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

Студент гр. 7383 _____ Корякин М.П.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры. В отличии 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. Понятно, что такое приложение должно запускаться в соответствии со стандартами ОС.

Описание функций.

- BYTE_TO_HEX - переводит число из AL в 2 16-ых символа и помещает их в AL и BH.
- PRINT - вызывает функцию печати строки.
- PODG - выполняет подготовку параметров для запуска загрузочного модуля.
- PODG_PAR - выполняет создание блок параметров.
- ZAP_MOD - выполняет запуск загрузочного модуля.
- OBR_OSH - функция обработки ошибок.
- PROV_ZAV - функция вывода причины и кода завершения загрузочного модуля.

Результат работ написанной программы.

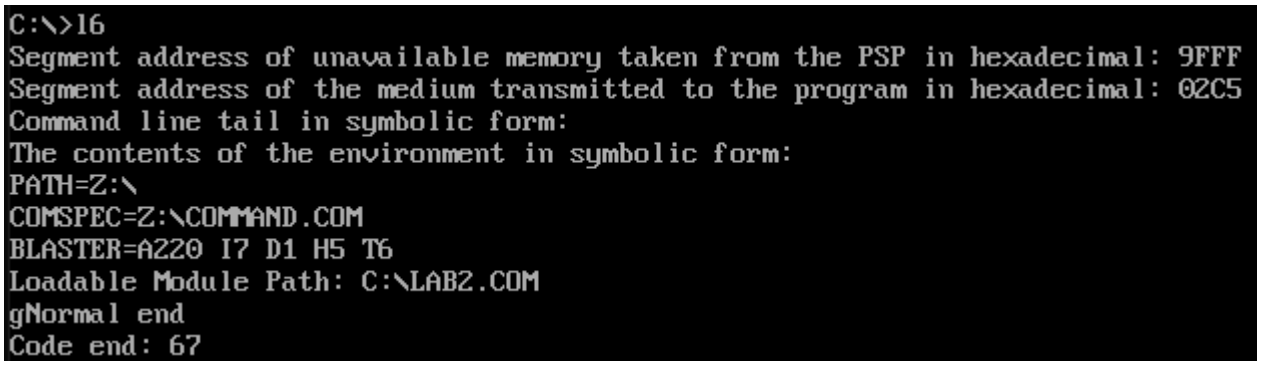

Рисунок 1 – Запуск программы при нахождении модулей в текущем каталоге с вводом символа «g».

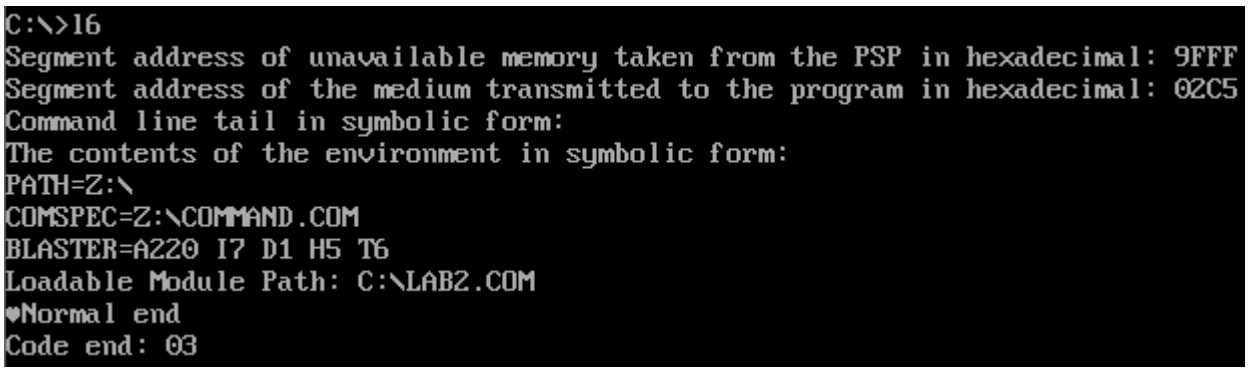

Рисунок 2 – Запуск программы при нахождении модулей в текущем каталоге с вводом сочетания клавиш «Ctrl+C».

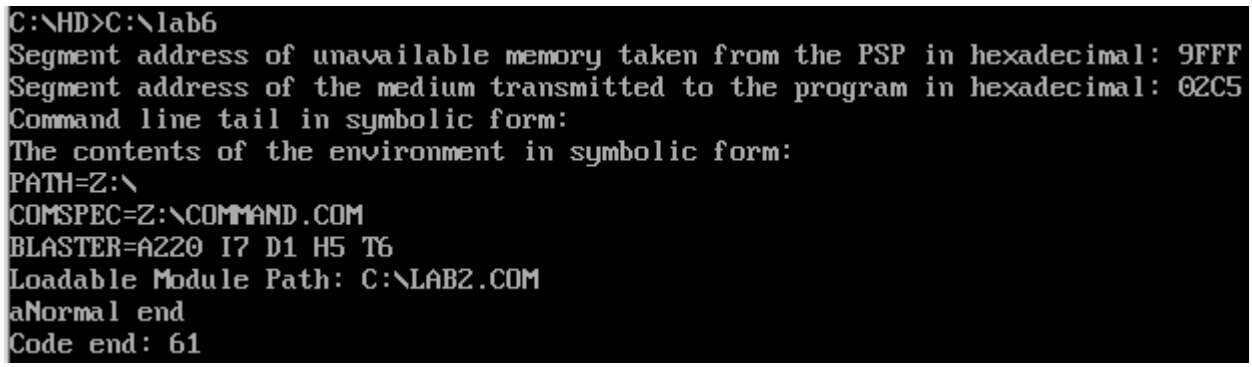

Рисунок 3 – Запуск программы из каталога, в котором не находятся модули с вводом клавиши «a».

Рисунок 4 – Запуск программы при нахождении модулей в разных каталогах.

Выводы.

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

Ответы на контрольные вопросы.

1. Как реализовано прерывание Ctrl-C?

При нажатие данного сочетания клавиш вызывается прерывание 23h, находящееся по адресу 0000:008Ch, и управление передается адресу, находящемуся в вышеописанном векторе прерывания.

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

   Вызываемая программа заканчивается в точке вызова функции 4Ch прерывания int 21h.

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

   Вызываемая программа заканчивается в точке вызова функции 01h прерывания int 21h.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
CODE SEGMENT
 ASSUME CS:CODE, DS:DATA, ES:DATA, SS:ASTACK
START: JMP BEGIN
;--------------------------------------
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
;--------------------------------------
PODG PROC
   mov ax,ASTACK
   sub ax,CODE
   add ax,100h
   mov bx,ax
   mov ah,4ah
   int 21h
   jnc podg_skip1
         call OBR_OSH
podg_skip1:

   call PODG_PAR
```

```
   push es
   push bx
   push si
   push ax
   mov es,es:[2ch]
   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
PODG ENDP
;---------------------------------------
PODG_PAR PROC
   mov ax, es:[2ch]
   mov PARAM, ax
```

```
   mov PARAM+2,es
   mov PARAM+4,80h
   ret
PODG_PAR ENDP
;---------------------------------------
ZAP_MOD PROC
   mov ax,ds
   mov es,ax
   mov bx,offset PARAM

   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 zap_mod_skip1
        call OBR_OSH
        jmp zap_mod_konec
   zap_mod_skip1:

   call PROV_ZAV

   zap_mod_konec:

   ret
ZAP_MOD ENDP
;---------------------------------------
OBR_OSH PROC
   mov dx,offset o
   call PRINT

   mov dx,offset o1
   cmp ax,1
   je osh_pechat
   mov dx,offset o2
   cmp ax,2
   je osh_pechat
   mov dx,offset o7
   cmp ax,7
   je osh_pechat
   mov dx,offset o8
   cmp ax,8
```

```
   je osh_pechat
   mov dx,offset o9
   cmp ax,9
   je osh_pechat
   mov dx,offset o10
   cmp ax,10
   je osh_pechat
   mov dx,offset o11
   cmp ax,11
   je osh_pechat
   osh_pechat:
   call PRINT
   mov dx,offset STRENDL
   call PRINT
   ret
OBR_OSH ENDP
;--------------------------------------
PROV_ZAV PROC
   mov al,00h
   mov ah,4dh
   int 21h
   mov dx, offset z0
   cmp ah, 0
   je prov_zav_pech_1
   mov dx,offset z1
   cmp ah,1
   je prov_zav_pech
   mov dx,offset z2
   cmp ah,2
   je prov_zav_pech
   mov dx,offset z3
   cmp ah,3
   je prov_zav_pech
   prov_zav_pech_1:
   call PRINT
   mov dx,offset STRENDL
   call PRINT
   mov dx, offset z
   prov_zav_pech:
   call PRINT
   cmp ah,0
   jne prov_zav_skip
   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
   prov_zav_skip:
   ret
PROV_ZAV ENDP
;--------------------------------------
BEGIN:
   mov ax,data
   mov ds,ax
   call PODG
   call ZAP_MOD
   xor AL,AL
   mov AH,4Ch
   int 21H
CODE ENDS
DATA SEGMENT
   o db 'Mistake: $'
   o1 db 'False number of function$'
   o2 db 'File not found$'
   o7 db 'Control block of mem is broken$'
   o8 db 'Insufficient memory$'
   o9 db 'Address block of mem is false$'
   o10 db 'Environment string is false$'
   o11 db 'False format$'

   z0 db 'Normal end$'
   z1 db 'End of Ctrl-Break$'
   z2 db 'End of mistake device $'
   z3 db 'End of function 31h$'
   z db 'Code end: $'
   STRENDL db 0DH,0AH,'$'

   PARAM      dw 0
              dd 0
              dd 0
              dd 0
   PROGR db 40h dup (0)
   KEEP_SS dw 0
   KEEP_SP dw 0
DATA ENDS
ASTACK SEGMENT STACK
   dw 100h dup (?)
ASTACK ENDS
 END START
```