

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. 7381

Дорох С.В.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2019

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.

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. При загрузке модулей типа .COM все сегментные регистры указывают на адрес PSP. При загрузке модуля типа .EXE сегментные регистры DS и ES указывают на PSP. Именно по этой причине значения этих регистров в модуле .EXE следует переопределять.

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.
4	6	Зарезервировано
0Ah (10)	4	Вектор прерывания 22h (IP,CS)
0Eh (14)	4	Вектор прерывания 23h (IP,CS)
12h (18)	4	Вектор прерывания 24h (IP,CS)
2Ch (44)	2	Сегментный адрес среды, передаваемой программе.
5Ch		Область форматируется как стандартный неоткрытый блок управления файлом (FCB)
6Ch		Область форматируется как стандартный неоткрытый блок управления файлом (FCB). Перекрывается, если FCB с адреса 5Ch открыт.
80h	1	Число символов в хвосте командной строки.
81h		Хвост командной строки - последовательность символов после имени вызываемого модуля.

Область среды содержит последовательность символьных строк вида: имя = параметр. Каждая строка завершается байтом нулей. В первой строке

указывается имя COMSPEC, которая определяет используемый командный процессор и путь к COMMAND.COM. Следующие строки содержат информацию, задаваемую командами PATH, PROMPT, SET. Среда заканчивается также байтом нулей. Таким образом, два нулевых байта являются признаком конца переменных среды. Затем идут два байта, содержащих 00h, 01h, после которых располагается маршрут загруженной программы. Маршрут также заканчивается байтом 00h.

Сведения о функциях управляющей программы:


Функция `tetr_to_hex` осуществляет перевод половины байта в символ.

Функция `byte_to_hex` осуществляет перевод байта, помещенного в `al`, в два символа в шестнадцатиричной системе счисления, помещая результат в `ax`.

Функция `wrd_to_hex` осуществляет перевод числового значения, помещенного в регистр `AX`, в символьную строку в шестнадцатиричной системе счисления, помещая результат в регистр `di`.

Функция `print` осуществляет вывод на экран.

Результат выполнение работы.



```
Address of inaccessible memory:9FFF
Segment address:0188
Tail of command line:
Enviroment contents:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Path of program:
C:\LAB2.COM
```

Рисунок 1. Результат запуска файла lab2.com

Выводы.

В ходе работы было проведено исследование интерфейса управляющей программы и загрузочных модулей, а также исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти.

1. На какую область памяти указывает адрес недоступной памяти?

Ответ: адрес недоступной памяти указывает на границу области, доступной для загрузки программ, и границу основной оперативной памяти.

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

Ответ: адрес располагается сразу за памятью, отведённой программе.

3. Можно ли в эту область памяти писать?

Ответ: можно.

Среда передаваемая программе.

1. Что такое среда?

Ответ: область памяти, содержащая переменные среды, которые могут использоваться приложениями для получения некоторой системной информации и для передачи данных между программами.

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

Ответ: при загрузке DOS; при запуске программы происходит лишь копирование среды в новую область памяти.

3. Откуда берется информация, записываемая в среду?

Ответ: информация записывается в среду из системного файла autoexec.bat.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ ТЕКСТ .COM МОДУЛЯ

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

```
START:  JMP  BEGIN
```

```
NO_ACCESS_M      db  'Address of inaccessible memory:
',0dh,0ah,'$'
SEG_ADDRESS_LINE      db  'Segment address:  ',0dh,0ah,'$'
COM_TEXT      db  'Tail of command line:  ',0dh,0ah,'$'
SREDA_CONTENTS_LINE  db  'Enviroment contents:',0dh,0ah,'$'
DIRECT_LINE      db  'Path of program:',0dh,0ah,'$'
NEW_LINE      db  ' ',0dh,0ah,'$'
```

```
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
```

```
        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
```

```

    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
    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_1
    or AL,30h
    mov [SI],AL
end_1:    pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP

```

```

NO_ACCESS_MEMORY    PROC NEAR

```

```

        push ax
        push di
        mov ax, ds:[2h]
        mov di, offset NO_ACCESS_M
        add di, 34
        call WRD_TO_HEX
        pop di
        pop ax
        ret
NO_ACCESS_MEMORY      ENDP

SEG_ADDRESS           PROC NEAR
        push ax
        push di

        mov ax, ds:[2Ch]
        mov di, offset SEG_ADDRESS_LINE
        add di, 19
        call WRD_TO_HEX

        pop di
        pop ax
        ret
SEG_ADDRESS           ENDP

TAIL_COM_LINE         PROC NEAR

        push ax
        push cx
        push dx
        push si
        push di

        xor cx, cx
        mov ch, ds:[80h]
        mov si, 81h
        mov di, offset COM_TEXT
        add di, 21
copy_Line:

```



```

        cmp ch, 0h
        je fin

        mov al, ds:[si]
        mov [di], al
        inc di
        inc si
        dec ch
        jmp copy_line
fin:
        mov al, 0h
        mov [di], al

        pop di
        pop si
        pop dx
        pop cx
        pop ax
        ret
TAIL_COM_LINE      ENDP

SREDA_CONTENTS     PROC NEAR
        push ax
        push dx
        push ds
        push es

        mov ah, 2h
        mov es, ds:[2Ch]
        xor si, si
copy:
        mov dl, es:[si]
        int 21h
        cmp dl, 0h
        je final
        inc si
        jmp copy

```

```

final:
    mov dx, offset NEW_LINE
    call PRINT
    inc si
    mov dl, es:[si]
    cmp dl, 0h
    jne copy

    mov dx, offset NEW_LINE
    call PRINT

    mov dx, offset DIRECT_LINE
    call PRINT

    add si, 3h
    mov ah, 02h
    mov es, ds:[2Ch]
write_dir:
    mov dl, es:[si]
    cmp dl, 0h
    je end_dir
    int 21h
    inc si
    jmp write_dir
end_dir:
    nop

    pop es
    pop ds
    pop dx
    pop ax
    ret
SREDA_CONTENTS      ENDP

PRINT      PROC near
    push ax
    mov    ah,09h

```

```

                                int          21h
                                pop ax
                                ret
PRINT      ENDP

; КОД
BEGIN:

                                call NO_ACCESS_MEMORY
                                mov dx, offset NO_ACCESS_M
                                call PRINT
                                mov dx, offset NEW_LINE
                                call PRINT

                                call SEG_ADDRESS
                                mov dx, offset SEG_ADDRESS_LINE
                                call PRINT
                                mov dx, offset NEW_LINE
                                call PRINT

                                call TAIL_COM_LINE
                                mov dx, offset COM_TEXT
                                call PRINT
                                mov dx, offset NEW_LINE
                                call PRINT
                                mov dx, offset SREDA_CONTENTS_LINE
                                call PRINT

                                call SREDA_CONTENTS

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

TESTPC      ENDS
                                END START

```