Turbo Assembler Version 3.1 Copyright (c) 1988, 1992 Borland International

Assembling file:   lab2com.asm
Error messages:   None
Warning messages: None
Passes:           1
Remaining memory: 452k

C:\Users\1\Desktop\ASSEMB~1\LW2\lab2com>tlink lab2com.obj /t
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

C:\USERS\1\DESKTOP\ASSEMB~1\LW2\LAB2COM>lab2com.com
vokurSukho_
```

video.exe

```
Администратор: Командная строка - tlink video.obj
C:\USERS\1\DESKTOP\ASSEMB~1\LW2\VIDEO>
```

Выводы

В ходе выполнения лабораторной работы были приобретены навыки: разработки одно- и многосегментных программ на языке ассемблер, использования функций прерываний для организации ввода-вывода, управление трансляцией и компоновкой.