

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.

Студент гр. 8382

Терехов А.Е.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

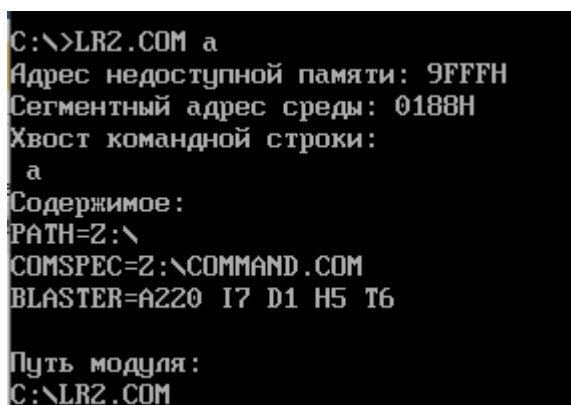
Цель работы.

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. С помощью данных в методических указаниях функций был написан исходник для загрузочного модуля с расширением .COM. Программа должна находить и выводить в консоль адрес недоступной памяти, адрес среды, хвост командной строки, содержимое области среды и путь загружаемого модуля.

2. После трансляции с использованием MASM, линковки и конвертации в COM-программу была получена рабочая программа. Программа работает корректно, как это можно заметить на рис.1.



```
C:\>LR2.COM а
Адрес недоступной памяти: 9FFFH
Сегментный адрес среды: 0188H
Хвост командной строки:
а
Содержимое:
PATH=Z:\
COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Путь модуля:
C:\LR2.COM
```

Рис.1. Запуск программы.

Ответы на вопросы.

Сегментный адрес недоступной памяти.

1. На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на адрес, с которого начинается блок недоступной памяти для загрузки пользовательских программ.

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

программе?

Адрес указывает на память, следующую следом за выделенной для программы памятью.

3. Можно ли в эту область памяти писать?

DOS никак не контролирует обращение к памяти, поэтому можно.

Среда, передаваемая программе.

1. Что такое среда?

Среда окружения — совокупность значений системных переменных, путей, открытых файловых дескрипторов и других ресурсов операционной системы, передаваемые процессу (программе) при его запуске.

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

Среда, передаваемая программе, является копией среды родительского процесса и создается во время загрузки программы в оперативную память. Корневая среда создается при запуске ОС.

3. Откуда берется информация, записываемая в среду?

Информация берется из процесса порождающего, вызываемую программу, будь то DOS или другая ОС, или другая программа. Корневая среда создается по информации из файла AUTOEXEC.BAT, создающегося при запуске ОС.

Вывод.

В процессе выполнения лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей, структура и содержание PSP, а также содержание среды.

ПРИЛОЖЕНИЕ А

Исходный код программы.

```
CODESEG SEGMENT
    ASSUME cs:CODESEG, ds:CODESEG, es:NOTHING, ss:NOTHING
    ORG 100H
START: jmp BEGIN
MEMORY      db'Адрес недоступной памяти:      H',0DH,0AH,'$'
ENV         db'Сегментный адрес среды:      H',0DH,0AH,'$'
NO_SYMBOL   db'Нет символов командной строки. ',0DH,0AH,'$'
TAIL        db'Хвост командной строки:',0DH,0AH,'$'
TAIL_CONT   db'                                ',0DH,0AH,'$'
ENV_CONT     db'Содержимое области среды:      ',0DH,0AH,'$'
PATH        db'Путь модуля:                    ',0DH,0AH,'$'
CONT        db'Содержимое:',0DH,0AH,'$'
N           db 0DH,0AH,'$'
; Процедура печати строки
WriteMsg PROC near
    mov ah,09h
    int 21h
    ret
WriteMsg ENDP
;-----
TETR_TO_HEX PROC near
    and al,0Fh
    cmp al,09
    jbe NEXT
    add al,07
NEXT:    add al,30h; код нуля
    ret
TETR_TO_HEX ENDP
;-----
BYTE_TO_HEX PROC near
; байт в al переводится в два символа шестн. числа в ah
    push cx
    mov ah,al
    call TETR_TO_HEX
    xchg al,ah
    mov cl,4
    shr al,cl
    call TETR_TO_HEX ;в al старшая цифра
    pop cx ;в ah младшая
    ret
BYTE_TO_HEX ENDP
;-----
WRD_TO_HEX PROC near
;перевод в 16 с/с 16-ти разрядного числа
; в ax - число, di - адрес последнего символа
    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
; перевод байта в 10с/с, si - адрес поля младшей цифры
; al содержит исходный байт
    push ax
    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
    pop ax
    ret
BYTE_TO_DEC ENDP
;-----

BEGIN:
    mov ax, ds:[02h]
    lea di, MEMORY
    add di, 29
    call WRD_TO_HEX
    lea dx, MEMORY
    call WriteMsg
    mov ax, ds:[2Ch]
    lea di, ENV
    add di, 27
    call WRD_TO_HEX
    lea dx, ENV
    call WriteMsg
    xor cx, cx
    mov cl, ds:[80h]
    test cl, cl
    jz NO_SMB
    xor di, di
    xor ax, ax
    mov si, 81h
    lea di, TAIL_CONT
TAIL_READ:
    lodsb
    stosb
    loop TAIL_READ
    lea dx, TAIL
    call WriteMsg
    lea dx, TAIL_CONT
    call WriteMsg
    jmp END_LOOP
NO_SMB: lea dx, NO_SYMBOL
    call WriteMsg
END_LOOP:
    mov dx, offset CONT

```

```

        call WriteMsg
        xor di, di
        mov bx, 2Ch
        mov ds, [bx]
READ:
        cmp byte ptr [di], 00h
        jz BSLASHN
        mov dl, [di]
        mov ah, 02h
        int 21h
        jmp PARAM
BSLASHN:
        push ds
        mov cx, cs
        mov ds, cx
        lea dx, N
        call WriteMsg
        pop ds
PARAM:
        inc di
        cmp word ptr [di], 0001h
        jz READPATH
        jmp READ
READPATH:
        push ds
        mov ax, cs
        mov ds, ax
        lea dx, PATH
        call WriteMsg
        pop ds
        add di, 2
LOOPPATH:
        cmp byte ptr [di], 00h
        jz ENDING
        mov dl, [di]
        mov ah, 02h
        int 21h
        inc di
        jmp LOOPPATH
ENDING:
        xor al, al
        mov ah, 4Ch
        int 21h
CODESEG ENDS
END START ;конец модуля, START - точка входа

```