**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**

Студент гр. 8381 _____ Облизанов А.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

**Цель работы.**

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

**Выполнение работы.**

Написание работы производилось на базе операционной системы Windows 10 в редакторе Visual Code. Сборка, отладка производились на базе эмулятора DOSBox 0.74-3.

Был написан текст исходного .EXE модуля с именем LR6.EXE. Описание процедур в программе представлено в табл. 1.

Таблица 1 – Описание процедур программы

| **Название** | **Назначение** |
|---|---|
| PRINT_STRING | Вывод на экран строки, адрес которой содержится в DX |
| MEMORY_FREE | Процедура освобождения лишней зарезервированной программой памяти |
| EXIT_PROGRAM | Процедура обработки нажатия символа и последующего завершения программы |
| Main | Основная процедура |

Основные этапы работы программы:

- Освобождение лишней выделенной памяти
- Загрузка дочернего модуля, если файл был найден в текущей директории
- Во время выполнения дочернего модуля считывается символ, нажатый пользователем, управление переходит основному модулю
- Проверяется код выхода из дочернего модуля, выводится соответствующее сообщение

Модуль LR2.COM был доработан таким образом, чтобы в конце его выполнения считывался символ с клавиатуры.

Результат запуска отлаженной программы, когда текущим каталогом является каталог с модулем LR2.COM, представлен на рис. 1.

```
C:\MASM>lr6.exe
Inaccessible memory adress:  9FFF
Program environment adress:  1508
Command line tail:
No command line tail
Program environment content:
COMSPEC=C:\WINDOWS\SYSTEM32\COMMAND.COM
 ALLUSERSPROFILE=C:\DOCUME~1\ALLUSE~1
 APPDATA=C:\Documents and Settings\Администратор\Application Data
 CLIENTNAME=Console
 COMMONPROGRAMFILES=C:\PROGRA~1\COMMON~1
 COMPUTERNAME=PAPA-DEB47E265C
 FP_NO_HOST_CHECK=NO
 HOMEDRIVE=C:
 HOMEPATH=\Documents and Settings\Администратор
 LOGONSERVER=\\PAPA-DEB47E265C
 NUMBER_OF_PROCESSORS=2
 OS=Windows_NT
 PATH=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
 PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
 PROCESSOR_ARCHITECTURE=x86
 PROCESSOR_IDENTIFIER=x86 Family 6 Model 142 Stepping 9, GenuineIntel
 PROCESSOR_LEVEL=6
 PROCESSOR_REVISION=8e09
 PROGRAMFILES=C:\PROGRA~1
 PROMPT=$P$G
 SESSIONNAME=Console
 SYSTEMDRIVE=C:
 SYSTEMROOT=C:\WINDOWS
 TEMP=C:\WINDOWS\TEMP
 TMP=C:\WINDOWS\TEMP
 USERDOMAIN=PAPA-DEB47E265C
 USERNAME=Администратор
 USERPROFILE=C:\Documents and Settings\Администратор
 BLASTER=A220 I5 D1 P330 T3

 Program environment content ended
Path:
C:\MASM\LR2.COM
q
Program ended with code 0
C:\MASM>
```

Рисунок 1 – Результат выполнения с модулем LR2.COM

Для выхода из программы была нажата клавиша “q”. Как видно, в конце программы отобразилась нажатая клавиша, а также сообщение о выходе из программы в нормальном режиме (код 0).

Вывод программы, если для выхода было нажато сочетание клавиш CTRL+C, представлен на рис. 2. Это отразилось в коде выхода из модуля, что отлавливается программой и выводится соответствующее сообщение.

```
C:\MASM>lr6.exe
Inaccessible memory adress:  9FFF
Program environment adress:  1508
Command line tail:
No command line tail
Program environment content:
COMSPEC=C:\WINDOWS\SYSTEM32\COMMAND.COM
 ALLUSERSPROFILE=C:\DOCUME~1\ALLUSE~1
 APPDATA=C:\Documents and Settings\Администратор\Application Data
 CLIENTNAME=Console
 COMMONPROGRAMFILES=C:\PROGRA~1\COMMON~1
 COMPUTERNAME=PAPA-DEB47E265C
 FP_NO_HOST_CHECK=NO
 HOMEDRIVE=C:
 HOMEPATH=\Documents and Settings\Администратор
 LOGONSERVER=\\PAPA-DEB47E265C
 NUMBER_OF_PROCESSORS=2
 OS=Windows_NT
 PATH=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem
 PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
 PROCESSOR_ARCHITECTURE=x86
 PROCESSOR_IDENTIFIER=x86 Family 6 Model 142 Stepping 9, GenuineIntel
 PROCESSOR_LEVEL=6
 PROCESSOR_REVISION=8e09
 PROGRAMFILES=C:\PROGRA~1
 PROMPT=$P$G
 SESSIONNAME=Console
 SYSTEMDRIVE=C:
 SYSTEMROOT=C:\WINDOWS
 TEMP=C:\WINDOWS\TEMP
 TMP=C:\WINDOWS\TEMP
 USERDOMAIN=PAPA-DEB47E265C
 USERNAME=Администратор
 USERPROFILE=C:\Documents and Settings\Администратор
 BLASTER=A220 I5 D1 P330 T3

 Program environment content ended
Path:
C:\MASM\LR2.COM
^C
Program ended by CRTL+C command

C:\MASM>
```

Рисунок 2 – Результат выполнения с LR2.COM и CTRL+C

Далее программа была запущена в другом каталоге, в котором нет модуля LR2.COM. Результат выполнения представлен на рис. 3.

```
C:\MASM>lr6.exe
File was not founded
C:\MASM\LR2.com
C:\MASM>
```

Рисунок 3 – Результат выполнения в каталоге без LR2.COM

В случае, когда два модуля находятся в разных каталогах, результат выполнения аналогичен рис. 3.

**Контрольные вопросы.**

**1. Как реализовано прерывание Ctrl-C?**

DOS вызывает INT 23H, когда распознает, что нажата комбинация Ctrl-Break. Уровень чувствительности DOS к Ctrl-Break может быть проверен или установлен посредством функции 33H:

- Если Break=ON, DOS распознает Ctrl-Break в течение всех функций, за исключением 06H и 07H.
- Если Break=OFF, DOS распознает Ctrl-Break лишь во время операций ввода-вывода с консолью, принтером и последовательными портами.

Адрес в этом векторе (0000:008c) – адрес, по которому передается управление, когда DOS распознает, что пользователь нажал Ctrl-Break (Ctrl-C). Адрес по вектору INT 23H копируется в поле PSP Ctrl-Break Address функциями DOS 26H (создать PSP) и 4cH (EXEC). Исходное значение адреса обработчика Ctrl-Break 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 в месте вызова функции 4Ch прерываний 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 (и была получена комбинация 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

STR_SUCCESS       db    13, 10, "Program ended with code $"
STR_NOFILENAME          db  "File was not founded", 13, 10, "$"
STR_CTRLC         db    "Program ended by CRTL+C command", 13, 10, "$"

PSP               dw    ?
SS_BACK           dw    ?
SP_BACK           dw    ?
FILENAME                db    50 dup(0)
ENDOFLINE         db    "$"
PARAM                   dw    7 dup(?)
MEMORY_ERROR      db    0

.stack 100h

.code

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         al, ah
            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_STRING      PROC  near
            push  AX
            mov   AH, 09h
```

```
            int         21h
            pop   AX
      ret
PRINT_STRING      ENDP

MEMORY_FREE       PROC
            lea   BX, PROGEND
            mov   AX, ES
            sub   BX, AX
            mov   CL, 4
            shr   BX, CL
            mov   AH, 4Ah
            int   21h
            jc          MCATCH
            jmp   MDEFAULT
      MCATCH:
            mov   MEMORY_ERROR, 1
      MDEFAULT:
      ret
MEMORY_FREE       ENDP

EXIT_PROGRAM      PROC
            mov   AH, 4Dh
            int   21h
            cmp   AH, 1
            je          ECTRLC
            lea   DX, STR_SUCCESS
            call  PRINT_STRING
            add         AH, '0'
            mov   DL, AH
            mov   AH, 2h
            int   21h
            jmp   EDEFAULT
      ECTRLC:
            lea   DX, STR_CTRLC
            call  PRINT_STRING
      EDEFAULT:
      ret
EXIT_PROGRAM      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, FILENAME
ECHAR:
      cmp   byte ptr ES:[SI], 00h
      je          ECHAREND
      inc   SI
      jmp   ENEXT
ECHAREND:
      inc   SI
ENEXT:
      cmp   word ptr ES:[SI], 0000h
      jne   ECHAR
      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, ENDOFLINE
      mov   [DI], DL
      pop   DX
      pop   ES
      pop   DI
      pop   SI
      call  MEMORY_FREE
      cmp   MEMORY_ERROR, 0
```

```
            jne   PDEFAULT
            push  DS
            pop   ES
            lea   DX, FILENAME
            lea   BX, param
            mov   SS_BACK, SS
            mov   SP_BACK, SP
            mov   AX, 4B00h
            int   21h
            mov   SS, SS_BACK
            mov   SP, SP_BACK
            jc          NOFILENAME
            jmp   MENDING
      NOFILENAME:
            lea   DX, STR_NOFILENAME
            call  PRINT_STRING
            lea   DX, FILENAME
            call  PRINT_STRING
            jmp   PDEFAULT
      MENDING:
            call  EXIT_PROGRAM
      PDEFAULT:
            mov   AH, 4Ch
            int   21h
main ENDP

PROGEND:

end main
```