

**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. 7383

\_\_\_\_\_

Прокопенко Н.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2019

**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.

#### **Постановка задачи:**

**Шаг 1.** Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет функции:

1) Подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором находится он сам. Вызываемому модулю передается новая среда, созданная вызывающим модулем и новая командная строка.

2) Вызываемый модуль запускается с использованием загрузчика.

3) После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы. Необходимо проверять причину завершения и, в зависимости от значения, выводить соответствующее сообщение. Если причина завершения 0, то выводится код завершения.

В качестве вызываемой программы необходимо взять программу ЛР 2, которая распечатывает среду и командную строку. Эту программу следует немного модифицировать, вставив перед выходом из нее обращение к функции ввода символа с клавиатуры. Введенное значение записывается в регистр AL и затем происходит обращение к функции выхода 4Ch прерывания int 21h.

**Шаг 2.** Запустите отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры.

Введите произвольный символ из числа A-Z. Посмотрите причину завершения и код. Занесите полученные данные в отчет.

**Шаг 3.** Запустите отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры.

Введите комбинацию символов Ctrl-C. Посмотрите причину завершения и код. Занесите полученные данные в отчет.

**Шаг 4.** Запустите отлаженную программу, когда текущим каталогом является какой-либо другой каталог, отличный от того, в котором содержатся разработанные программные модули.

Повторите ввод комбинаций клавиш. Занесите полученные данные в отчет.

**Шаг 5.** Запустите отлаженную программу, когда модули находятся в разных каталогах. Занесите полученные данные в отчет.

### **Описание функций и структур данных.**

Описание процедур и структур данных представлены в табл. 1 и табл.2 соответственно.

Таблица 1 – описание процедур

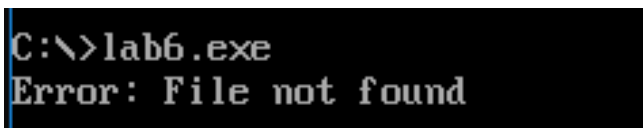
<b>Название процедуры</b>	<b>Назначение</b>
BYTE_TO_HEX	переводит число из AL в 2 16-ых символа и помещает их в AL и BH
PRINT	вызывает функцию печати строки
PREP	выполняет подготовку параметров для запуска загрузочного модуля
PREP_PAR	выполняет создание блок параметров

START_MOD	Окончание таблицы 1 выполняет запуск загрузочного модуля
PROC_ER	функция обработки ошибок

Таблица 2 – описание структур данных

Название поля данных	Тип	Назначение
er	db	Error
er1	db	function Number invalid
er2	db	File not found
er7	db	Destroyed the control unit of memory
er8	db	Insufficient memory
er9	db	Invalid memory block address
er10	db	Wrong environment string
er11	db	Wrong format
end0	db	Normal completion
end1	db	Ctrl-Break completion
end2	db	Termination by device error
end3	db	Completion by function 31h
end_code	db	completion Code

Примеры работы программы по шагам представлены на рисунке 1, рисунке 2, рисунке 3, рисунке 4.



```
C:\>lab6.exe
Error: File not found
```

Рисунок 1 – Запуск программы при условии, что программный и загрузочный модуль находятся в разных каталогах

```

C:\>lab6.exe
Segment address of the first byte of inaccessible memory: 9FFF
Segmental address of the environment passed to the program: 02C6
Command-line tail:
The contents of the environment area in the symbolic form:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Load module path:
C:\LAB2.COM
Z
Normal completion
completion Code: 7A

```

Рисунок 2 – Запуск программы из каталога с разработанными модулями и  
ВВОД СИМВОЛА Z

```

C:\>lab6.exe
Segment address of the first byte of inaccessible memory: 9FFF
Segmental address of the environment passed to the program: 02C6
Command-line tail:
The contents of the environment area in the symbolic form:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Load module path:
C:\LAB2.COM
♥
Normal completion
completion Code: 03

```

Рисунок 3 – Запуск программы из каталога с разработанными модулями с  
ВВОДОМ комбинации Ctrl-C

```

C:\>TASM\lab6.exe
Segment address of the first byte of inaccessible memory: 9FFF
Segmental address of the environment passed to the program: 02C6
Command-line tail:
The contents of the environment area in the symbolic form:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Load module path:
C:\TASM\LAB2.COM
♥
Normal completion
completion Code: 03

```

Рисунок 4 – Запуск программы из внешней директории

## Выводы.

В ходе лабораторной работы был построен загрузочный модуль динамической структуры, а также модифицирован ранее построенный

программный модуль. Изучены дополнительные функции работы с памятью и исследованы возможности использования интерфейса между вызывающим и вызываемым модулями по управлению и по данным.

Код программы lab6\_.ASM представлен в приложении А.

### **Ответы на контрольные вопросы.**

- Как реализовано прерывание Ctrl-C?

При нажатии клавиш Ctrl-C управление передаётся по адресу 0000:008Ch. Этот адрес копируется в PSP функциями 26h и 4Ch и восстанавливается из PSP при выходе из программы.

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

- В какой точке заканчивается вызываемая программа, если код причины завершения 0?

Если код завершения 0, то программа завершается при выполнении функции 4Ch прерывания int 21h.

- В какой точке заканчивается вызываемая программа по прерыванию Ctrl-C?

Если во время выполнения программы было нажато Ctrl-C, то программа завершится непосредственно в том месте, в котором произошло нажатие сочетания клавиш (то есть в месте ожидания нажатия клавиши: 01h вектора прерывания 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], 'c'
    add si, 1
    mov PROGR[si], 'o'
    add si, 1
    mov PROGR[si], 'm'
    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 STRENDL
call PRINT
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_code

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 'Error: $'
    er1   db 'function Number invalid$'
    er2   db 'File not found$'
    er7   db 'Destroyed the control unit of memory$'
    er8   db 'Insufficient memory$'
    er9   db 'Invalid memory block address$'
    er10  db 'Wrong environment string$'
    er11  db 'Wrong format$'

    ; причины завершения
    end0  db 'Normal completion$'
    end1  db 'Ctrl-Break$completion'
    end2  db 'Termination by device error$'
    end3  db 'Completion by function 31h$'
    end_code db 'completion Code: $'

    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 STARTin
```