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

Студентка гр. 9381 _____ Москаленко Е.М.

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.

Функции и структуры данных.

В программе используются следующие процедуры:

Названия функций	Описание
TETR_TO_HEX	Перевод четырех младших битов регистре AL в 16-ричную цифру.
BYTE_TO_HEX	Перевод байта из AL в число 16-ной с.с. Символы записываются в регистры AL и AH.
WRD_TO_HEX	Перевод слова из AH в число в 16-ной с.с. Записывается в виде 4 символов по адресу из DI.
PRINT	Вывод строки на экран при помощи функции 9h прерывания 21h.

Последовательность действий.

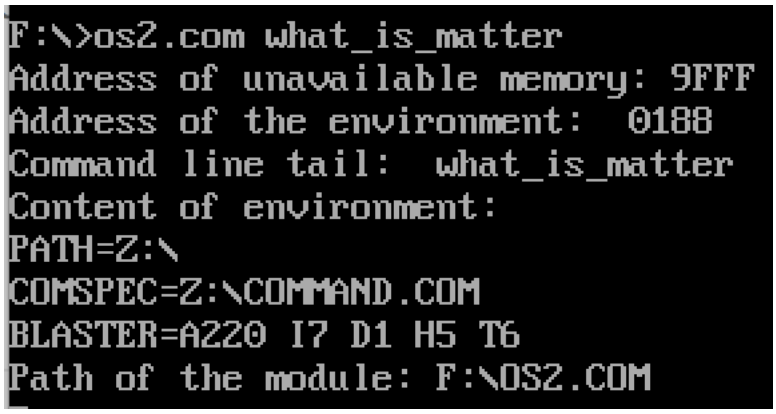
1) Сначала программа выводит адрес недоступной памяти в 16-ричном виде, взятой из PSP (смещение 2h). Перевод осуществляется посредством вызова процедуры BYTE_TO_HEX.
2) Затем программа выводит сегментный адрес среды, передаваемой программе, в 16-ричном виде (смещение 2Ch).
3) После этого выводится хвост командной строки в символьном виде. По смещению 80h программа считывает количество символом в хвосте. Если он пустой – выводит об этом сообщение.
4) Далее программа выводит содержимое области среды в символьном виде, используя метки ENVIRONMENT, READ, END_LINE и END_.
5) Затем программа выводит на экран путь загружаемого модуля, используя метки READING_PATH, CYCLE_PATH, END_CYCLE.

Ход работы.

Был написан текст исходного COM модуля os2.asm. Далее был отлажен модуль os2.com и загрузочный файл был протестирован.

Тест 1. Без аргументов.

Тест 2. С аргументами.



Результаты исследования проблем.

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на первый байт после памяти, выделенной программе.

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

Адрес расположен в префиксе сегмента программы PSP по смещению 2h.

3. Можно ли в эту область памяти писать?

Можно, потому что в DOS общее адресное пространство.

Среда, передаваемая программе

1. Что такое среда?

Среда (блок окружения) – совокупность переменных окружения.

Переменная окружения – символьная строка в коде ASCII, имеющая структуру:

<имя переменной> = <первое значение>; … ;<последнее значение>00h

Переменные хранят информацию о состоянии системы.

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 этого приложения.

3. Откуда берется информация, записываемая в среду?

Из системного пакетного файла AUTOEXEC.BAT, расположенном в корневом каталоге загрузочного устройства.

Выводы.

В ходе исполнения работы был исследован интерфейс управляющей программы и загрузочных модулей. Также исследован префикс сегмента программы PSP и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

Файл os2.asm

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

UNAVAILABLE_ADDRESS db  'Address of unavailable memory:    ',0DH, 0AH,
'$'
ENVIROMENT_ADDRESS  db  'Address of the environment:      ',0DH, 0AH,
'$'
TAIL_MESSAGE        db  'Command line tail: ','$'
NO_TAIL             db  'No arguments',0DH, 0AH, '$'
ENVIRONMENT_MESSAGE db  'Content of environment: ',0DH, 0AH,'$'
PATH_MESSAGE        db  'Path of the module: ','$'
PATH_MODULE         db   83 DUP(0DH,'$')
TAIL                db   83 DUP(0DH, 0AH,'$')
ENVIRONMENT_CONTENT db   128 DUP('$')

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 AL переводится в два символа 16с.с. 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
□□□□□□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

        PRINT  PROC NEAR    ; вывод строки на экран
              push ax
              mov ah, 9h
              int 21h
              pop ax
              ret
        PRINT ENDP

BEGIN:
  mov ax, ds:[2h]
  lea di, UNAVAILABLE_ADDRESS
```

```
  add di, 34
      call WRD_TO_HEX
      lea dx, UNAVAILABLE_ADDRESS
      call PRINT

  mov ax, ds:[2Ch]
  lea di, ENVIROMENT_ADDRESS
  add di, 32
      call WRD_TO_HEX
      lea dx,  ENVIROMENT_ADDRESS
      call PRINT

  lea dx, TAIL_MESSAGE
  call PRINT
  mov cl, ds:[80h]
  cmp cl, 0
  je EMPTY
  mov si, 0
  lea bx, TAIL

WRITE_TAIL:
     mov dl, ds:[81h+si]
     mov [bx+si], dl
     inc si
     loop WRITE_TAIL
     lea dx, TAIL
     call PRINT
     jmp ENVIRONMENT

EMPTY:
    lea dx, NO_TAIL
    call PRINT

ENVIRONMENT:
    lea dx, ENVIRONMENT_MESSAGE
    call PRINT
    lea di, ENVIRONMENT_CONTENT
    mov         ax, ds:[2Ch]
```

```
    mov         ds, ax
    mov si, 0

READ:
    lodsb
    cmp    al, 0
    jne    END_

END_LINE:
    mov al, 0Ah
    stosb
    lodsb
    cmp    al, 0h
    jne END_
    mov byte ptr [di], 0Dh
    mov byte ptr [di+1], '$'
    mov bx, ds
    mov ax, es
    mov ds, ax
    lea dx, ENVIRONMENT_CONTENT
    call PRINT
    jmp READING_PATH

END_:
    stosb
    jmp READ


READING_PATH:
     lea dx, PATH_MESSAGE
     call PRINT
     add si, 2
     mov  ds, ds:[2Ch]
     lea di, PATH_MODULE
CYCLE_PATH:
     lodsb
     cmp  al, 0
     je  END_CYCLE
```

```
    stosb
    jmp     CYCLE_PATH
END_CYCLE:
    mov  bx, ds
    mov  ax, es
    mov  ds, ax
    lea dx, PATH_MODULE
    call PRINT

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