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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студентка гр. 7381 _____ Процветкина А.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Исследование интерфейса управляющей программы, загрузочных модулей, 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. Необходимая в ходе работы информация о префиксе сегмента программы приведена в табл.1.

Таблица 1 – 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.
2Ch	2	Сегментный адрес среды, передаваемой программе.
80h	1	Число символов в хвосте командной строки
81h		Хвост командной строки

Область среды содержит последовательность символьных строк вида

имя = параметр

Каждая строка завершается байтом нулей, как и сама среда. За окончанием среды идут два байта 00h, 01h, после которых располагается маршрут загруженной программы. Маршрут так же заканчивается байтом 00h.

Постановка задачи.

Написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

1) Сегментный адрес недоступной памяти, взятый из PSP (HEX)

2) Сегментный адрес среды, передаваемой программе (HEX)

3) Хвост командной строки (ASCII)

4) Содержимое области среды (ASCII)

5) Путь загружаемого модуля (ASCII)

Выполнение работы.

По данным табл.1 сегментный адрес недоступной памяти хранится в PSP со смещением 2. Составленная программа извлекает эту информацию и преобразует ее в строковый вид функцией WRITE. Абсолютно аналогичные действия производятся с сегментным адресом среды, передаваемой программе. Информация о количестве символов в хвосте хранится в PSP со смещением 80h, а сам хвост следом – по смещению 81h. Очевидно, что нужно запустить цикл, считывающий по символу из адреса, смещающегося на каждом шаге на единицу вниз от 81h. Адрес начала среды, как указано в табл. 1, помещается в PSP со смещением 2Ch. Начиная с этого адреса, в памяти расположено содержимое области среды в виде ASCIIZ строк. Программа осуществляет их посимвольное считывание до момента, когда будут обнаружены два нулевых байта подряд. Известно, что следом за этими байтами хранятся последовательно байты 00h и 01h, а следом за ними путь загружаемого модуля. Этот путь считывается программой, пока не встретится одиночный нуль-байт.

Каждая сформированная ASCII-строка с требуемой информацией выводится непосредственно после считывания.

Результат работы программы представлен на рис. 1, а исходный код составленной программы представлен в приложении А.

Рисунок 1 – Результат работы программы

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

Ответ: Очевидно, что на ту, которую не следует модифицировать.

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

Ответ: Сегмент находится сразу после выделенной программе памяти.

3) Можно ли в эту область памяти писать?

Ответ: Нельзя.

Среда передаваемая программе

1) Что такое среда?

Ответ: Среда – совокупность переменных среды, которые могут использоваться приложениями для получения некоторой системной информации и для передачи данных между программами. Переменная среды – символьная строка в коде ASCII вида *имя* = *параметр*.

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

Ответ: При загрузке программы.

3) Откуда берется информация, записываемая в среду?

Ответ: Блок окружения наследуется от программы-родителя.

Выводы.

В ходе лабораторной работы был написан .COM модуль, извлекающий информацию о загрузочном модуле и среде из PSP и выводящий ее на экран. Исследован интерфейс управляющей программы, а так же изучены особенности такого понятия как среда.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ ТЕКСТ .COM МОДУЛЯ

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

str1 db 'Unaccessable memory starts from: ', '$'
mem_addr db 5 dup(?)
str2 db 'Segment adress provided to the program: ', '$'
seg_addr db 5 dup(?)
tail_msg db 'Tail of the command line: ', '$'
tail db 50 dup(?)
content_msg db 'Enviroment content: ', 0Dh, 0Ah, '$'
content db 256 dup(?)
path_msg db 'Path of the program: ', '$'
path db 50 dup(?)

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            ;num stored in AL into
   ASCII in AX in hex
        push CX
        mov     AH, AL
        call TETR_TO_HEX
        xchg AL,AH
        mov     CL, 4
        shr     AL, CL
        call TETR_TO_HEX
        pop     CX
        ret
BYTE_TO_HEX     ENDP

WRITE proc near
        mov cx, 2
cycle:
```

```
        xchg al, ah
        push ax
        call BYTE_TO_HEX
        mov [si], al
        inc si
        mov [si], ah
        inc si
        pop ax
        loop cycle

        mov byte ptr [si], '$'
        ret
WRITE endp

WRT_CONTENT proc near
        mov bx, 2Ch
        mov ax, es:bx
        mov es, ax
        xor si, si                              ;moving in
           enviroment
search:
        cmp byte ptr [es:si], 0                         ;EOL
        jne wrt
        mov byte ptr [di], 0Dh
        mov byte ptr [di+1], 0Ah
        add di, 2
        inc si
        cmp byte ptr [es:si], 0                         ;end of
           table
        je end_table
wrt:
        mov al, [es:si]
        mov [di], al
        inc di
        inc si
        jmp search

end_table:
        add si, 3
        mov byte ptr [di], '$'
        mov di, offset path
wrt_path:
        cmp byte ptr [es:si], 0
        je done
        mov bl, [es:si]
```

```
        mov [di], bl
        inc di
        inc si
        jmp wrt_path
done:
        mov byte ptr [di], 0Dh
        mov byte ptr [di+1], 0Ah
        mov byte ptr [di+2], '$'
        ret
WRT_CONTENT endp

BEGIN:
        mov bx, 2
        mov ax, es:bx
        mov si, offset mem_addr
        call WRITE

        mov ah, 09h
        mov dx, offset str1
        int 21h

        mov dx, offset mem_addr
        int 21h

        mov dl, 0Dh                             ;new line
        mov ah, 02h
        int 21h
        mov dl, 0Ah
        int 21h

        mov bx, 2Ch
        mov ax, es:bx
        mov si, offset seg_addr
        call WRITE

        mov ah, 09h
        mov dx, offset str2
        int 21h

        mov dx, offset seg_addr
        int 21h

        mov dl, 0Dh                             ;new line
        mov ah, 02h
        int 21h
```

```
        mov dl, 0Ah
        int 21h

        mov bx, 80h
        xor ch, ch
        mov cl, es:bx                   ;how long is the tail
        mov bx, 81h
        add bx, cx                              ;starting from the
           end
        dec bx
        mov si, offset tail
        add si, cx
        mov byte ptr [si], '$'
        dec si
wrt_tail:
        cmp cx, 0
        je skip
        mov ah, es:bx
        mov [si], ah
        dec si
        dec bx
        dec cx
        jmp wrt_tail

skip:
        mov dx, offset tail_msg
        mov ah, 09h
        int 21h

        mov dx, offset tail
        int 21h

        mov dl, 0Dh                             ;new line
        mov ah, 02h
        int 21h
        mov dl, 0Ah
        int 21h

        mov di, offset content
        call WRT_CONTENT
        mov dx, offset content_msg
        mov ah, 09h
        int 21h
        mov dx, offset content
        int 21h
```

```
        mov dx, offset path_msg
        int 21h
        mov dx, offset path
        int 21h

        xor     al, al
        mov     ah, 4Ch
        int     21h
CSEG    ENDS

END     START
```