

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

Звега А.Р.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

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

Задание.

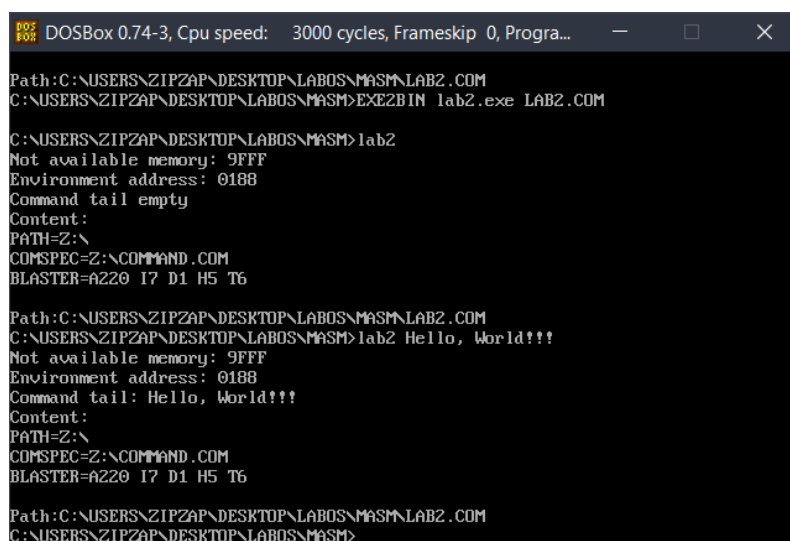
Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую информацию:

- 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.

Шаг 2. Оформление отчета в соответствии с требованиями. В отчет включите скриншот с запуском программы и результатами.

Выполнение работы

Была разработана .COM программа (исходный код в приложении А.). Примеры работы программы представлены на рисунке 1.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
Path:C:\USERS\ZIPZAP\DESKTOP\LABOS\MASM\LAB2.COM
C:\USERS\ZIPZAP\DESKTOP\LABOS\MASM>EXE2BIN lab2.exe LAB2.COM

C:\USERS\ZIPZAP\DESKTOP\LABOS\MASM>lab2
Not available memory: 9FFF
Environment address: 0188
Command tail empty
Content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Path:C:\USERS\ZIPZAP\DESKTOP\LABOS\MASM\LAB2.COM
C:\USERS\ZIPZAP\DESKTOP\LABOS\MASM>lab2 Hello, World!!!
Not available memory: 9FFF
Environment address: 0188
Command tail: Hello, World!!!
Content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Path:C:\USERS\ZIPZAP\DESKTOP\LABOS\MASM\LAB2.COM
C:\USERS\ZIPZAP\DESKTOP\LABOS\MASM>
```

Рисунок 1 - Пример работы программы

Контрольные вопросы.

Сегментный адрес недоступной памяти:

1) На какую область памяти указывает адрес недоступной памяти?

На адрес следующего сегмента в памяти, после программы.

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

После памяти, отведенной программе.

3) Можно ли в эту область памяти писать?

Можно, так как в DOS нет необходимых механизмов защиты от этого.

Среда передаваемая программе:

1) Что такое среда?

Среда – это участок памяти, хранящий системные переменных, пути и другие ресурсы операционной системы, передаваемые программе.

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

При загрузке ОС, а перед запуском приложения она может быть изменена в соответствии с требованиями приложения. Запуск программ осуществляется командным интерпретатором - `command.com`, который имеет свою среду. Когда одна программа запускает другую, то запущенная программа получает копию среды родителя, но можно создать другую среду.

3) Откуда берется информация, записываемая в среду?

Из системного файла `autoexec.bat`, при запуске ОС, который расположен в корневом каталоге загрузочного устройства.

Выводы.

Исследован интерфейс управляющей программы и загрузочных модулей. Изучен префикс сегмента программы (PSP) и среда, передаваемая программе

Приложение А.

Название файла: lab2.asm

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

NOT_AVAILABLE_STRING db "Not available memory: ",0dh,0ah,'\$'

ADDRESS_STRING db "Environment address: ",0dh,0ah,'\$'

COMMAND_EMPTY_STRING db "Command tail empty",'\$'

COMMAND_TAIL_STRING db "Command tail:",'\$'

NEW_LINE_STRING db 0dh,0ah,'\$'

CONTENT_STRING db "Content:",0dh,0ah,'\$'

PATH_STRING db "Path:",'\$'

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

```

```

WRITE_STR proc near
    push ax
    mov ah,9h
    int 21h
    pop ax
    ret
WRITE_STR ENDP

```

```

PATH proc near
    push ax
    push cx
    push dx
    push di

    mov dx,offset PATH_STRING
    call WRITE_STR
    add di,3

loop_path:
    mov dl,es:[di]
    cmp dl,0
    je end_of_path
    mov ah,02h
    int 21h
    inc di
    jmp loop_path

end_of_path:
    pop ax
    pop cx
    pop dx
    pop di
    ret
PATH ENDP

```

```

NOT_AVAILABLE_MEMORY proc near
    push ax
    push dx
    push di

    mov ax,ds:[02h]
    mov di,offset NOT_AVAILABLE_STRING
    add di,25
    call WRD_TO_HEX

```

```

        mov dx,offset NOT_AVAILABLE_STRING
        call WRITE_STR

        pop ax
        pop dx
        pop di
        ret
NOT_AVAILABLE_MEMORY ENDP

ENVIRONMENT_ADDRESS proc near
        push ax
        push dx
        push di

        mov ax,ds:[02ch]
        mov di,offset ADDRESS_STRING
        add di,24
        call WRD_TO_HEX
        mov dx,offset ADDRESS_STRING
        call WRITE_STR

        pop ax
        pop dx
        pop di
        ret
ENVIRONMENT_ADDRESS ENDP

TAIL proc near
        push ax
        push cx
        push dx
        push di

        mov cL,ds:[080h]
        cmp cL,0
        je print_empty

        mov dx,offset COMMAND_TAIL_STRING
        call WRITE_STR

        mov ch,0
        mov di,0
loop_tail:
        mov dl,ds:[081h+di]
        mov ah,02h
        int 21h

        inc di
        loop loop_tail
        jmp end_of_tail

```

```

print_empty:
    mov dx,offset COMMAND_EMPTY_STRING
    call WRITE_STR

end_of_tail:
    mov dx,offset NEW_LINE_STRING
    call WRITE_STR

    pop ax
    pop cx
    pop dx
    pop di
    ret

TAIL ENDP

CONTENT proc near
    push ax
    push cx
    push dx
    push di

    mov dx,offset CONTENT_STRING
    call WRITE_STR

    mov ax,ds:[2ch]
    mov es,ax
    mov di,0
cmp_new_line:
    mov dl,es:[di]
    cmp dl,0
    jz new_line
loop_new_line:
    mov ah,02h
    int 21h
    inc di
    jmp cmp_new_line
new_line:
    mov dx,offset NEW_LINE_STRING
    call WRITE_STR

    inc di
    mov dl,es:[di]
    cmp dl,0
    jnz loop_new_line

    mov dx,offset NEW_LINE_STRING
    call WRITE_STR
    call PATH

    pop ax
    pop cx

```

```
        pop dx
        pop di
        ret
CONTENT ENDP
```

BEGIN:

```
        call NOT_AVAILABLE_MEMORY
        call ENVIRONMENT_ADDRESS
        call TAIL
        call CONTENT

        xor AL,AL
        mov AH,4Ch
        int 21H
TESTPC ENDS
END START
```