

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

Студентка гр. 9383

Лихашва А.Д.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Постановка задачи

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 в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Сведения о функциях и структурах данных.

В данной программе используются следующие функции и структуры данных:

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа, который записывается в AL
BYTE_TO_HEX	Перевод значений байта в число 16-ой СС и его представление в виде двух символов
WRD_TO_HEX	Перевод слова в число 16-ой СС и представление его в виде четырех символов
BYTE_TO_DEC	Перевод значения байта в число 10-ой СС и представляет его в виду символов
PRINT_STRING	Вывод строки на экран
MEMORY_ADDRESS	Печать на экран сегментный адрес недоступной памяти из PSP
ENV_ADDRESS	Печать на экран сегментный адрес среды, передаваемой программе
TAIL_COMMAND	Печать на экран хвоста командной строки
AREA_ENVIROMENT	Печать на экран содержимого области среды
PATH	Печать на экран путь загружаемого

	модуля
--	--------

Выполнение шагов лабораторной работы:

1 шаг:

- 1) На экран печатается сегментный адрес недоступной памяти из PSP в шестнадцатеричном виде
- 2) На экран печатается сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде
- 3) На экран печатается хвост командной строки в символьном виде
- 4) На экран печатается содержимое области среды в символьном виде
- 5) На экран печатается путь загружаемого модуля

Результаты, полученные программой:

```
C:\>lab2.com
Segment address of the unavailable memory: 9FFF
Segment address of the environment: 0188
Command tail is empty
Content of the environment area:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Path of the loaded module:
C:\LAB2.COM
C:\>
```

Рис. 1. - Пример работы программы

```
C:\>lab2.com hello
Segment address of the unavailable memory: 9FFF
Segment address of the environment: 0188
Tail command of the string:
hello
Content of the environment area:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Path of the loaded module:
C:\LAB2.COM
C:\>
```

Рис. 2. - Пример работы программы

2 шаг:

Был оформлен отчет в соответствии с требованиями. В отчете включены скриншоты с запуском программы и результатами.

Ответы на контрольные вопросы

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на адрес следующего сегмента памяти после участка памяти, отведенного под программу.

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

В PSP по смещению 02h.

3) Можно ли в эту область памяти писать?

Можно, потому что DOS не имеет механизмов защиты перезаписи памяти программ, для которых эта память не выделялась.

Среда передаваемая программе

1) Что такое среда?

Среда — это участок памяти, который содержит в себе значения переменных среды, путей и других данных операционной системы. Переменные среды хранят информацию о состоянии системы.

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

Среда создается при загрузке модуля в оперативную систему.

3) Откуда берется информация, записываемая в среду?

Данная информация берется из файла AUTOEXEC.BAT, который расположен в корневом каталоге загрузочного устройства.

Заключение.

Был исследован интерфейс управляющей программы и загрузочных модулей. Был исследован префикс сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: lab2.asm

```
TESTPC  SEGMENT

    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

    ORG 100H

START:  JMP BEGIN


; Данные

SEGMENT_MEMORY_ADDRESS DB 'Segment address of the
unavailable memory:      ', 0DH, 0AH, '$'

SEGMENT_ADDRESS_ENV DB 'Segment address of the environment:
', 0DH, 0AH, '$'

TAIL_COMMAND_STRING DB 'Tail command of the string: ',
0DH, 0AH, '$'

EMPTY_STRING DB 'Command tail is empty', '$'

NEW_STRING DB 0DH, 0AH, '$'

AREA_ENV DB 'Content of the environment area: ', 0DH, 0AH, '$'

PATH_MODULE DB 'Path of the loaded module:  ', 0DH, 0AH, '$'

SPACE DB '          ', 0DH, 0AH, '$'


; Процедуры

;-----

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

; Байт в AL переводится в два символа шест. числа в AX

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 цифра

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 в 10 с/с, SI - адрес поля младшей цифры

```

        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

```

```

PRINT_STRING PROC near
        push ax
        mov AH,09h
        int 21h
        pop ax
        ret
PRINT_STRING endp

```

```

MEMORY_ADDRESS PROC near
        mov ax, ds:[02h]
        mov di, offset SEGMENT_MEMORY_ADDRESS

```

```

        add di, 46
        call WRD_TO_HEX
        mov dx, offset SEGMENT_MEMORY_ADDRESS
        call PRINT_STRING
        mov dx, offset SPACE
        call PRINT_STRING
        ret
MEMORY_ADDRESS ENDP

```

```

ENV_ADDRESS PROC near
        mov ax, ds:[02ch]
        mov di, offset SEGMENT_ADDRESS_ENV
        add di, 39
        call WRD_TO_HEX
        mov dx, offset SEGMENT_ADDRESS_ENV
        call PRINT_STRING
        mov dx, offset SPACE
        call PRINT_STRING
        ret
ENV_ADDRESS ENDP

```

```

TAIL_COMMAND PROC near
        push cx
        xor cx, cx

        mov cl, ds:[80h]
        cmp cl, 0
        je empty

        mov dx, offset TAIL_COMMAND_STRING
        call PRINT_STRING

        mov si, 0

```



```

print_tail:
    mov dl, ds:[81h + si]
    mov ah, 02h
    int 21h
    inc si

    loop print_tail
    mov dx, offset SPACE
    call PRINT_STRING

    jmp end_tail

```

```

empty:
    mov dx, offset EMPTY_STRING
    call PRINT_STRING
    mov dx, offset SPACE
    call PRINT_STRING

```

```

end_tail:
    pop cx
    mov dx, offset SPACE
    call PRINT_STRING
    ret

```

```

TAIL_COMMAND ENDP

```

```

AREA_ENVIROMENT PROC near
    mov dx, offset AREA_ENV
    call PRINT_STRING

    mov ax, ds:[2ch]
    mov es, ax
    mov di, 0

```

```

area1:
    mov dl, es:[di]
    cmp dl, 0
    je newline

area2:
    mov ah, 02h
    int 21h
    inc di
    jmp area1

newline:
    mov dx, offset SPACE
    call PRINT_STRING
    inc di
    mov dl, es:[di]
    cmp dl, 0
    jne area2
    mov dx, offset NEW_STRING
    call PRINT_STRING
    ret

```

```

AREA_ENVIROMENT ENDP

```

```

PATH PROC near
    mov dx, offset PATH_MODULE
    call PRINT_STRING
    add di, 3

```

```

path1:
    mov dl, es:[di]
    cmp dl, 0
    je end_path
    mov ah, 02h
    int 21h

```

```
inc di  
jmp path1
```

```
end_path:  
ret
```

```
PATH ENDP
```

```
BEGIN:  
call MEMORY_ADDRESS  
call ENV_ADDRESS  
call TAIL_COMMAND  
call AREA_ENVIROMENT  
call PATH  
  
xor AL, AL  
mov AH, 4Ch  
int 21H
```

```
TESTPC ENDS
```

```
END START
```