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. 0382 _____ Крючков А.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

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.

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.

Сохранить результаты, полученные программой, и включить их в отчет.

Шаг 2. Оформить отчет в соответствии с требованиями. В отчет включить скриншот с запуском программы и результатами. Ответить на контрольные вопросы.

Выполнение работы.

Был написан и отлажен программный модуль типа **.COM**, который выбирает распечатывает необходимую в задании информацию.

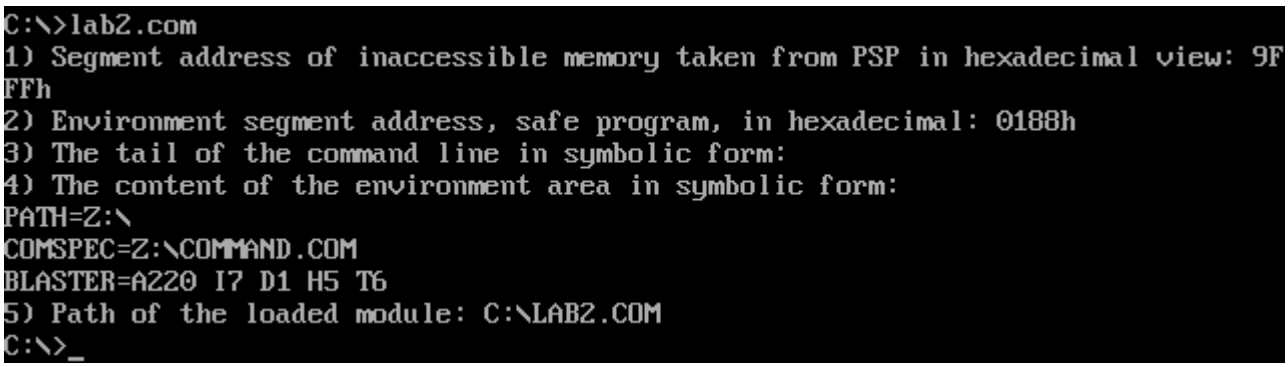

Рисунок 1. Пример работы программы

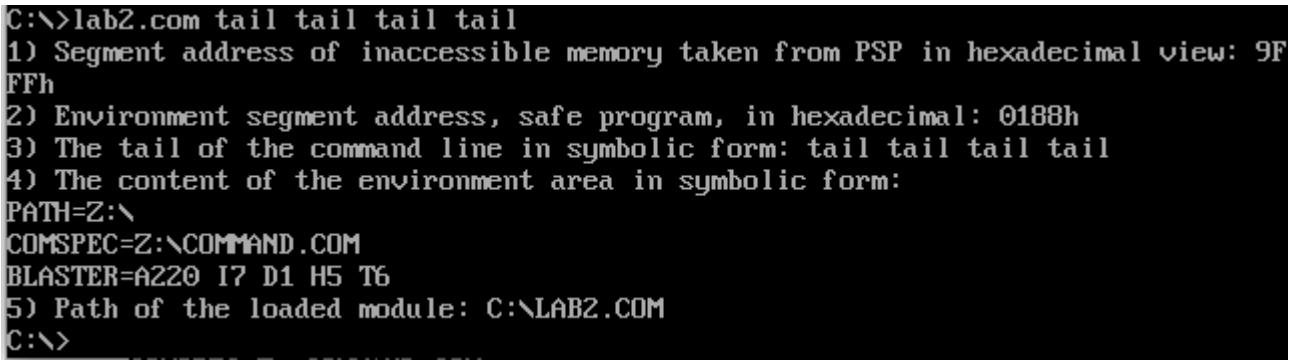

Рисунок 2. Пример работы программы с наличием символов в хвосте командной строки

Ответы на вопросы.

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной среды?

Ответ: На сегмент, находящийся после памяти, выделенной для .com программы.

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

Ответ: Сразу после памяти, выделенной под программу.

3. Можно ли в эту область памяти писать?

Ответ: Да — в DOS нет защиты от перезаписи памяти.

Среда передаваемая программе

1. Что такое среда?

Ответ: Среда — это область памяти, хранящая переменные среды, хранящие информацию о состоянии системы, получаемая программой при её загрузке.

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

Ответ: При запуске ОС.

3. Откуда берется информация, записываемая в среду?

Ответ: Эта информация берется из файла AUTOEXEC.BAT.

Вывод.

В ходе выполнения лабораторной работы было проведено исследование интерфейса управляющей программы и загрузочных модулей.

Приложение А

Файл com.asm

```
CODE SEGMENT
ASSUME CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING
ORG 100H
START: jmp BEGIN

SEG   DB "1) Segment address of inaccessible memory taken from PSP
in hexadecimal view:     h",0DH,0AH,'$'
ENV  DB "2) Environment segment address, safe program, in
hexadecimal:     h",0DH,0AH,'$'
TAIL DB "3) The tail of the command line in symbolic form:",'$'
ENV_CONTENT DB "4) The content of the environment area in symbolic
form: ",0DH,0AH,'$'
PATH DB "5) Path of the loaded module: ",'$'
NEWLINE   DB 0DH,0AH,'$'

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

PRINT PROC NEAR
push AX
mov AH, 09h
int 21h
pop AX
ret
PRINT ENDP

lab2 PROC NEAR
push AX
push CX
push DX
push DI
push ES
mov DX, offset SEG
mov DI, DX
add DI, 81
mov AX, CS:[2]
call WRD_TO_HEX
call PRINT
mov DX, offset ENV
mov DI, DX
add DI, 65
mov AX, CS:[2Ch]
call WRD_TO_HEX
call PRINT
mov DX, offset TAIL
call PRINT
xor CX,CX
mov CL, CS:[80h]
cmp CL, 0
mov AH, 02h
je lend
mov DI, 81h
printTail:
mov DL, CS:[DI]
int 21h
inc DI
loop printTail
lend:
mov DX, offset NEWLINE
call PRINT
mov DX, offset ENV_CONTENT
call PRINT
mov DX, CS:[2Ch]
mov ES, DX
mov DI, 0
```

```
next2:
mov DL, ES:[DI]
print2:
int 21h
inc DI
cmp DL, 0
jne next2
mov DX, offset NEWLINE
call PRINT
mov DL, ES:[DI]
cmp DL, 0
jne print2
mov DX, offset PATH
call PRINT
add DI, 3

next3:
mov DL, ES:[DI]
int 21h
inc DI
cmp DL, 0
jne next3
pop ES
pop DI
pop DX
pop CX
pop AX
ret
lab2 ENDP

BEGIN:
call lab2
xor AL,AL
mov AH,4Ch
int 21H
CODE ENDS
END START
```