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

Студент гр. 9381 _____ Судаков Е.В.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург

2021

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.

Ход работы.

1. Был написан .com модуль, распечатывающий 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.

2. Был оформлен отчет в соответствии с требованиями. Результат выполнения программы приведен на рисунке ниже.

Рис.1. Результат запуска программы

Процедуры.

Название процедуры	Назначение
GET_SEG_UNAVAIL	Выводит на экран сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде.
GET_SEG_ENV_INFO	Выводит на экран сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.
GET_TAIL_CMD_INFO	Выводит на экран хвост командной строки в символьном виде.
GET_ENV_INTRINSIC_INFO	Выводит на экран содержимое области среды в символьном виде.
GET_COM_PATH	Выводит на экран путь загружаемого модуля.

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

Указывает на сегментный адрес первого байта, находящегося после памяти, выделенной программе.

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

После памяти, выделенной программе.

3) Можно ли в эту область памяти писать?

Можно, т.к. в MSDOS нет механизма защиты памяти.

Среда передаваемая программе

1) Что такое среда?

Среда(окружение) - область памяти, в которой в виде символьных строк записаны значения переменных, называемых переменными окружения. Каждая строка имеет формат

переменная = значение

и заканчивается нулевым байтом. Имеется ряд переменных окружения, имена которых зарезервированы и известны системе, однако пользователь может включать в окружение и свои переменные для использования их прикладными программами.

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

Окружение для командного процессора, создается в процессе начальной загрузки. Дочерние процессы(программы), пользуются этим окружением, могут включать и в него “персональные” переменные, адресованные конкретной программе.

3) Откуда берется информация, записываемая в среду?

Из файла AUTOEXEC.BAT

Заключение.

Приложение А. Исходный код.

Файл lab2_com.asm.

```
LAB2 SEGMENT
          ASSUME CS:LAB2, DS:LAB2, ES:NOTHING, SS:NOTHING
          ORG 100H
START:    JMP  BEGIN

SEG_UNAVAIL_INFO db 'Segment address of unavailable memory:     ',
13, 10, '$';
```

```
SEG_ENV_INFO db 'Segment address of environment :     ', 13, 10,
'$';
TAIL_CMD_INFO db 'Tail of command : ',  '$';
ENV_INTRINSIC_INFO db 'Intrinsics of environment area : ', 13, 10,
'$';
COM_PATH db 'Path of module : ', '$';
NEWLINE db 0dh,0ah, '$'

;ПРОЦЕДУРЫ
;-----------------------------------------------------------------
---------------
; Перевод тетрады (4-ех младших байтов AL) в 16-ичную СС и ее
представление в виде символа
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

; Перевод байта AL в 16-ичную СС и его представление в виде
символов
BYTE_TO_HEX PROC NEAR
     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

; Перевод слова AX в 16-ичную СС и его представление в виде
символов
WORD_TO_HEX PROC NEAR
     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
WORD_TO_HEX ENDP

; Перевод байта AL в 10-ичную СС и его представление в виде
символов
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

; Вызывает функцию вывода строки на экран
PRINT PROC NEAR
     push ax
     mov ah, 09h
     int 21h
     pop ax
     ret
PRINT ENDP

;-------------------------------------------------------------------
---------------

GET_SEG_UNAVAIL PROC NEAR
    mov ax, ds:[02h]
    mov di, offset SEG_UNAVAIL_INFO + 42
    call WORD_TO_HEX
    mov dx, offset SEG_UNAVAIL_INFO
    call PRINT
    ret
GET_SEG_UNAVAIL ENDP

GET_SEG_ENV_INFO PROC NEAR
    mov ax, ds:[2ch]
```

```
    mov di, offset SEG_ENV_INFO + 36
    call WORD_TO_HEX
    mov dx, offset SEG_ENV_INFO
    call PRINT
    ret
GET_SEG_ENV_INFO ENDP

GET_TAIL_CMD_INFO PROC NEAR
    mov dx, offset TAIL_CMD_INFO
    call PRINT
    mov cl, ds:[080h]
    cmp cl, 0
    je return_tail_cmd_info

    xor di, di
    xor ch, ch
    mov ah, 02h
print_loop:
    mov dl, ds:[081h + di]
    int 21h
    inc di
    loop print_loop;

return_tail_cmd_info:
    mov dx, offset NEWLINE
    call PRINT
    xor ah, ah
    ret
GET_TAIL_CMD_INFO ENDP

GET_ENV_INTRINSIC_INFO PROC NEAR

    mov dx, offset ENV_INTRINSIC_INFO
    call PRINT
```

```
    mov es, ds:[2ch]
    xor di, di
should_print_more:
    mov dl, es:[di]
    cmp dl, 0
    je print_newline

print_line:
    mov dl, es:[di]
    mov ah, 02h
    int 21h
    inc di
    jmp should_print_more

print_newline:
    mov dx, offset NEWLINE
    call PRINT
    inc di
    mov dl, es:[di]
    cmp dl, 0
    jne should_print_more

return_env_intr_info:
    mov dx, offset NEWLINE
    call PRINT
    ret
GET_ENV_INTRINSIC_INFO ENDP

GET_COM_PATH PROC NEAR
    mov dx, offset COM_PATH
    call PRINT

    add di, 3

print_path:
```

```
    mov dl, es:[di]
    cmp dl, 0
    je return_com_path

    mov ah, 02h
    int 21h
    inc di
    jmp print_path


return_com_path:
    ret
GET_COM_PATH ENDP

BEGIN:

        call GET_SEG_UNAVAIL
        call GET_SEG_ENV_INFO
        call GET_TAIL_CMD_INFO
        call GET_ENV_INTRINSIC_INFO
        call GET_COM_PATH

     xor al, al
     mov ah, 4ch
     int 21h
     ret

LAB2      ENDS
     END  START
```