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. 8382 Мирончик П.Д.

Преподаватель Ефремов М.А.

Санкт-Петербург

2019

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей, префикса сегмента программы и среды, передаваемой программе.

Ход работы.

Был написан COM файл, который выводит следующие данные:

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.

После выполнения программы в терминал выводятся следующие данные:

Рисунок 1 – результат исполнения COM модуля

Сегментный адрес недоступной памяти.

1. На какую область указывает адрес недоступной памяти?

На первый байт, который находится за доступной (выделенной) программе памятью.

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

Сразу после выделенной программе памяти.

3. Можно ли в эту область писать?

Можно.

Среда, передаваемая программе.

1. Что такое среда?

Среда – набор системных переменных, которые передаются программе при запуске.

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

Среда создается при загрузке ОС.

3. Откуда берется информация, записываемая в среду?

Из родительской среды – в программе выделяется область памяти, в которую копируется родительская среда.

Дополнительные вопросы:

* Откуда "родительская среда" берет информацию? И что этой родительской программой является?

Изначально родительская среда инициализируется из файла autoexec.bat. Затем, при запуске одной программы другой, эта среда по умолчанию просто копируется.

Для программ, запускаемых из консоли, родительской программой будет command.com.

Вывод.

В ходе лабораторной работы были исследованы интерфейсы управляющей программы и загрузочных модулей, особенности передачи

программе управляющего блока, содержащего адреса и системные данные, префикса сегмента программы и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А.

КОД ИСХОДНОГО COM МОДУЛЯ

```
MAIN    SEGMENT
        ASSUME CS:MAIN, DS:MAIN, SS:NOTHING
        ORG 100H

START:  JMP BEGIN
; DATA
UNAVAILABLE_ADDRESS db 'Unavailable memory address: $'
ENVIROMENT_ADDRESS db 'Enviroment address: $'
COMMAND_LINE_EDGE db 'End of command line: $'
NO_COMMAND_LINE_EGDE db 'End of command line is empty',10,13,'$'
ENVIDOMENT_DATA db 'Enviroment data: $'
MODULE_PATH db 'Loaded modlue path: $'
ENDL db 13,10,'$'

; PROCEDURES
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
; input:        AL=F8h (число)
; output:       AL={f}, AH={8} (в фигурных скобках символы)
;
; переводит AL в два символа в 16-й сс в AX
; в AL находится старшая, в AH младшая цифры
        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
;--------------------------
WRD_TO_HEX      PROC    near
; input:        AX=FH7Ah (число)
;               DI={адрес} (указатель на последний символ в памяти, куда
будет записан результат)
; output:       начиная с [DI-3] лежат символы числа в 16-й сс
;               AX не сохраняет начальное значение
;
; перевод AX в 16-ю сс
        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
; input:        AL=0Fh (число)
;               SI={адрес} (адрес поля младшей цифры)
;
; перевод AL в 10-ю сс
        push    cx
        push    dx
        push    ax
        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     ax,10
        je      end_l
        or      al,30h
        mov     [si],al
end_l:
        pop     ax
        pop     dx
        pop     cx
        ret
BYTE_TO_DEC     ENDP

WRITE_AL_HEX PROC NEAR
        push ax
        push dx
        call BYTE_TO_HEX

        mov dl, al
        mov al, ah
        mov ah, 02h
        int 21h

        mov dl, al
        int 21h
        pop dx
        pop ax
        ret
```

```
WRITE_AL_HEX ENDP

WRITE_ENDL PROC NEAR
        push ax
        push dx

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

        pop dx
        pop ax
        ret
WRITE_ENDL ENDP

;--------------------------
; CODE
BEGIN:

    ; Сегментный адрес недоступной памяти
    mov dx, offset UNAVAILABLE_ADDRESS
    mov ah, 09h
    int 21h

    mov bx, ds:[02h]
    mov al, bh
    call WRITE_AL_HEX
    mov al, bl
    call WRITE_AL_HEX
    call WRITE_ENDL

    ; Сегментный адрес среды
    mov dx, offset ENVIROMENT_ADDRESS
    mov ah, 09h
    int 21h

    mov bx, ds:[2Ch]
    mov al, bh
    call WRITE_AL_HEX
    mov al, bl
    call WRITE_AL_HEX
    call WRITE_ENDL

    ; Хвост командной строки в символьном виде
    mov ch, 0h
    mov cl, ds:[80h]

    cmp cl, 0
    je no_edge

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

    mov bx, 0
edge_loop:
    mov dl, ds:[81h+bx]
    mov ah, 02h
```

```
    int 21h

    inc bx
    loop edge_loop

    call WRITE_ENDL

no_edge:
    mov dx, offset NO_COMMAND_LINE_EGDE
    mov ah, 09h
    int 21h

    ; Вывод данных области среды
    mov dx, offset ENVIDOMENT_DATA
    mov ah, 09h
    int 21h

    mov es, ds:[2Ch]
    mov bx, 0
    print_env_variable:
        mov dl, es:[bx]

        cmp dl, 0
        je variable_end

        mov ah, 02h
        int 21h
        inc bx
        jmp print_env_variable
    variable_end:
        mov dl, es:[bx+1]
        call WRITE_ENDL
        cmp dl, 0
        je enviroment_end

        inc bx
        jmp print_env_variable
enviroment_end:
    ; Вывод пути загружаемого модуля
    mov dx, offset MODULE_PATH
    mov ah, 09h
    int 21h

    add bx, 2
    path_loop:
        mov dl, es:[bx]
        cmp dl, 0
        jne print_path_byte
        cmp byte ptr es:[bx+1], 0
        je path_end
    print_path_byte:
        mov ah, 02h
        int 21h
        inc bx
        jmp path_loop
path_end:

; Выход в DOS
```

```
dos_exit:
        xor     al,al
        mov     ah,4Ch
        int     21h
MAIN    ENDS
        END START
```