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 _____ Мосин К.К.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей.

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 программы. 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.

Табл. 1 - 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.

Задание.

Шаг 1. Написать и отладить 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

Выполнение работы.

Был разработан модуль типа .COM. Пример работы программы представлен на рисунке 1.

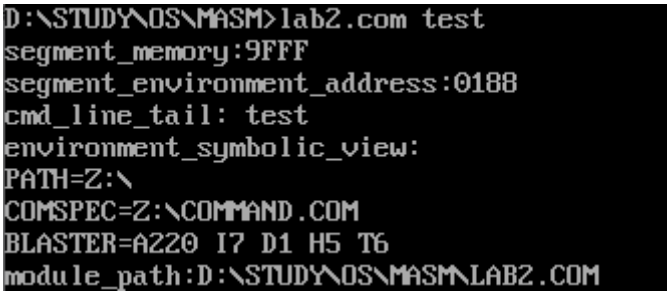

Рис 1. - Результат выполнения .COM модуля

Контрольные вопросы.

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

На начало сегмента, расположенный сразу после выделенной под программу памяти.

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

В PSP по смещению 2ch.

3) Можно ли в эту область памяти писать?

В DOS не существует защиты памяти, следовательно изменение участков памяти не составит труда.

Среда передаваемая программе

1) Что такое среда?

Участок памяти, предназначенный для хранения переменных и их значений в символьном виде.

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

Перед запуском приложения. Эта среда копируется в адресное пространство запущенной программы для соответствующих дальнейших изменений. Также, в программу, которую запустила другая программа, копируется среда родительской программы.

3) Откуда берется информация, записываемая в среду?

Существует файл autoexec.bat, который исполняет интерпретатор cmd command.com в DOS.

Выводы.

В ходе выполнения лабораторной работы была исследована структура интерфейса управляющей программы и загрузочных модулей

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab2.asm

```
testpc segment
assume cs:testpc,ds:testpc,es:nothing,ss:nothing
org 100h
start:
jmp begin

segment_memory db 'segment_memory:     ',0dh,0ah,'$'
segment_environment_address db 'segment_environment_address:     ',0dh,0ah,'$'
cmd_line_tail db 'cmd_line_tail:','$'
empty_cmd_line_tail db 'empty_cmd_line',0dh,0ah,'$'
caret_transfer db 0dh,0ah,'$'
environment_symbolic_view db 'environment_symbolic_view:',0dh,0ah,'$'
module_path db 'module_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],ah
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_l
or al,30h
mov [si],al
end_l:
pop dx
pop cx
ret
byte_to_dec endp

print proc near
push ax
mov ah,09h
int 21h
pop ax
ret
print endp

begin:
mov ax,ds:[02h]
mov di,offset segment_memory+18
call wrd_to_hex
mov dx,offset segment_memory
```

```
call print



mov ax,ds:[02ch]
mov di,offset segment_environment_address+31
call wrd_to_hex
mov dx,offset segment_environment_address
call print



xor di,di
mov cl,ds:[080h]
cmp cl,00h
je cmd_line_is_empty
mov dx,offset cmd_line_tail
call print
mov ah,02h

line_loop:
mov dl,ds:[081h+di]
int 21h
inc di
loop line_loop
mov dx,offset caret_transfer
call print
```

```
jmp exit_1

cmd_line_is_empty:
mov dx,offset empty_cmd_line_tail
call print




exit_1:
xor di,di
mov dx,offset environment_symbolic_view
call print
mov ax,ds:[2ch]
mov es,ax

environment_loop:
mov dl,es:[di]
cmp dl,00h
je environment_loop_end
mov ah,02h
int 21h
inc di
jmp environment_loop

environment_loop_end:
inc di
mov dl,es:[di]
cmp dl,00h
je exit_2
mov dx,offset caret_transfer
```

```
call print
jmp environment_loop



exit_2:
mov dx,offset caret_transfer
call print
mov dx,offset module_path
call print
add di,3
path_loop:
mov dl,es:[di]
cmp dl,00h
je exit_3
mov ah,02h
int 21h
inc di
jmp path_loop



exit_3:
xor al,al
mov ah,4ch
int 21h
testpc ends
end start
```