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 _____ Герасев Г.

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 префикса сегмента программы (PSP) и среды, передаваемой программе.

Сведения о функциях и структурах.

TETR_TO_HEX: процедура перевода из 10-ой сс в символы

BYTE_TO_HEX: процедура перевода байта из 16-ой сс в символы

WRD_TO_HEX: перевод слова из 16-ой сс в символы

BYTE_TO_DEC: перевод байта из 16-ой сс в 10-юю и символы

PRINT: процедура вывода строки в терминал

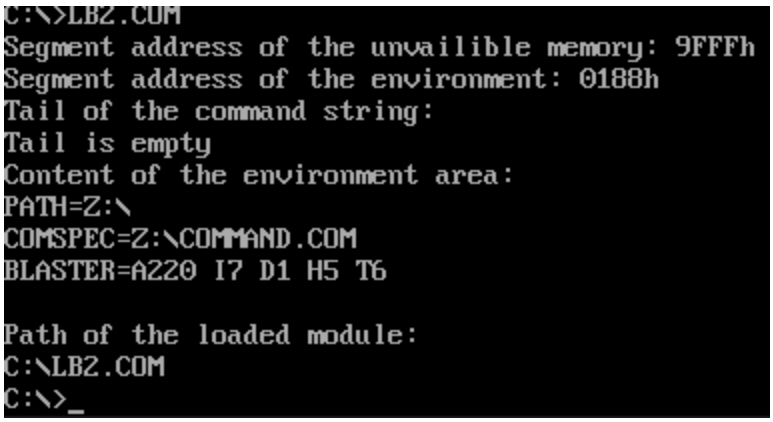
GIVE_INFO: процедура вывода требуемой информации из PSP

Последовательность действий, выполняемых утилитой.

1. Выводится сегментный адрес недоступной памяти, взятый из PSP с использованием процедуры WRD_TO_HEX.
2. Выводится сегментный адрес среды, передаваемой программе, с использованием процедуры WRD_TO_HEX.
3. Выводится tail командной строки, записанный в отдельной строке.
4. Выводится содержимое области среды посимвольно.
5. Выводится путь загружаемого модуля посимвольно.

Путь их определения написан в методических указаниях.

Вывод результата, полученного программой:



Выводы.

В ходе выполнения лабораторной работы была написана программа для вывода определенной информации из префикса сегмента программы и среды, исследован интерфейс управляющей программы и загрузочных модулей.

Получена программа, выводящая требуемую информацию из префикса PSP.

Контрольные вопросы по лабораторной работе №2

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на область основной оперативной памяти.

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

Он расположен за областью памяти, отведенной программе.

2) Можно ли в эту область памяти писать?

В эту область памяти можно писать, используя адресацию для сегментного регистра.

Среда передаваемая программе

1) Что такое среда?

Среда - область памяти, содержащая значения системных переменных, путей и другие данные операционной системы.

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

Среда создается при загрузке модуля в оперативную память.

3) Откуда берется информация, записываемая в среду?

Информация, записываемая в среду, берется из реестра операционной системы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab2.asm

```
TESTPC     SEGMENT
           ASSUME  CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
                org 100h
START:     JMP     BEGIN
; ДАННЫЕ
MEMORY_MSG db 13, 10, "Locked memory address:     h$" ; 17 symbols
ENVIROMENT_MSG db 13, 10, "Environment address:     h$" ; 23 symbols
TAIL_MSG db 13, 10, "Command line tail:        $" ; 21 symbols
EMPTY_MSG db 13, 10, "There are no sybmols$"
CONTENT_MSG db 13, 10, "Content:", 13, 10, "$"
ENTER db 13, 10, "$"
PATH db 13, 10, "Path:", 13, 10, "$" ; 8 symbols

;ПРОЦЕДУРЫ
;-------------------------------
PRINT PROC near
      push ax
      mov ah, 09h
      int 21h
      pop ax
      ret
PRINT ENDP
;-------------------------------
GIVE_INFO PROC near
      ; Memory
      mov ax, ds:[02h]
      mov di, offset MEMORY_MSG
      add di, 28
      call WRD_TO_HEX
      mov dx, offset MEMORY_MSG
      call PRINT

      ; Environment
      mov ax, ds:[2Ch]
      mov di, offset ENVIROMENT_MSG
      add di, 26
      call WRD_TO_HEX
      mov dx, offset ENVIROMENT_MSG
      call PRINT

      ; Tail
      xor cx, cx
      mov cl, ds:[80h]
      mov si, offset TAIL_MSG
      add si, 20
      test cl, cl
      jz empty
      xor di, di
      xor ax, ax
      readtail:
            mov al, ds:[81h+di]
            mov [si], al
            inc di
            inc si
            loop readtail
            mov dx, offset TAIL_MSG
            call PRINT
            jmp nextaction
```

```
      empty:
            mov dx, offset EMPTY_MSG
            call PRINT
      nextaction: nop

      ; Envrironment content
      mov dx, offset CONTENT_MSG
      call PRINT
      xor di, di
      mov bx, 2Ch
      mov ds, [bx]
      readstring:
            cmp byte ptr [di], 00h
            jz pressenter
            mov dl, [di]
            mov ah, 02h
            int 21h
            jmp findend
      pressenter:
            push ds
            mov cx, cs
            mov ds, cx
            mov dx, offset ENTER
            call PRINT
            pop ds
      findend:
            inc di
            cmp word ptr [di], 0001h
            jz readpath
            jmp readstring
      readpath:
            push ds
            mov ax, cs
            mov ds, ax
            mov dx, offset PATH
            call PRINT
            pop ds
            add di, 2
      pathloop:
            cmp byte ptr [di], 00h
            jz final
            mov dl, [di]
            mov ah, 02h
            int 21h
            inc di
            jmp pathloop
      final:
            ret
GIVE_INFO ENDP
;-------------------------------

;-------------------------------
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 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 - 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
;--------------------------------
; КОД
BEGIN:
                push ax
                push bx
                mov ah, 4Ah
                mov bx, 100h ; 256 paragraphs
                int 21h
                pop bx
                pop ax

                call GIVE_INFO
                mov ah, 10h
                int 16h
```

```
; Выход в DOS
           xor     AL,AL
           mov     AH,4Ch
           int     21H

TESTPC     ENDS
END        START
```