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

ОТЧЕТ

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

Студентка гр. 8381 _____ Ивлева О.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Исследование возможности встраивания пользовательского обработчика прерываний в стандартный обработчик от клавиатуры.

Выполнение работы.

Сборка, отладка производились на базе эмулятора DOSBox 0.74-3.

Был написан текст исходного .EXE модуля с именем lab6.EXE. Описание процедур в программе представлено в табл. 1.

Таблица 1 – Описание процедур программы

Название	**Назначение**
Main	Основная процедура
PEXIT	Процедура обработки нажатия символа и последующего завершения программы
STRING	Вывод на экран строки, адрес которой содержится в DX
FREEMEM	Процедура освобождения лишней зарезервированной программой памяти

Программа выполняет освобождение лишней памяти, загружает дочерний модуль, если файл удалось найти файл в папке с лабораторной работой. Во время выполнения дочернего модуля считывается символ, нажатый пользователем, управление переходит основному модулю. Проверяется код выхода из дочернего модуля и выводится сообщение с кодом.

В модуле lab2.COM выполняется считывание символа с клавиатуры в конце программы.

Для завершения работы была нажата клавиша «a». При завершении был выведен ее код 61. Выполнение программы представлено на рис. 1.

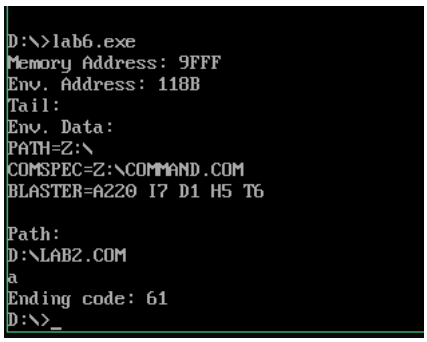

Рисунок 1 – Результат выполнения с модулем lab2.COM

Вывод программы при нажатии ctrl+c, представлен на рис. 2.

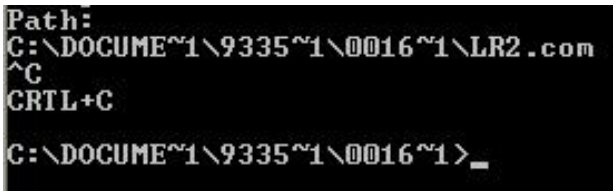

Рисунок 2 – Вывод программы при нажатии ctrl+c

Далее программа была запущена в другой папке, где нет lab2.COM. Результат выполнения представлен на рис. 3.

Рисунок 3 –Вывод программы без lab2.COM

Далее программа была запущена, когда два модуля находятся в разных каталогах, результат выполнения представлен на рис. 4.

Рисунок 4 – Два модуля находятся в разных каталогах

Контрольные вопросы.

1. Как реализовано прерывание Ctrl-C?

Если нажата комбинация клавиш Ctrl-C, вызывается прерывание INT 23H.

При нажатии Ctrl-C управление переходит по адресу в векторе (0000:008c). Адрес по вектору INT 23H копируется в поля PSP функциями DOS 26H (создание PSP) и 4cH (EXEC). Исходное значение адреса обработчика Ctrl-C восстанавливается из PSP при завершении программы.

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

Если код причины завершения 0, то вызываемая программа заканчивается стандартно, то есть в месте вызова функции 4Ch прерывания int 21h.

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

В точке вызова функции 01h прерывания int 21h, где программа ожидала ввод с клавиатуры, а получила нажатую комбинацию клавиш Ctrl-C.

Выводы.

В ходе выполнения данной лабораторной работы была исследована работа и организация загрузочных модулей динамической структуры. Были приобретены навыки по загрузке и завершению дочерних модулей.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR6.ASM

```
.model small
.data

PSP               dw    ?
SPSAVE                  dw    ?
SSSAVE                  dw    ?

SUCCESS_INFO      db    13, 10, "Ending code: $"
NOFILE_INFO       db  "No file", 13, 10, "$"
CTRLC_INFO        db    "CRTL+C", 13, 10, "$"

PARAMETERS        dw    7 dup(?)
ERROR                   db    0
FILE              db    50 dup(0)
EOF                     db    "$"

.stack 128h

.code

HEX_BYTE_PRINT    PROC
            push  AX
            push  BX
            push  DX

            mov   AH, 0
            mov   BL, 10h
            div   BL
            mov   DX, AX
            mov   AH, 02h
            cmp   DL, 0Ah
            jl          PRINT
            add   DL, 07h
      PRINT:
            add   DL, '0'
            int   21h;

            mov   DL, DH
            cmp   DL, 0Ah
            jl          PRINT_EXT
            add   DL, 07h
      PRINT_EXT:
            add   DL, '0'
            int   21h;
```

```
            pop   DX
            pop   BX
            pop   AX
      ret
HEX_BYTE_PRINT    ENDP

WRITE       PROC  near
            push  AX
            mov   AH, 09h
            int         21h
            pop   AX
            ret
WRITE       ENDP

FREEMEM     PROC
            lea   BX, MARK
            mov   AX, ES
            sub   BX, AX
            mov   CL, 4
            shr   BX, CL
            mov   AH, 4Ah
            int   21h
            jc          CATCH
            jmp   DEFAULT1
      CATCH:
            mov   ERROR, 1
      DEFAULT1:
            ret
FREEMEM     ENDP

PEXIT       PROC
            mov   AH, 4Dh
            int   21h
            cmp   AH, 1
            je          CTRLC
            lea   DX, SUCCESS_INFO
            call  WRITE
            call  HEX_BYTE_PRINT
            mov   DL, AH
            mov   AH, 2h
            int   21h
            jmp   DEFAULT2
      CTRLC:
            lea   DX, CTRLC_INFO
            call  WRITE
      DEFAULT2:
```

```
      ret
PEXIT       ENDP

Main proc
            mov   AX, @data
            mov   DS, AX
            push  SI
            push  DI
            push  ES
            push  DX
            mov   ES, ES:[2Ch]
            xor   SI, SI
            lea   DI, FILE
      CHAR:
            cmp   byte ptr ES:[SI], 00h
            je          CHAREND
            inc   SI
            jmp   NEXT1
      CHAREND:
            inc   SI
      NEXT1:
            cmp   word ptr ES:[SI], 0000h
            jne   CHAR
            add   SI, 4
      NCHAR:
            cmp   byte ptr ES:[SI], 00h
            je          START
            mov   DL, ES:[SI]
            mov   [DI], DL
            inc   SI
            inc   DI
            jmp   NCHAR
      START:
            sub   DI, 5
            mov   DL, '2'
            mov   [DI], DL
            add   DI, 2
            mov   DL, 'c'
            mov   [DI], DL
            inc   DI
            mov   DL, 'o'
            mov   [DI], DL
            inc   DI
            mov   DL, 'm'
            mov   [DI], DL
            inc   DI
            mov   DL, 0h
```

```
            mov   [DI], DL
            inc   DI
            mov   DL, EOF
            mov   [DI], DL
            pop   DX
            pop   ES
            pop   DI
            pop   SI
            call  FREEMEM
            cmp   ERROR, 0
            jne   PDEFAULT
            push  DS
            pop   ES
            lea   DX, FILE
            lea   BX, PARAMETERS
            mov   SSSAVE, SS
            mov   SPSAVE, SP
            mov   AX, 4B00h
            int   21h
            mov   SS, SSSAVE
            mov   SP, SPSAVE
            jc          NOFILE
            jmp   MENDING
      NOFILE:
            lea   DX, NOFILE_INFO
            call  WRITE
            lea   DX, FILE
            call  WRITE
            jmp   PDEFAULT
      MENDING:
            call  PEXIT
      PDEFAULT:
            mov   AH, 4Ch
            int   21h
main ENDP

MARK:

end main
```