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 _____ Крейсманн К.В.

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.

Рисунок 1 - Примеры работы программы

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ ПО ЛАБОРАТОРНОЙ №2

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

На адрес следующего сегмента в памяти, расположенного после программы.

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

В PSP по смещению 02h.

3) Можно ли в эту область памяти писать?

Можно, т.к. DOS не располагает необходимыми механизмами защиты от этого.

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?

Изначально среда создается при загрузке ОС, но перед запуском приложения она может быть изменена в соответствии с требованиями приложения. Запуск программ осуществляется командным интерпретатором - command.com, который имеет свою среду, которая называется корневой. Когда одна программа запускает другую, то запущенная программа получает свой собственный экземпляр блока среды, точно такой же как и у родителя, хотя при надобности можно создать совершенно другую среду.

3) Откуда берется информация, записываемая в среду?

Из системного файла autoexec.bat, при запуске ОС. Он 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.

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

;ДАННЫЕ

UNAVAILABLE_STRING db "Unavailable memory:     ",0dh,0ah,'$'
ADDRESS_STRING db "Environment address:     ",0dh,0ah,'$'
COMMAND_EMPTY_STRING db "Command tail is empty",'$'
COMMAND_TAIL_STRING  db "Command tail:",'$'
NEWLINE_STRING db 0dh,0ah,'$'
CONTENT_STRING db "Content:",0dh,0ah,'$'
SPACE_STRING db "    ",'$'
PATH_STRING db "Path:",'$'

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
; байт в AL преводится 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- числа, 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

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

```
            ret
WRITE_STR ENDP

UNAVAILABLE_MEMORY proc near
            mov ax,ds:[02h]
            mov di,offset UNAVAILABLE_STRING
            add di,23
            call WRD_TO_HEX
            mov dx,offset UNAVAILABLE_STRING
            call WRITE_STR
            ret
UNAVAILABLE_MEMORY ENDP

ENVIRONMENT_ADDRESS proc near
            mov ax,ds:[02ch]
            mov di,offset ADDRESS_STRING
            add di,24
            call WRD_TO_HEX
            mov dx,offset ADDRESS_STRING
            call WRITE_STR
            ret
ENVIRONMENT_ADDRESS ENDP

COMMAND_TAIL proc near
            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
metka:
            mov dl,ds:[081h+di]
            mov ah,02h ;для вывода одного символа
            int 21h

            inc di
            loop metka
            jmp end_of_proc

print_empty:
            mov dx,offset COMMAND_EMPTY_STRING
            call WRITE_STR

end_of_proc:
            mov dx,offset NEWLINE_STRING
            call WRITE_STR
            ret
COMMAND_TAIL ENDP

CONTENT proc near
            mov dx,offset CONTENT_STRING
            call WRITE_STR
            mov dx,offset SPACE_STRING
            call WRITE_STR

            mov ax,ds:[2ch]
            mov es,ax
            mov di,0
metka1:
            mov dl,es:[di]
            cmp dl,0
```

```
              je newline_metka
metka2:
              mov ah,02h
              int 21h
              inc di
              jmp metka1
newline_metka:
              mov dx,offset NEWLINE_STRING
              call WRITE_STR
              mov dx,offset SPACE_STRING
              call WRITE_STR

              inc di
              mov dl,es:[di]
              cmp dl,0
              jne metka2

              mov dx,offset NEWLINE_STRING
              call WRITE_STR
              call PATH
              ret
CONTENT ENDP

PATH proc near
              mov dx,offset PATH_STRING
              call WRITE_STR

              add  di,3 ;добавляем 3, так как после среды идут два байта 00h,01h, а затем
маршрут
metka3:
              mov dl,es:[di]
              cmp dl,0
              je end_of_proc2
              mov ah,02h
              int 21h
              inc di
              jmp metka3

end_of_proc2:
              ret
PATH ENDP


BEGIN:
              call UNAVAILABLE_MEMORY
              call ENVIRONMENT_ADDRESS
              call COMMAND_TAIL
              call CONTENT
; Выход в DOS
              xor AL,AL
              mov AH,4Ch
              int 21H
TESTPC ENDS
                        END START ;конец модуля, START - точка входа
```