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

Студент гр. 9383 _____ Моисейченко К. А.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2021

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.

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.

Сохраните результаты, полученные программой, и включите их в отчет.

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

Контрольные вопросы по лабораторной работе.

Сегментный адрес недоступной памяти.

1) На какую область памяти указывает адрес недоступной памяти?

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

3) Можно ли в эту область памяти писать?

Среда, передаваемая программе.

1) Что такое среда?

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

3) Откуда берется информация, записываемая в среду?

Выполнение работы.

Шаг 1.

Была разработана .COM программа (исходный код см. в приложении А). Результат работы программы представлен на рисунке 1.

Рисунок 1 - Результат работы .COM программы

Ответы на вопросы:

Сегментный адрес недоступной памяти.

1) На какую область памяти указывает адрес недоступной памяти?

На адрес первого байта после памяти, отведенной под программу.

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

За памятью, отведенной программе, тоесть в сторону увеличения адресов.

3) Можно ли в эту область памяти писать?

Можно, так как DOS не располагает механизмами защиты от редактирования программами не отведенной им памяти.

Среда, передаваемая программе.

1) Что такое среда?

Среда - это совокупность значений системных переменных, путей, открытых файловых дескрипторов и других ресурсов операционной системы, передаваемых программе.

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

Изначальная среда создается при загрузке ОС, но перед запуском приложения она может быть изменена в соответствии с требованиями приложения.

3) Откуда берется информация, записываемая в среду?

Информация берется из системного файла AUTOEXEC.BAT, который расположен в корневом каталоге загрузочного устройства.

Выводы.

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

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab2_com.asm

```
TESTPC SEGMENT
ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
ORG 100H
START: JMP BEGIN
; ДАННЫЕ

PC db 'Type of PC : PC' , 0dh, 0ah,'$'
PC_XT db 'Type of PC : PC/XT' , 0dh, 0ah, '$'
AT db 'Type of PC : AT', 0dh, 0ah, '$'
PS2_M30 db 'Type of PC : PS2 model 30', 0dh, 0ah, '$'
PS2_M50_M60 db 'Type of PC : PS2 model 50 or 60', 0dh,0ah,'$'
PS2_M80 db 'Type of PC : PS2 model 80', 0dh,0ah,'$'
PCjr  db 'Type of PC : PCjr',0dh,0ah,'$'
PC_CONV db 'Type of PC: PC Convertible' , 0dh, 0ah, '$'
ERROR db 'Error ', 0dh,0ah,'$'
DOS db 'DOS version:  .  ',0dh,0ah,'$'
OEM db 'OEM number:   ',0dh,0ah,'$'
USER db 'User serial number:       h',0dh,0ah,'$'

;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 шестн. 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
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 10с/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_l
or AL,30h
mov [SI],AL
end_l: pop DX
pop CX
ret
BYTE_TO_DEC ENDP
;--------------------------------
; КОД

PC_TYPE PROC near
mov ax, 0f000h
mov es,ax
mov di, 0fffeh
mov ah, es:[di]

cmp ah, 0FFh
jne PRINT_PC_XT_1
mov dx, offset PC
jmp PRINT_TYPE

PRINT_PC_XT_1:
cmp ah, 0FEh
jne PRINT_PC_XT_2
mov dx,offset PC_XT
jmp PRINT_TYPE

PRINT_PC_XT_2:
cmp ah,0FBh
```

```
jne PRINT_AT
mov dx, offset PC_XT
jmp PRINT_TYPE

PRINT_AT:
cmp ah,0FCh
jne PRINT_PS2_M30
mov dx,offset AT
jmp PRINT_TYPE

PRINT_PS2_M30:
cmp ah,0FAh
jne PRINT_PS2_M50_M60
mov dx, offset PS2_M30
jmp PRINT_TYPE

PRINT_PS2_M50_M60:
cmp ah,0FCh
jne PRINT_PS2_M80
mov dx,offset PS2_M50_M60
jmp PRINT_TYPE

PRINT_PS2_M80:
cmp ah,0F8h
jne PRINT_PCjr
mov dx,offset PCjr
jmp PRINT_TYPE

PRINT_PCjr:
cmp ah,0FDh
jne PRINT_PC_CONV
mov dx, offset PC_CONV
jmp PRINT_TYPE

PRINT_PC_CONV:
cmp ah,0FDh
jne PRINT_ERROR
mov dx, offset PC_CONV
jmp PRINT_TYPE

PRINT_ERROR:
mov dx,offset ERROR
jmp PRINT_TYPE

PRINT_TYPE:
mov ah,9h
int 21h

ret
PC_TYPE ENDP

OS_TYPE PROC near

mov ah,30h
int 21h
```

```
mov si,offset DOS
add si,13
push ax
call BYTE_TO_DEC

pop ax
add si,3
mov al,ah
call BYTE_TO_DEC

mov dx,offset DOS
mov ah,9h
int 21h

mov si,offset OEM
add si,14
mov al,bh
call BYTE_TO_DEC

mov dx, offset OEM
mov ah,9h
int 21h

mov di, offset USER
add di,25
mov ax,cx
call WRD_TO_HEX
mov al,bl
call BYTE_TO_HEX
mov [di-2],ax

mov dx,offset USER
mov ah,9h
int 21h

ret
OS_TYPE ENDP

BEGIN:
call PC_TYPE
call OS_TYPE

; Выход в DOS
xor AL,AL
mov AH,4Ch
int 21H
TESTPC ENDS
END START ;конец модуля, START - точка входа
```