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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студентка гр. 8383 _____ Максимова А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системные данные. Так загрузчик строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Основные теоретические положения.

При начальной загрузке программы формируется PSP, который размещается в начале первого сегмента программы. PSP занимает 256 байт и располагается с адреса, кратного границе сегмента. При загрузке модулей типа **.COM** все сегментные регистры указывают на адрес PSP. При загрузке модуля типа **.EXE** сегментные регистры DS и ES указывают на PSP. Именно по этой причине значения этих регистров в модуле **.EXE** следует переопределять.

Область среды содержит последовательность символьных строк вида:

имя=параметр

Каждая строка завершается байтом нулей.

В первой строке указывается имя COMSPEC, которая определяет используемый командный процессор и путь к COMMAND.COM. Следующие строки содержат информацию, задаваемую командами PATH, PROMT, SET.

Среда заканчивается также байтом нулей. Таким образом, два нулевых байта являются признаком конца переменных среды. Затем идут два байта, содержащих 00h, 01h, после которых располагается маршрут загруженной программы. Маршрут также заканчивается байтом 00h.

Offset (Смещение)	Size (Размер)	Contents (Содержание)
00-01	2 байта (код)	Содержит код INT 20 выхода из программы в стиле CP/M (для совместимости)
02-03	машинное слово (2 байта)	Сегмент, расположенный сразу после выделенной программе памяти
04	байт	Зарезервировано
05-09	5 байтов (код)	Содержит код CALL FAR для вызова функций DOS в стиле CP/M (для совместимости)
0A-0D	dword (4 байта)	Адрес обработчика Terminate предыдущей программы (предыдущий INT 22)
0E-11	dword	Адрес обработчика Break предыдущей программы (предыдущий INT 23)
12-15	dword	Адрес обработчика критических ошибок предыдущей программы (предыдущий INT 24)
16-17	машинное слово	Сегмент PSP вызывающего процесса (как правило, command.com — внутренний)
18-2B	20 байт	en:Job File Table (внутренняя)
2C-2D	машинное слово	Сегмент переменных среды
2E-31	dword	SS:SP на входе к последнему вызову INT 21 (внутренний)
32-33	машинное слово	максимальное количество открытых файлов (внутренний — см. ниже)
34-37	dword	Адрес ручных записей (внутренний — см. ниже)
38-4F	24 байта	Зарезервировано
50-52	3 байта (код)	Для вызова к DOS (всегда содержит INT 21 + RETF)
53-5B	9 байт	Зарезервировано
5C-6B	16 байт	Закрытый уровень FCB 1
6C-7F	20 байт	Закрытый уровень FCB (перезаписан, если FCB 1 открыт)
80	1 байт	Количество символов в командной строке
81-FF	127 байт	Командная строка (завершается 0Dh)

Рисунок 1 - формат PSP

Процедуры, используемые в программе

Название:	Предназначение:
TETR_TO_HEX	Процедура перевода тетрады в шестнадцатеричную цифру (символ)
BYTE_TO_HEX	Процедура перевода байта AL в два символа шестнадцатеричного числа
WRD_TO_HEX	Процедура перевода слова в

	шестнадцатеричную систему счисления
PRINTF	Процедура печати
GET_ADDRESS_PR	Процедура получения и печати сегментных адресов недоступной памяти и среды в шестнадцатеричном виде
GET_TAIL_PR	Процедура печати хвоста командной строки
GET_NUMBER_CHAR	Процедура получения числа символов в хвосте командной строки, печатающая предупреждение, если количество символов равно нулю, или обращающаяся к процедуре печати содержимого хвоста при неравенстве нулю
GET_CONTENT_AREA	Процедура для вывода содержимого области среды и пути загружаемого модуля в символьном виде

Выполнение работы

В файле LR2.ASM был написан текст исходного модуля типа **.COM,** выбирающего и рассчитывающего следующую информацию:

1. Сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде.
2. Сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.
3. Хвост командной строки в символьном виде.
4. Содержимое области среды в символьном виде.
5. Путь загружаемого модуля.

Тестирование.

Рисунок 2 - Запуск программы

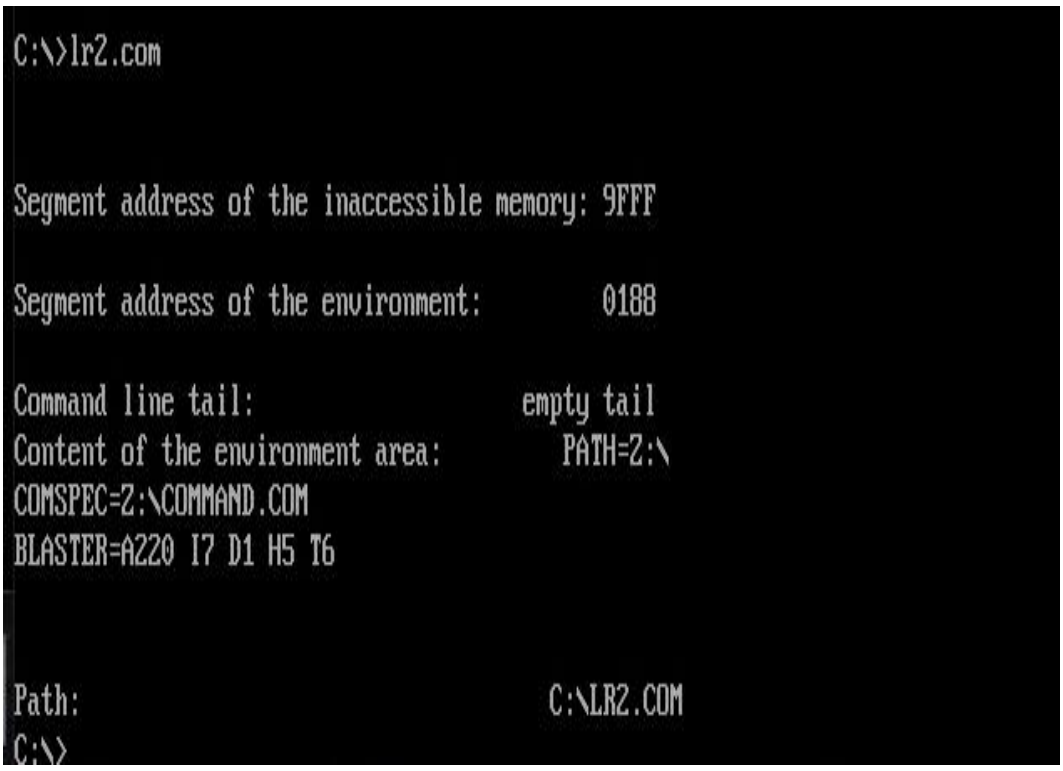

Рисунок 3 - Результат работы программы, входные данные отсутствуют

Рисунок 4 - Результат работы программы, входные данные есть

Ответы на контрольные вопросы

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на границу основной оперативной памяти (стандартной), располагаемой до сегментного адреса 9FFFh, со старшей памятью, зарезервированной DOS. Более подробное распределение адресного пространства см. на рис. 5.

Рисунок 5 - Распределение адресного пространства

2. Где расположен этот адрес по отношению области памяти, отведенной программе?

Как видно из рисунка 5, этот адрес располагается за областью памяти, отведенной под пользовательскую программу.

3. Можно ли в эту область памяти писать?

Да, можно, потому что DOS не отслеживает обращение программ к памяти, и соответственно ее защита не предусмотрена.

Среда передаваемая программе

1. Что такое среда?

Это последовательность символьных строк, каждая из которых завершается байтом нулей, вида *имя = параметр,* где *имя* и *параметр* - это символьные величины.

2. Когда создается среда? Перед запуском приложения или в другое время?

Корневая среда создается при загрузке ОС. Запускаемая после пользовательская программа получают копию этой среды, созданную по умолчанию, которая в случае необходимости, может быть изменена.

3. Откуда берется информация, записываемая в среду?

Информация, записываемая в среду, берется из системного пакетного файла AUTOEXEC.BAT, устанавливающего ключевые переменные окружения, такие как PATH (переменная, представляющая собой набор каталогов, в которых хранятся исполняемые файлы).

Выводы.

В результате выполнения лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей, а так же интерфейс префикса сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А

Содержимое файлы LR2.ASM

```
COMMENT @
Максимова Анастасия, группа 8383, 2 лабораторная
@

CODE          SEGMENT
              ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING
              ORG 100H

START:   JMP        MAIN

EOF                 EQU        '$'
SETPRECISION  EQU        50

;ДАННЫЕ
ADDRESS_MEMORY                      DB         0DH, 0AH, 0AH, 'Segment
   address of the inaccessible memory:          ', EOF
ADDRESS_ENVIRONMENT           DB         0DH, 0AH, 0AH, 'Segment address of
   the environment:                  ', EOF
TAIL_STRING                         DB         0DH, 0AH, 0AH, 'Command line
   tail:                    ', EOF
WHERE_MY_TAIL                 DB         'empty tail', EOF
CONTENT_AREA                        DB         0DH, 0AH, 'Content of the
   environment area:         ', EOF
ENDL                                DB         0DH, 0AH, EOF
WAY                                 DB         0DH, 0AH, 'Path:
   ', EOF

;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
         ;байт в AL переводится в два символа шестн. числа в AX

                    push    CX
                    mov     AH, AL
                    call    TETR_TO_HEX
                    xchg    AL, AH
                    mov     CL, 4
                    shr     AL, CL
                    call    TETR_TO_HEX
               ;в AL - старшая цифра
                    pop     CX
               ;в AH - младшая
                    ret
BYTE_TO_HEX         ENDP
;--------------------------------------------------------
WRD_TO_HEX          PROC  NEAR
         ;перевод в 16 с/с 16-ти разрядного числа

                    ;в AX - число, DI - адрес последнего символа
                    push  BX
```

```
                    mov         BH, AH
                    call  BYTE_TO_HEX
                    mov         [DI], AH
                    dec         DI
                    mov         [DI], AL
                    dec         DI
                    mov         AL, BH
                    call  BYTE_TO_HEX
                    mov         [DI], AH
                    dec         DI
                    mov         [DI], AL
                    pop         BX
                    ret
WRD_TO_HEX          ENDP
;--------------------------------------------------------
PRINTF         PROC  NEAR
          push    AX
          mov         AH, 09h
         int    21h
          pop      AX
          retn
PRINTF         ENDP
;--------------------------------------------------------
GET_ADDRESS_PR           PROC  NEAR
                    push  AX
                    push  DI
                    push  DX

                    ;Первое - сегментный адрес недоступной памяти, взятый из
   PSP
                    mov   AX, DS:[0002h]
                    mov     DI, offset ADDRESS_MEMORY
                    add         DI, SETPRECISION
                    call  WRD_TO_HEX
```

```
                    mov         DX, offset ADDRESS_MEMORY
                    call  PRINTF

                    sub     AX, AX
                    sub     DI, DI
                    sub     DX, DX

                    ;Второе - сегментный адрес среды, передаваемый программ
                    mov   AX, DS:[002Ch]
                    mov     DI, offset ADDRESS_ENVIRONMENT
                    add         DI, SETPRECISION
                    call  WRD_TO_HEX

                    mov         DX, offset ADDRESS_ENVIRONMENT
                    call  PRINTF

                    pop         DX
                    pop   DI
                    pop         AX

                    retn

GET_ADDRESS_PR          ENDP
;--------------------------------------------------------
GET_TAIL_PR         PROC  NEAR ;если хвост не пустой - выводим его
              push    DI
              push    AX
              push    DX

              sub         DI, DI
              sub         AX, AX

REPEAT:
```

```
                mov         AL, DS:[0081h + DI]

                push    AX
                sub     DX, DX

                mov     DL, AL
                mov     AH, 02h
                int     21h

                pop     AX

                inc     DI
                loop    REPEAT

                pop     DX
                pop     AX
                pop     DI
                retn
GET_TAIL_PR         ENDP
;-------------------------------------------------------
;Третье - кол-во символов в хвосте, если 0, то печатаем предупреждение
GET_NUMBER_CHAR     PROC  NEAR
                push  CX
                push    DX

                mov         DX, offset TAIL_STRING
                call  PRINTF

                sub   CX, CX
                mov     CL, DS:[0080h] ;количество символов в хвосте

                cmp     CL, 00h
                je          EMPTY
                call  GET_TAIL_PR
```

```
                jmp     EXIT


EMPTY:          mov         DX, offset WHERE_MY_TAIL
                call  PRINTF

EXIT:
                pop     DX
                pop         CX

                retn
GET_NUMBER_CHAR     ENDP
;-------------------------------------------------------------
;четвертое - пятое - содержимое области среды и путь
GET_CONTENT_AREA    PROC  NEAR
                push  AX
                push  DX
                push  DI

                mov         DX, offset CONTENT_AREA
                call  PRINTF

                sub   DI, DI
                sub     AX, AX

                mov ES, DS:[002Ch]

CICLE_READ:
                mov         AL, ES:[DI]
                cmp     AL, 00h             ;первый нуль
                je          NEW_STRING      ;печатаем новую строку

                push    AX
                sub     DX, DX
```

```
                mov     DL, AL
                mov     AH, 02h             ;печатаем символ
                int     21h
                pop     AX

                inc     DI
                jmp     CICLE_READ

NEW_STRING:
                mov        DX, offset ENDL
                call  PRINTF
                inc     DI

                mov        AL, ES:[DI]
                cmp     AL, 00h       ;второй нуль
                jne     CICLE_READ

                mov        DX, offset ENDL
                call  PRINTF

FIND_PATH:      inc DI
                mov        AL, ES:[DI]
                cmp     AL, 01h
                jne     FIND_PATH
                add     DI, 2

                mov        DX, offset WAY
                call  PRINTF

COUT_PATH:      cmp     AL, 00h
                je      BYE

                mov        AL, ES:[DI]
                push    AX
```

```
                sub     DX, DX
                mov     DL, AL
                mov     AH, 02h       ;печатаем символ
                int     21h
                pop     AX

                inc     DI
                jmp     COUT_PATH

BYE:
                pop   DI
                pop   DX
                pop   AX

                retn
GET_CONTENT_AREA    ENDP
;-------------------------------------------------------------

MAIN:
          call  GET_ADDRESS_PR        ; 1 и 2 задания
          call  GET_NUMBER_CHAR       ; 3 задание
          call  GET_CONTENT_AREA; 4 и 5 задание

;выход в DOS
          sub         AL, AL
          mov         AH, 4Ch
          int         21h

CODE      ENDS
          END   START           ;конец модуля, START - точка входа
```