

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №6
по дисциплине «Операционные системы»
Тема: Построение модуля динамической структуры

Студентка гр. 9382

Балаева М.О.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

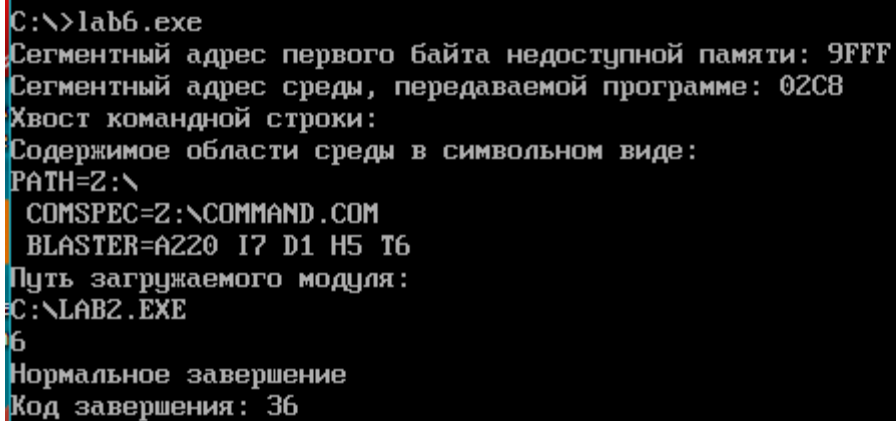
Исследование возможности построения загрузочного модуля динамической структуры. В отличие от предыдущих лабораторных работ в этой работе рассматривается приложение, состоящее из нескольких модулей, а не из одного модуля простой структуры. В этом случае разумно предположить, что все модули приложения находятся в одном каталоге и полный путь в этот каталог можно взять из среды, как это делалось в работе 2. Понятно, что такое приложение должно запускаться в соответствии со стандартами ОС. В работе исследуется интерфейс между вызывающим и вызываемым модулями по управлению и по данным. Для запуска вызываемого модуля используется функция 4B00h прерывания int 21h. Все загрузочные модули находятся в одном каталоге. Необходимо обеспечить возможность запуска модуля динамической структуры из любого каталога.

Описание функций и структур данных.

Название функции	Назначение
BYTE_TO_HEX	переводит число из AL в 2 16-ых символа и помещает их в AL и BH
PRINT	вызывает функцию печати строки
PREP	выполняет подготовку параметров для запуска загрузочного модуля
PREP_PAR	выполняет создание блок параметров
START_MOD	выполняет запуск загрузочного модуля
PROC_ER	функция обработки ошибок
ENTER_END	функция вывода причины и кода завершения загрузочного модуля

Ход работы.

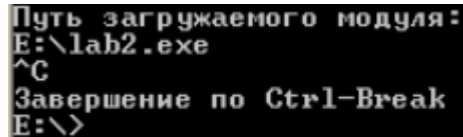
- Запуск программы, когда она находится в текущем каталоге с разработанным модулем lab2.exe и последующий ввод случайного буквенного символа на рис.1:



```
C:\>lab6.exe
Сегментный адрес первого байта недоступной памяти: 9FFF
Сегментный адрес среды, передаваемой программе: 02C8
Хвост командной строки:
Содержимое области среды в символьном виде:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Путь загружаемого модуля:
C:\LAB2.EXE
6
Нормальное завершение
Код завершения: 36
```

Рис.1

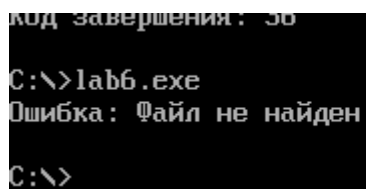
- К сожалению, DOS-BOX игнорирует комбинацию клавиш Ctrl-C, поэтому программа будет запущена в командной строке Windows 32-разрядного и затем вызвана комбинация клавиш на рис.2:



```
Путь загружаемого модуля:
E:\lab2.exe
^C
Завершение по Ctrl-Break
E:\>
```

Рис.2

- Пример запуска программы, когда модуль и программа находятся в разных каталогах на рис.3:



```
Код завершения: 36
C:\>lab6.exe
Ошибка: Файл не найден
C:\>
```

Рис.3

Заключение.

В процессе выполнения данной лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры.

Ответы на контрольные вопросы.

1. Как реализовано прерывание CTRL+C?

Вектор прерывания 23h, находящийся по адресу 0000:008Ch, содержит адрес, по которому DOS передает управление после обнаружения нажатия пользователем клавиш Ctrl-C.

Обычная системная обработка Ctrl-C сводится к немедленному снятию программы.

2. В какой точке заканчивается вызываемая программа, если код завершения 0?

При выполнении функции 4Ch прерывания int 21h;

3. В какой точке заканчивается вызываемая программа по прерыванию Ctrl+C?

В точке вызова функции 01h прерывания int 21h

ПРИЛОЖЕНИЕ А

lab6.asm

```
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, ES:DATA, SS:ASTACK
START: JMP BEGIN
; ПРОЦЕДУРЫ
;-----
TETR_TO_HEX PROC near
    and AL,0Fh
    cmp AL,09
    jbe NEXT
    add AL,07
NEXT: add AL,30h
    ret
TETR_TO_HEX ENDP
;-----
BYTE_TO_HEX PROC near
    push CX
    mov AH,AL
    call TETR_TO_HEX
    xchg AL,AH
    mov CL,4
    shr AL,CL
    call TETR_TO_HEX
    pop CX
    ret
BYTE_TO_HEX ENDP
;-----
PRINT PROC
    push ax
    mov AH,09h
    int 21h
    pop ax
    ret
PRINT ENDP
;-----
PREP PROC
    mov ax,ASTACK
    sub ax,CODE
    add ax,100h
    mov bx,ax
    mov ah,4ah
    int 21h
    jnc PREP_skip1
        call PROC_ER
    PREP_skip1:

    ; подготавливаем блок параметров:
    call PREP_PAR

    ; определяем путь до программы:
    push es
    push bx
    push si
    push ax
    mov es,es:[2ch] ; в es сегментный адрес среды
    mov bx,-1
    SREDA_ZIKL:
        add bx,1
        cmp word ptr es:[bx],0000h
        jne SREDA_ZIKL
    add bx,4
```

```

mov si,-1
PUT_ZIKL:
    add si,1
    mov al,es:[bx+si]
    mov PROGR[si],al
    cmp byte ptr es:[bx+si],00h
    jne PUT_ZIKL

add si,1
PUT_ZIKL2:
    mov PROGR[si],0
    sub si,1
    cmp byte ptr es:[bx+si],'\ '
    jne PUT_ZIKL2
add si,1
mov PROGR[si],'l'
add si,1
mov PROGR[si],'a'
add si,1
mov PROGR[si],'b'
add si,1
mov PROGR[si],'2'
add si,1
mov PROGR[si], '.'
add si,1
mov PROGR[si],'e'
add si,1
mov PROGR[si],'x'
add si,1
mov PROGR[si],'e'
pop ax
pop si
pop bx
pop es

ret
PREP ENDP
;-----
PREP_PAR PROC
    mov ax, es:[2ch]
    mov ARG, ax
    mov ARG+2,es ; Сегментный адрес параметров командной строки(PSP)
    mov ARG+4,80h ; Смещение параметров командной строки
    ret
PREP_PAR ENDP
;-----
START_MOD PROC

    mov ax,ds
    mov es,ax
    mov bx,offset ARG

    mov dx,offset PROGR

    mov KEEP_SS, SS
    mov KEEP_SP, SP

    mov ax,4B00h
    int 21h

```

```

    push ax
    mov ax,DATA
    mov ds,ax
    pop ax
    mov SS,KEEP_SS
    mov SP,KEEP_SP
    jnc START_MOD_skip1
        call PROC_ER
        jmp START_MOD_konec
START_MOD_skip1:

    call ENTER_END

START_MOD_konec:

    ret
START_MOD ENDP
;-----
PROC_ER PROC
    mov dx,offset er
    call PRINT

    mov dx,offset er1
    cmp ax,1
    je osh_pechat
    mov dx,offset er2
    cmp ax,2
    je osh_pechat
    mov dx,offset er7
    cmp ax,7
    je osh_pechat
    mov dx,offset er8
    cmp ax,8
    je osh_pechat
    mov dx,offset er9
    cmp ax,9
    je osh_pechat
    mov dx,offset er10
    cmp ax,10
    je osh_pechat
    mov dx,offset er11
    cmp ax,11
    je osh_pechat

    osh_pechat:
    call PRINT
    mov dx,offset STRENDL
    call PRINT

    ret
PROC_ER ENDP
;-----
ENTER_END PROC
    ; получаем в al код завершения, в ah - причину:
    mov al,00h
    mov ah,4dh
    int 21h

    mov dx, offset end0
    cmp ah, 0
    je ENTER_END_pech_1

```

```

        mov dx,offset end1
        cmp ah,1
        je ENTER_END_pech
        mov dx,offset end2
        cmp ah,2
        je ENTER_END_pech
        mov dx,offset end3
        cmp ah,3
        je ENTER_END_pech

ENTER_END_pech_1:
        call PRINT
        mov dx,offset STRENDL
        call PRINT
        mov dx, offset end_cod

ENTER_END_pech:
        call PRINT

        cmp ah,0
        jne ENTER_END_skip

; печать кода завершения:
        call BYTE_TO_HEX
        push ax
        mov ah,02h
        mov dl,al
        int 21h
        pop ax
        mov dl,ah
        mov ah,02h
        int 21h
        mov dx,offset STRENDL
        call PRINT
ENTER_END_skip:

        ret
ENTER_END ENDP
;-----
BEGIN:
        mov ax,data
        mov ds,ax
        call PREP
        call START_MOD
        xor AL,AL
        mov AH,4Ch
        int 21h
CODE ENDS
; ДАННЫЕ
DATA SEGMENT
        er db 'Ошибка: $'
        er1 db 'Номер функции неверен$'
        er2 db 'Файл не найден$'
        er7 db 'Разрушен управляющий блок памяти$'
        er8 db 'Недостаточный объем памяти$'
        er9 db 'Неверный адрес блока памяти$'
        er10 db 'Неправильная строка среды$'
        er11 db 'Неправильный формат$'

; причины завершения
end0 db 'Нормальное завершение$'
end1 db 'Завершение по Ctrl-Break$'
end2 db 'Завершение по ошибке устройства$'

```



```

end3 db 'Завершение по функции 31h$'
end_cod db 'Код завершения: $'

STRENDL db 0DH,0AH,'$'

; блок параметров
ARG dw 0 ; сегментный адрес среды
      dd 0 ; сегмент и смещение командной строки
      dd 0 ; сегмент и смещение первого FCB
      dd 0 ; сегмент и смещение второго FCB

; путь и имя вызываемой программы
PROGR db 40h dup (0)
; переменные для хранения SS и SP
KEEP_SS dw 0
KEEP_SP dw 0
DATA ENDS
; CTEK
ASTACK SEGMENT STACK
      dw 100h dup (?)
ASTACK ENDS
END START

```