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. 6383 _____ Михеева Е. Е.

Преподаватель _____ Губкин А. Ф.

Санкт-Петербург

2018

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системный 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.

Формат PSP:

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 (IP, CS).
0Eh(14)	4	Вектор прерывания 23h (IP, CS).
12h(18)	4	Вектор прерывания 24h (IP, CS).
2Ch(44)	2	Сегментный адрес среды, передаваемой программе.
5Ch		Область форматируется как стандартный неоткрытый блок управления файлом (FCB).
6Ch		Область форматируется как стандартный неоткрытый блок управления файлом (FCB). Перекрывается, если FCB с адреса 5Ch открыт.
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, 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.

Процедуры, используемые в программе

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа
BYTE_TO_HEX	Перевод байта в 16-ной с/с в символьный код
WRD_TO_HEX	Перевод слова в 16-ной с/с в символьный код
BYTE_TO_DEC	Перевод байта в 16-ной с/с в символьный код в 10-ной с/cprint
PRINT	Вывод строки на экран
PCTYPE	Определение кода типа PC
PRINT	Загружает в регистр ah код функции печати и вызывает прерывание int 21
INFO	Выводит требуемую в работе информацию

Структуры, используемые в программе

Структура	Тип	Описание
MEM	Строка, содержащая символы размером 1 байт	Содержит сегментный адрес недоступной памяти
ENV	Строка, содержащая символы размером 1 байт	Содержит сегментный адрес среды
TAIL	Строка, содержащая символы размером 1 байт	Содержит хвост командной строки в символьном виде
CONT	Строка, содержащая символы размером 1 байт	Содержит содержимое области среды в символьном виде

PATH	Строка, содержащая символы размером 1 байт	Содержит путь загружаемого модуля

Ход работы.

1. Результат работы программы:

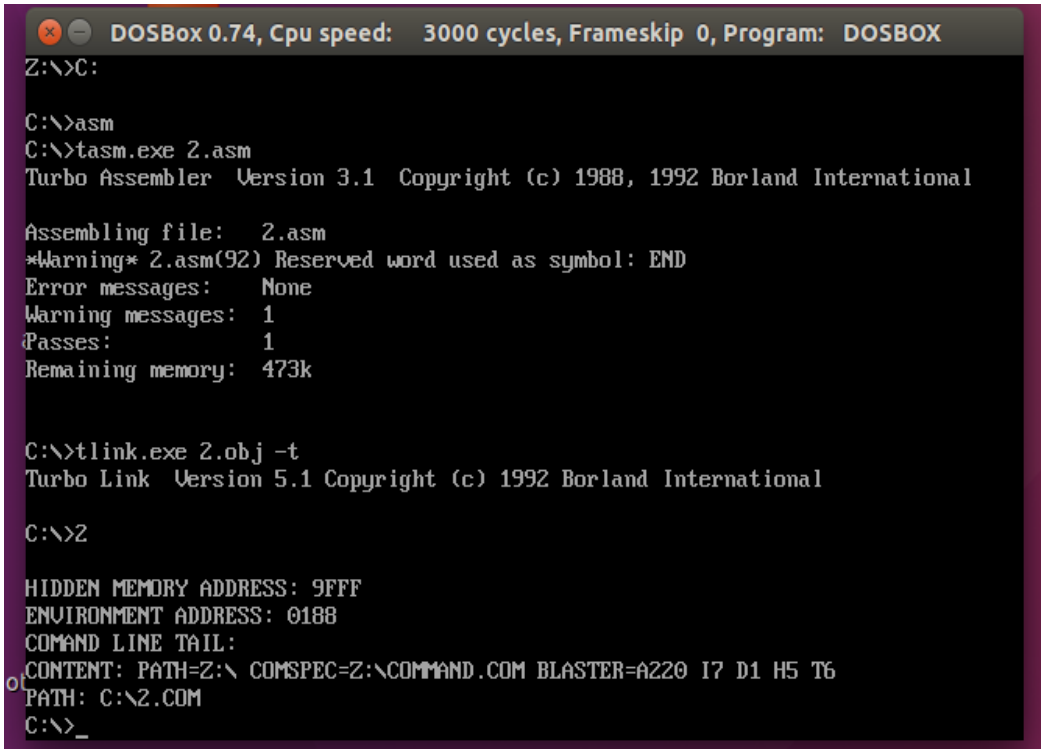

Рис. 1. Результат работы программы.

Ответы на вопросы.

Сегментный адрес недоступной памяти:

1. На какую область памяти указывает адрес недоступной памяти?
   На область памяти ROM BIOS.
2. Где расположен этот адрес по отношению области памяти, отведённой программе?

Этот адрес располагается с адреса 9FFF.

3. Можно ли в эту область памяти писать?

Нет, она доступна только на чтение

Среда передаваемая программе:

1. Что такое среда?

Среда – это область памяти, в которой в виде символьных строк записаны значения переменных (имя=параметр), называемых переменными среды. Они содержат данные о некоторых директориях операционной системы и конфигурации компьютера, которые передаются программе, когда она запускается.

2. Когда создается среда? Перед запуском приложения или в другое время?
Среда создаётся при загрузке DOS. При запуске программы эта среда только копируется в новую область памяти.

3. Откуда берется информация, записываемая в среду?
Информация для записи берётся из системного файла AUTOEXEC.BAT.

Вывод.

В ходе лабораторной работы были исследованы интерфейсы управляющей программы и загрузочных модулей, а также префикс сегмента программы (PSP) и среды, передаваемой программе. Была написана программа, которая выводит на экран сегментный адрес недоступной памяти, адрес среды, передаваемой программе, хвост командной строки и путь загружаемого модуля.

ПРИЛОЖЕНИЕ А. Код программы

```
TESTPC     SEGMENT
          ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
                org 100h
START:     JMP     BEGIN

MEM db 13, 10, "HIDDEN MEMORY ADDRESS:     $" ; 17 symbols
ENV db 13, 10, "ENVIRONMENT ADDRESS:     $" ; 23 symbols
TAIL db 13, 10, "COMAND LINE TAIL:        $" ; 21 symbols
CONT db 13, 10, "CONTENT: ", "$"
PATH db 13, 10, "PATH: ", "$" ; 8 symbols


;------------------------------
PRINT PROC near
        mov ah, 09h
        int 21h
        ret
PRINT ENDP
;------------------------------
INFO PROC near
        ; Hidden memory
        mov ax, ds:[02h]
        mov di, offset MEM
```

```
add di, 28
call WRD_TO_HEX
mov dx, offset MEM
call PRINT

; Environment
mov ax, ds:[2Ch]
mov di, offset ENV
add di, 26
call WRD_TO_HEX
mov dx, offset ENV
call PRINT

; Tail
xor cx, cx
mov cl, ds:[80h]
mov si, offset TAIL
add si, 20
cmp cl,0
jz empty
xor di, di
xor ax, ax
read_write_tail:
        mov al, ds:[81h+di]
        mov [si], al
        inc di
        inc si
        jmp read_write_tail
        mov dx, offset TAIL
```

```
        call PRINT
        jmp content
empty:
        mov dx, offset TAIL
        call PRINT
content:

; Content
mov dx, offset CONT
call PRINT
xor di, di
mov bx, 2Ch
mov ds, [bx]
readcontent:
        cmp byte ptr [di], 00h
        mov dl, [di]
        mov ah, 02h
        int 21h

        inc di
        cmp word ptr [di], 0001h
        jz readpath
        jmp readcontent
readpath:
        push ds
        mov ax, cs
        mov ds, ax
        mov dx, offset PATH
        call PRINT
```

```
            pop ds
            add di, 2
        pathloop:
            cmp byte ptr [di], 00h
            jz end
            mov dl, [di]
            mov ah, 02h
            int 21h
            inc di
            jmp pathloop
        end:
            ret
INFO 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
; áàéò â AL ïåðåâîäèòñÿ â ääà ñèìâîëà øåñòí. ÷èñëà â AX
          push     CX
          mov      AH,AL
```

```
        call    TETR_TO_HEX
        xchg    AL,AH
        mov     CL,4
        shr     AL,CL
        call    TETR_TO_HEX ;â AL ñòàðøàÿ öèôðà
        pop     CX          ;â AH ìëàäøàÿ
        ret
BYTE_TO_HEX  ENDP
;-------------------------------
WRD_TO_HEX   PROC  near
;ïåðåâîä â 16 ñ/ñ 16-òè ðàçðÿäíîãî ÷èñëà
; â AX - ÷èñëî, DI - àäðåñ ïîñëåäíåãî ñèìâîëà
        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
;--------------------------------------------
BYTE_TO_DEC   PROC  near
```

```
        push    CX
        push    DX
        xor     AH,AH
        xor     DX,DX
        mov     CX,10
loop_bd:  div     CX
        or      DL,30h
        mov     [SI],DL
              dec             si
        xor     DX,DX
        cmp     AX,10
        jae     loop_bd
        cmp     AL,00h
        je      end_l
        or      AL,30h
        mov     [SI],AL

end_l:    pop     DX
        pop     CX
        ret
BYTE_TO_DEC   ENDP
;-------------------------------
; ÊÎÄ
BEGIN:
              call INFO
              mov ah, 10h
              int 16h
; Âûõîä â DOS
```

```
        xor     AL,AL
        mov     AH,4Ch
        int     21H

TESTPC  ENDS
END     START   ;êîíåö ìîäóëÿ, START - òî÷êà âõîäà
```