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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Операционные системы»
Тема: «Исследование интерфейсов программных модулей»

Студент гр. 7381 _____ Трушников А.П.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2019

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.

Тип IBM PC узнается путем считывания предпоследнего байта с ROM BIOS. Его значение сравнивается с кодами таблицы, представленной ниже.

Смещение	Длина поля(байт)	Содержимое поля
0	2	int 20h
2	2	Сегментный адрес первого байта недоступной памяти. Программа не должна модифицировать содержимое памяти за этим адресом.
4	6	Зарезервировано
0Ah(10)	4	Вектор прерывания 22h
0Eh(14)	4	Вектор прерывания 23h
12h(18)	4	Вектор прерывания 24h
2Ch(44)	2	Сегментный адрес среды, передаваемой программе.
5Ch		Область форматируется как стандартный неоткрытый блок управления файлом
6Ch		Область форматируется как стандартный неоткрытый блок управления файлом
80h	1	Число символов в хвосте командной строки
81h		Хвост командной строки – последовательность символов после имени вызываемого модуля

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, PROMPT, 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.

Выполнение работы.

Результат работы программы:

Ответы на вопросы.

Сегментный адрес недоступной памяти:

1. Адрес недоступной памяти указывает на конечную границу основной оперативной памяти, а также на границу области, доступной для загрузки программ.
2. Адрес недоступной памяти расположен дальше области памяти, отведённой программе.
3. Теоретически да, так как данная память не защищена от записи.

Среда, передаваемая программе:

1. Среда – область памяти, содержащая переменные среды, записанные в ней в виде строк формата “*имя=параметр*”.
2. Среда создаётся при загрузке в DOS, а не перед запуском приложения.
3. Информация, записываемая в среду, берётся из системного файла «autoexec.bat».

Выводы.

В ходе лабораторной работы были исследован интерфейс управляющей программы и загрузочного модуля, а также префикс сегмента программы (PSP) и среды, передаваемой программе. Написана программа, выводящая сегментный адрес недоступной памяти, сегментный адрес среды, передаваемой в программе, в шестнадцатеричном виде, хвост командной строки в символьном виде, содержимое области среды в символьном виде, путь загружаемого модуля.

ПРИЛОЖЕНИЕ А

Исходный код программы

```
TESTPC          SEGMENT
                ASSUME      CS:TESTPC,  DS:TESTPC,  ES:NOTHING, SS:NOTHING
                ORG         100H
START:          JMP         BEGIN

; ДАННЫЕ
UNAVAILABLE_M   db          'Segment address of unavailable memory:     ',0dh,0ah,'$'
ENVIRONMENT_A   db          'Segment address of environment:    ',0dh,0ah,'$'
TAIL            db          'Tail:',0dh,0ah,'$'
SOD_SRED        db          'Content of the environment: ' , '$'
PATH            db          'Way to module: ' , '$'
ENDL            db          0dh,0ah,'$'

NEW_LINE        PROC  near
                lea         dx,ENDL
                call  Write_msg
                ret
NEW_LINE        ENDP

Write_msg       PROC  near
                mov         ah,09h
                int         21h
                ret
Write_msg       ENDP

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
; первод 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

; определяем сегментный адрес недоступной памяти
DEFINE_UN_MEMORY                PROC  near
                push    ax
                mov     ax,es:[2]
                lea             di,UNAVAILABLE_M
                add     di,42
                call    WRD_TO_HEX
```

```
                pop             ax
                ret
DEFINE_UN_MEMORY                ENDP

;определяем сегментый адрес среды, передаваемой программе
DEFINE_EN_A             PROC  near
                push    ax
                mov     ax,es:[2Ch]
                lea             di,ENVIRONMENT_A
                add     di,34
                call    WRD_TO_HEX
                pop             ax
                ret
DEFINE_EN_A             ENDP

;выводим хвост командной строки в символьном виде
DEFINE_TAIL             PROC  near
                push    ax
                push    cx
        xor     ax, ax
        mov     al, es:[80h]
        add     al, 81h
        mov     si, ax
        push    es:[si]
        mov     byte ptr es:[si+1], '$'
        push    ds
        mov     cx, es
        mov     ds, cx
        mov     dx, 81h
        call    Write_msg
                pop     ds
        pop     es:[si]
        pop             cx
        pop             ax
                ret
DEFINE_TAIL             ENDP

; Определяем содержимое области среды и путь к модулю
```

```
DEFINE_SODOS    PROC  near
                push    es
                push    ax
                push    bx
                push    cx
                mov             bx,1
                mov             es,es:[2ch]
                mov             si,0
        RE1:
                call    NEW_LINE
                mov             ax,si
        RE:
                cmp     byte ptr es:[si], 0
                je              NEXT2
                inc             si
                jmp     RE
        NEXT2:
                push    es:[si]
                mov             byte ptr es:[si], '$'
                push    ds
                mov             cx,es
                mov             ds,cx
                mov             dx,ax
                call    Write_msg
                pop             ds
                pop             es:[si]
                cmp             bx,0
                jz              LAST
                inc             si
                cmp     byte ptr es:[si], 01h
        jne     RE1
        call    NEW_LINE
        lea             dx,PATH
        call    Write_msg
        mov             bx,0
        add     si,2
        jmp     RE1
  LAST:
```

```
        call    NEW_LINE
                pop             cx
                pop             bx
                pop             ax
                pop             es
                ret
DEFINE_SODOS    ENDP

BEGIN:
                call    DEFINE_UN_MEMORY
                call    DEFINE_EN_A
                mov             dx, offset UNAVAILABLE_M
                call    Write_msg
                mov             dx, offset ENVIRONMENT_A
                call    Write_msg
                mov     dx, offset TAIL
                call    Write_msg
                call    DEFINE_TAIL
                call    NEW_LINE
                mov             dx, offset SOD_SRED
                call    Write_msg
                call    DEFINE_SODOS

                ; выход в DOS
                xor             al,al
                mov     ah, 01h
                int             21h
                mov     ah, 04Ch
                int     21h
                ret
TESTPC          ENDS
                END     START
```