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. 0382 _____ Сергеев Д.А.

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.

Задание.

Написать и отладить программный модуль .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.

Выполнение работы.

В ходе работы был взят шаблон из методических материалов, также были написаны процедуры:

- IncMem выводит на консоль сегментный адрес недоступной памяти.
- SegAd выводит на консоль сегментный адрес среды, передаваемой программе.
- PrintSym выводит на консоль символ, который находится в регистре dl.
- CmdTail выводит на консоль хвост командной строки, сначала в регистр cx поступает кол-во символов в хвосте, а затем хвост

считывается посимвольно, после чего с помощью процедуры PrintSym, выводится на экран.

Рисунок 1 – Результат работа программы, если имеется хвост командной строки

Рисунок 2 – Результат работы программы, если хвоста командной строки нет

- Env выводит на консоль содержимое области среды, как и в предыдущей процедуре символы считываются по одному начиная с определенного адреса, разделением между строками является нулевой байт, а конец области среды – два нулевых байта.
- Mpath выводит на консоль путь загружаемого модуля, эти данные хранятся через 2 байта после конца области среды, так что к адресу оставшемуся после процедуры Env добавляется 2, и затем посимвольно печатается путь

Исходный программный код смотрите в приложении А.

Контрольные вопросы.

1. Сегментный адрес недоступной памяти
   - На какую область памяти указывает адрес недоступной памяти?

Ответ: адрес недоступной памяти указывает на первый байт после области памяти, которая была отведена на программу.

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

Ответ: Адрес расположен сразу после области памяти, отведенной программе (начиная с адреса 9FFF)

- Можно ли в эту область памяти писать?

Ответ: Можно, т.к. DOS не имеет механизмов защиты от перезаписи памяти различными программами

2. Среда передаваемая программе
    - Что такое среда?

    Ответ: Среда – область памяти, хранящая информацию о состоянии системы в формате: имя=параметр

    - Когда создаётся среда? Перед запуском приложения или в другое время?

    Ответ: В начале среда создаётся при запуске операционной системы, а уже при запуске программы текущая среда копируется в адресное пространство программы.

    - Откуда берется информация, записываемая в среду?

    Ответ: Для DOS информация берётся из специального системного файла – autoexec.bat, который располагается в корневой директории

Выводы.

В результате выполнения работы, был исследован интерфейс управляющей программы и загрузочных модулей, префикс сегмента программы (PSP) и среда передаваемая программе.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: prog.asm

```
TESTPC SEGMENT
   ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
   ORG 100H
START: JMP BEGIN
; Данные
PC db  'Your PC type is -> PC',0DH,0AH,'$'
InacMemory db 'Inaccessable memory:     h',0DH,0AH,'$'
SegAdr db 'Segment Address:     h',0DH,0AH,'$'
Tail db 'Tail:','$'
Enviroment db 'Enviroment:',0DH,0AH,'$'
ModulePath db 'Path: ','$'

; 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
   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
;-------------------------------
WRITESTRING PROC near
   mov AH,09h
   int 21h
   ret
WRITESTRING ENDP

IncMem PROC near
   mov ax,ds:[2]
   mov di, offset InacMemory
   add di,24
   call WRD_TO_HEX
   mov dx, offset InacMemory
   call WRITESTRING
   ret
IncMem ENDP

SegAd PROC near
   mov ax,ds:[2Ch]
   mov di, offset SegAdr
   add di,20
   call WRD_TO_HEX
   mov dx, offset SegAdr
   call WRITESTRING
   ret
SegAd ENDP

PrintSym PROC near
   push ax
   mov ah,02h
   int 21H
   pop ax
   ret
```

```
PrintSym ENDP

CmdTail PROC near
   mov cl,ds:[80h]
   mov dx,offset Tail
   call WRITESTRING
   cmp cl,0
   je end_t
   mov di,81h
metka:
   mov dl,ds:[di]
   call PrintSym
   inc di
   loop metka
end_t:
   mov dl,0DH
   call PrintSym
   mov dl,0AH
   call PrintSym
   ret
CmdTail ENDP

Env PROC near
   mov es,ds:[2Ch]
   xor di,di
   mov dx,offset Enviroment
   call WRITESTRING
metka2:
   mov dl,es:[di]
   cmp dl,0
   je end_e
   call PrintSym
   inc di
   jmp metka2
end_e:
   mov dl,0DH
   call PrintSym
   mov dl,0AH
   call PrintSym
```

```
   inc di
   mov dl,es:[di]
   cmp dl,0
   jne metka2
   ret
Env ENDP

MPath PROC near
   add di,3
   mov dx, offset ModulePath
   call WRITESTRING
metka3:
   mov dl,es:[di]
   cmp dl,0
   je end_m
   call PrintSym
   inc di
   jmp metka3
end_m:
   mov dl,0DH
   call PrintSym
   mov dl,0AH
   call PrintSym
   ret
Mpath ENDP

; Код
BEGIN:
   call IncMem
   call SegAd
   call CmdTail
   call Env
   call MPath
   xor AL,AL
   mov AH,4Ch
   int 21H
TESTPC ENDS
END START
```