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. 9383 _____ Арутюнян С.Н.

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.

Порядок выполнения работы

Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает 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.

Выполнение работы

В процессе выполенения лабораторной работы была разработана .COM-программа, исходный текст которой приведен в приложении А.

Рис. 1. Пример работы программы без аргументов

Рис. 2. Пример работы программы с аргументами

Ответы на вопросы

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

На первый байт после куска памяти, отведенной под программу.

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

В PSP по смещению 2Ch.

3. Можно ли в эту область памяти писать?

Можно, т. к. DOS не имеет механизмов защиты от перезаписи памяти различными программами, которая им не отведена.

Среда, передаваемая программе

1. Что такое среда?

Область памяти, хранящая переменные среды. Переменные среды хранят некоторую информацию о состоянии системы, например, путь к домашней директории компьютера.

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

Изначальная среда создается при запуске ОС. Эта среда копируется (и после чего может измениться в соответствии с требованиями конкретной программы, если ей это необходимо) в адресное пространство запущенной программы. Также, в программу, которую запустила другая программа, копируется среда родительской программы.

3. Откуда берется информация, записываемая в среду?

Информация берется из файла AUTOEXEC.BAT, который расположен в корневом каталоге загрузочного устройства (диск, дискета, флешка, с которой была запущена ОС).

Заключение

В процессе выполнения лабораторной работы были получены навыки работы с PSP, а также изучены интерфейсы программных модулей DOS.

Приложение А

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

start: jmp begin

; DATA

UNAVAILABLE_MEM_GREET db 'Unavailable memory: $'
ENV_ADDRESS_GREET db 'Environment address: $'
EMPTY_COMMAND_TAIL db 'The command tail is empty$'
COMMAND_TAIL_STR db 'Command tail:$'
CONTENT_GREET db 'Content of the environment:', 0dh, 0ah, '$'
PATH_GREET db 'Path: $'
HEXED_AX db '    ', '$' ; ax - 2 байта, 2**16 содержит 5 символов => резервируем 5 байт + символ конца строки
NEWLINE db 0dh, 0ah, '$'


; Перевод ax в строку HEXED_AX
AX_TO_STRING proc near

   push bx
   push di

   xor bx, bx

   mov di, offset HEXED_AX

   ; high byte
   mov bl, ah
   and bl, 0f0h
   call BL_TO_ASCII_HEX
   mov [di], bl
```

```
    inc di

    mov bl, ah
    and bl, 00fh
    call BL_TO_ASCII_HEX
    mov [di], bl
    inc di

    ; low byte
    mov bl, al
    and bl, 0f0h
    call BL_TO_ASCII_HEX
    mov [di], bl
    inc di

    mov bl, al
    and bl, 00fh
    call BL_TO_ASCII_HEX
    mov [di], bl
    inc di

    mov byte ptr [di], '$'

    pop di
    pop bx
    ret

AX_TO_STRING endp

BL_TO_ASCII_HEX proc near

    push cx

    cmp bx, 16  ; если bl >= 16, то нужно сдвинуть все вправо на 4 бита
    jl start_converting
```

```
    mov cl, 4
    shr bx, cl

start_converting:
    cmp bl, 0ah
    jge bl_is_letter

    ; bl is digit
    add bl, 48
    jmp exit

bl_is_letter:
    add bl, 55

exit:
    pop cx
    ret

BL_TO_ASCII_HEX endp

PRINT_NEWLINE proc near

    push ax
    push dx

    mov dx, offset NEWLINE
    mov ah, 9h
    int 21h

    pop dx
    pop ax

    ret

PRINT_NEWLINE endp
```

```
; Печатает строку в dx
WRITE_STRING proc near

    push ax
    mov ah, 9h
    int 21h
    pop ax

    ret

WRITE_STRING endp


PSP_UNAVAILABLE proc near

    push ax
    push dx

    mov dx, offset UNAVAILABLE_MEM_GREET
    call WRITE_STRING

    mov ax, ds:[02h]
    call AX_TO_STRING
    mov dx, offset HEXED_AX
    call WRITE_STRING

    pop dx
    pop ax
    ret

PSP_UNAVAILABLE endp


SEGMENT_ENV_ADDRESS proc near
```

```
    push ax
    push dx

    mov dx, offset ENV_ADDRESS_GREET
    call WRITE_STRING

    mov ax, ds:[2ch]
    call AX_TO_STRING
    mov dx, offset HEXED_AX
    call WRITE_STRING

    pop dx
    pop ax
    ret

SEGMENT_ENV_ADDRESS endp


COMMAND_TAIL proc near

    push cx
    push dx
    push si

    xor cx, cx

    ; Сначала вытаскиваем количество параметров
    mov cl, ds:[80h]
    cmp cl, 0
    je empty_tail

    mov dx, offset COMMAND_TAIL_STR
    call WRITE_STRING
```

```
    ; Для чтения
    mov si, 0
    ; Для печати
    read_tail:
        mov dl, ds:[81h+si]
        mov ah, 02h
        int 21h
        inc si
        loop read_tail

    jmp command_tail_exit

empty_tail:
    mov dx, offset EMPTY_COMMAND_TAIL
    call WRITE_STRING

command_tail_exit:
    pop si
    pop dx
    pop cx
    ret

COMMAND_TAIL endp

ENV_CONTENT proc near

    push ax
    push dx
    push si
    push ds

    mov dx, offset CONTENT_GREET
    call WRITE_STRING

    xor si, si
```

```
    mov ds, ds:[2ch]

    read_line_loop:
        mov dl, [si]
        cmp dl, 0
        je end_of_line

        mov ah, 02h
        int 21h

        inc si
        jmp read_line_loop

    end_of_line:
        inc si
        mov dl, [si]
        cmp dl, 0               ; если конец всей области среды
        je content_end

        pop ds
        call PRINT_NEWLINE
        push ds
        mov ds, ds:[2ch]

        jmp read_line_loop      ; иначе возвращаемся и читаем заново

content_end:
    pop ds
    call PRINT_NEWLINE
    mov dx, offset PATH_GREET
    call WRITE_STRING
    push ds
    mov ds, ds:[2ch]

    ; Теперь нужно вывести путь программы, si указывает на последний ноль среды
```

```
    ; после него идут 00h и 01h, а потом уже путь.
    add si, 3
    read_path_loop:
        mov dl, [si]
        cmp dl, 0
        je env_printing_exit

        mov ah, 02h
        int 21h
        inc si
        jmp read_path_loop

env_printing_exit:
    pop ds
    pop si
    pop dx
    pop ax
    ret

ENV_CONTENT endp



begin:
    ; 1. Вывести адрес, содержащийся в байтах 2 и 3 от начала PSP
    call PSP_UNAVAILABLE
    call PRINT_NEWLINE

    ; 2. Вывести сегментный адрес среды
    call SEGMENT_ENV_ADDRESS
    call PRINT_NEWLINE

    ; 3. Вывести хвост командной строки
    call COMMAND_TAIL
    call PRINT_NEWLINE
```

```
    ; 4, 5. Вывести содержимое области среды в символьном виде
    call ENV_CONTENT

    ; выход в DOS
    xor al, al
    mov ah, 4ch
    int 21h

TESTPC  ENDS
        END start
```