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. 9382 _____ Савельев И.С.

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 префикс 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.

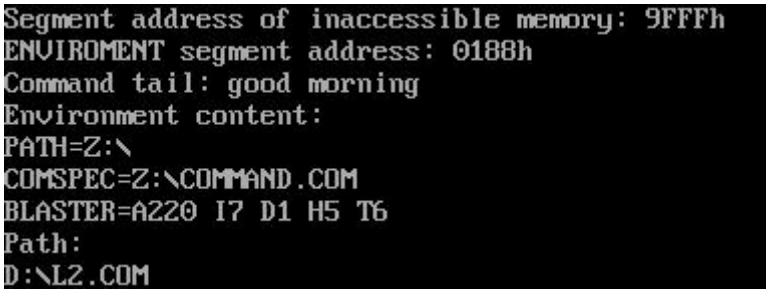
Шаг 2. Оформление отчета в соответствии с требованиями. В отчет включить скриншот с запуском программы и результатами.

Выполнение работы.

В результате выполнения лабораторной работы была написана программа, которая выводит сегментный адрес недоступной памяти, сегментный адрес среды, хвост командной строки, содержимое области среды и путь загружаемого модуля. Исходный код программы приведен в приложении Б.

Пример работы программы без аргументов.

Пример работы программы с аргументами



Вывод.

В процессе выполнения данной лабораторной работы были исследованы интерфейс управляющей программы, загрузочные модули, префикс сегмента программы и среды, передаваемой программе.

Приложение А. Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти:

1) На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на адрес следующего сегмента памяти, идущего после участка памяти отведенного для программы.

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

Этот адрес находится сразу за выделенным для программы сегментом. В PSP со смещением 2CH.

3) Можно ли в эту область памяти писать?

Да, можно, так как у DOS нет механизмов защиты от перезаписи не отведенной программам памяти.

Среда, передаваемая программе:

1) Что такое среда?

Среда – это область памяти, хранящая информацию, необходимую для работы ОС и приложений. В среде находятся переменные, которые содержат информацию о состоянии системы.

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

Среда создается при запуске ОС. Эта среда копируется и передается в адресное пространство запущенной программе, она может быть изменена в соответствии с потребностями запущенной программы.

3) Откуда берется информация, записываемая в среду?

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

Приложение Б. Исходный код программы.

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, ds:TESTPC, ES:NOTHING, SS:NOTHING
     ORG 100H

START: jmp BEGIN

; Данные

UNAVAILABLE_MEM_STR        db 'Segment address of inaccessible memory:
     h',0DH,0AH,'$'
ENVIROMENT_STR             db 'ENVIROMENT segment address:
h',0DH,0AH,'$'
TAIL_COMMAND_STR           db 'Command tail:                ',0DH,0AH,'$'
PATH_STR                        db 'Path:
',0DH,0AH,'$'
EMPTY_TAIL_STR             db 'Command line tail is empty!',0DH,0AH,'$'
NEW_STR                    db                                    0DH,
0AH, '$'
ENVIROMENT_CONTENT_STR     db   'Environment         content:         ',
     '$'

; 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

MYPRINTS PROC near
    push ax
       mov ah, 09h
       int 21h
    pop ax
       ret
MYPRINTS ENDP

MYPRINT PROC near
    push ax
```

```
      mov ah, 02h
      int 21h
    pop ax
      ret
MYPRINT ENDP

UNAVAILABLE_MEM PROC near
      mov ax, ds:[02h]
      mov di, offset UNAVAILABLE_MEM_STR
      add di, 43
      call WRD_TO_HEX
      mov dx, offset UNAVAILABLE_MEM_STR
      call MYPRINTS
      ret
UNAVAILABLE_MEM ENDP

ENVIROMENT PROC near
    mov ax, ds:[2Ch]
      mov di, offset ENVIROMENT_STR
      add di, 31
      call WRD_TO_HEX
      mov dx, offset ENVIROMENT_STR
      call MYPRINTS
      ret
ENVIROMENT ENDP

TAIL proc near
     push cx
     xor cx, cx
    mov cl, ds:[80h]
    mov si, offset TAIL_COMMAND_STR
    add si, 13
     cmp cl, 0h
     je empty
    xor di, di
    xor ax, ax

tail_loop:
    mov al, ds:[81h+di]
     inc di
     mov [si], al
    inc si
    loop tail_loop
    mov dx, offset TAIL_COMMAND_STR
    jmp end_tail

empty:
    mov dx, offset EMPTY_TAIL_STR

end_tail:
     call MYPRINTS
```

```
      pop cx
      ret
TAIL ENDP


ENVIROMENT_CONTENT PROC near
      push dx
      push ax
      push si
      push ds
      xor si, si
      mov dx, offset ENVIROMENT_CONTENT_STR
      call MYPRINTS
      mov dx, offset NEW_STR
      call MYPRINTS
      mov ds,ds:[2CH]

read_enviroment:
      mov dl,[si]
      cmp dl,0
      je end_line
      call MYPRINT
      inc si
      jmp read_enviroment

end_line:
      inc si
      mov dl,[si]
      cmp dl,0
      je end_read

      pop ds
      mov dx, offset NEW_STR
      call MYPRINTS
      push ds
      mov ds,ds:[2Ch]

      jmp read_enviroment

end_read:
      pop ds
      mov dx, offset NEW_STR
      call MYPRINTS
      mov dx, offset PATH_STR
      call MYPRINTS
      push ds
      mov ds,ds:[2Ch]
      add si, 3

read_pth:
      mov dl,[si]
```

```
      cmp dl,0
      je leave_point

      call MYPRINT
      inc si
      jmp read_pth

leave_point:
      pop ds
      pop si
      pop ax
      pop dx
      ret

ENVIROMENT_CONTENT ENDP

; КОД
BEGIN:
                  call UNAVAILABLE_MEM
                  call ENVIROMENT
                  call TAIL
            call ENVIROMENT_CONTENT

             xor         al,al
             mov         AH,4Ch
             int         21H
TESTPC       ENDS
             END      START
```