МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №2

по дисциплине «Операционные системы»

ТЕМА: ИССЛЕДОВАНИЕ ИНТЕРФЕЙСОВ ПРОГРАММНЫХ МОДУЛЕЙ

Студентка гр. 7383 _____ Чемова К.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

Постановка задачи.

Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системный данные. Так загрузчик строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Ход работы.

Был написан командный модуль типа .COM, использующий следующие функции:

- TETR_TO_HEX – переводит число в шестнадцатеричную систему счисления из двоичной;
- BYTE_TO_HEX – переводит байтовое число из регистре AL в шестнадцатеричную систему счисления
- WRD_TO_HEX – переводит число из регистре AX в шестнадцатеричную систему счисления;
- BYTE_TO_DEC – переводит число в двоичную систему счисления;
- SEG_ADDRESS_INACCESS_MEM – получает сегментный адрес недоступной памяти;
- SEG_ADDRESS_ENV – получает сегментный адрес среды;
- TAIL – получает хвост командной строки
- ENV_CONTENT – получает содержимое области среды и выводит его на экран;
- PATH – получает путь загружаемого модуля и выводит его на экран.

На рис. 1 и 2 представлена работа программы.

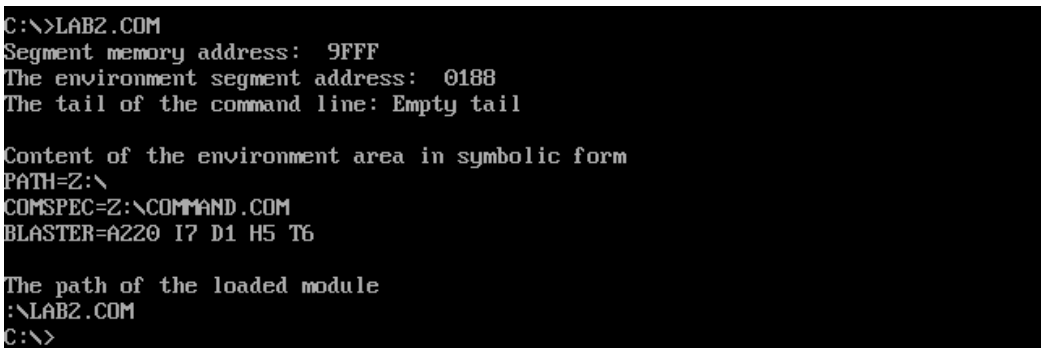

Рисунок 1 – Работа программы

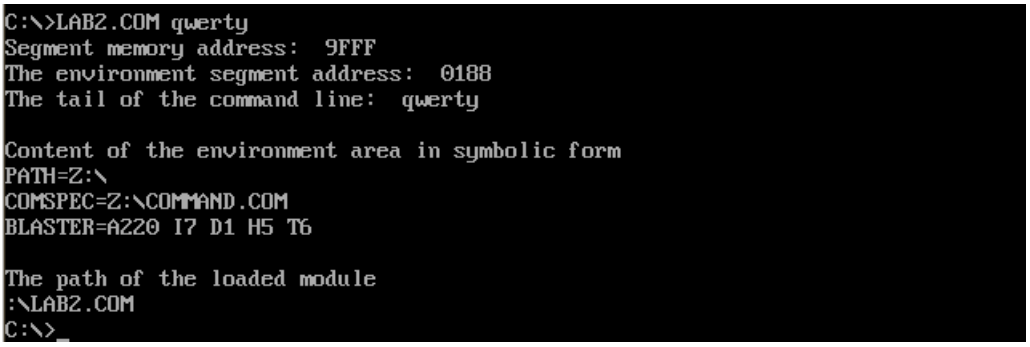

Рисунок 2 – Хвост командной строки

Выводы.

В ходе выполнения лабораторной работы были исследованы интерфейс управляющей программы и загрузочных модулей. Был написан загружаемый модуль типа .COM. Код программы представлен в приложении А.

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на границу оперативной памяти и на границу области, доступной для загрузки программ.

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

Данный адрес расположен сразу после отведенной программе области памяти.

3. Можно ли в эту область памяти писать?

Можно, из-за отсутствия в DOS защиты памяти.

Среда, передаваемая программе

1. Что такое среда?

Среда представляет собой область памяти, в которой в виде символьных строк записаны значения переменных, называемых переменными среды.

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

Среда создается при загрузке DOS, а при запуске приложения копируется в новую область памяти.

3. Откуда берется информация, записываемая в среду?

Информация берется из системного файла autoexec.bat.

ПРИЛОЖЕНИЕ А

lab2.ASM

```
TESTPC SEGMENT
         ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
         ORG 100H
START: JMP BEGIN
;-------------------------------------------------------------------------------
;ДАННЫЕ
mem_add      db 'Segment memory address:     ', 0DH, 0AH, '$'
env_add      db 'The environment segment address:     ', 0DH, 0AH, '$'
env_cont     db 'Content of the environment area in symbolic form      ', 0DH, 0AH,
'$'
m_path       db 'The path of the loaded module      ', 0DH, 0AH, '$'
tail_str     db 'The tail of the command line: ','$'
empty_tail   db 'Empty tail', 0DH, 0AH, '$'
endl         db ' ', 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 переводиться в два символа шестнадцатеричного 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
       pop   CX
       ret
BYTE_TO_HEX  ENDP
;-------------------------------------------------------------------------------
WRD_TO_HEX  PROC near
;перевод в 16 СС 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 10 СС, 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
;--------------------------------------------------------------------------------
PRINT_MSG  PROC near
;вывод строки на экран
       push   AX
       mov    AH, 09h
       int    21H
       pop    AX
       ret
PRINT_MSG  ENDP
;--------------------------------------------------------------------------------
SEG_ADDRESS_INACCESS_MEM  PROC near
;получение сегментного адреса недоступной памяти
       push   AX
       push   DI
       mov    AX, DS:[02h]
       mov    DI, OFFSET mem_add
       add    DI, 28
       call   WRD_TO_HEX
       pop    DI
       pop    AX
       ret
SEG_ADDRESS_INACCESS_MEM  ENDP
;--------------------------------------------------------------------------------
SEG_ADDRESS_ENV  PROC near
;получение сегментного адреса среды
       push   AX
       push   DI
       mov    AX, DS:[2Ch]
       mov    DI, OFFSET env_add
       add    DI, 37
       call   WRD_TO_HEX
       pop    DI
       pop    AX
       ret
SEG_ADDRESS_ENV  ENDP
```

```
;-------------------------------------------------------------------------------
TAIL  PROC near
;получение хвоста командной строки
       push   AX
       push   CX
       push   DX
       push   SI
       mov    DX, OFFSET tail_str
       call   PRINT_MSG
       mov    CL, DS:[80h]
       cmp    CL, 00h
       je     empty_str
       mov    SI, 81h
       mov    AH, 02h
str_content:
       mov    DL, DS:[SI]
       int    21h
       inc    SI
       loop   str_content
       mov    DX, OFFSET endl
       call   PRINT_MSG
       jmp    str_end
empty_str:
       mov    AL, 00h
       mov    [DI], AL
       mov    DX, OFFSET empty_tail
       call   PRINT_MSG
str_end:
       pop    SI
       pop    DX
       pop    CX
       pop    AX
       ret
TAIL  ENDP
;-------------------------------------------------------------------------------
ENV_CONTENT  PROC near
;получение содержимого области среды
       push   AX
       push   DX
       push   DS
       push   ES
       mov    DX, OFFSET env_cont
       call   PRINT_MSG
       mov    AH, 02h
       mov    ES, DS:[2Ch]
       xor    SI, SI
environment1:
       mov    DL, ES:[SI]
       int    21h
       cmp    DL, 00h
       je     environment2
       inc    SI
       jmp    environment1
environment2:
       mov    DX, OFFSET endl
       call   PRINT_MSG
       inc    SI
       mov    DL, ES:[SI]
       cmp    DL, 00h
       jne    environment1
       mov    DX, OFFSET endl
```

```
       call   PRINT_MSG
       pop    ES
       pop    DS
       pop    DX
       pop    AX
       ret
ENV_CONTENT  ENDP
;-------------------------------------------------------------------------------
PATH  PROC near
;получение пути загружаемого модуля
       push   AX
       push   DX
       push   DS
       push   ES
       mov    DX, OFFSET m_path
       call   PRINT_MSG
       add    SI, 04h
       mov    AH, 02h
       mov    ES, DS:[2Ch]
path_:
       mov    DL, ES:[SI]
       cmp    DL, 00h
       je     _path_
       int    21h
       inc    SI
       jmp    path_
_path_:
       pop     ES
       pop    DS
       pop    DX
       pop    AX
       ret
PATH  ENDP
;-------------------------------------------------------------------------------
--------
BEGIN:
       call   SEG_ADDRESS_INACCESS_MEM
       mov    DX, OFFSET mem_add
       call   PRINT_MSG
       call   SEG_ADDRESS_ENV
       mov    DX, OFFSET env_add
       call   PRINT_MSG
       call   TAIL
       mov    DX, OFFSET endl
       call   PRINT_MSG
       call   ENV_CONTENT
       call   PATH
;ВЫХОД ИЗ DOS
;-------------------------------------------------------------------------------
       xor    AL, AL
       mov    AH, 4CH
       int    21h
TESTPC  ENDS
        END  START
```