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

Студентка гр. 9382 _____ Пя 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 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_string: процедура вывода строки на экран

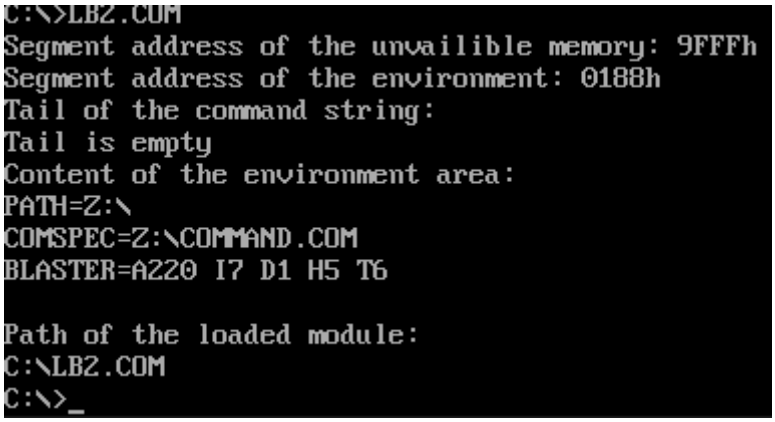
print_information: процедура вывода требуемой информации из содержимого PSP

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 хвост 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.

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: Lb2.ASM

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
 ORG 100H
START: JMP BEGIN
; ДАННЫЕ
seg_address_of_unavailable_memory db 'Segment address of the unvailible memory:
h',0DH,0AH,'$'
seg_address_of_environment db 'Segment address of the environment:     h',0DH,0AH,'$'
tail_of_command_string db 'Tail of the command string:',0DH,0AH,'$'
no_tail db 'Tail is empty',0DH,0AH,'$'
new_string db 0DH,0AH,'$'
tail db '                                                          ',0DH,0AH,'$'
content_of_environment_area db 'Content of the environment area: ',0DH,0AH,'$'
path_of_loaded_module db 'Path of the loaded module:',0DH,0AH,'$'
;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 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
print_string proc near
    mov ah, 09h
    int 21h
    ret
print_string endp

print_information proc near
seg_memory:
    mov ax, ds:[02h]
    mov di, offset seg_address_of_unavailable_memory
    add di, 45
    call WRD_TO_HEX
    mov dx, offset seg_address_of_unavailable_memory
    call print_string

seg_environment:
    mov ax, ds:[2Ch]
    mov di, offset seg_address_of_environment
    add di, 39
    call WRD_TO_HEX
    mov dx, offset seg_address_of_environment
    call print_string

tail_com:
    mov dx, offset tail_of_command_string
    call print_string
    sub cx, cx
    sub ax, ax
    sub di, di
    mov cl, ds:[80h]
    mov si, offset tail
    cmp cl, 0
    je if_zero
string_loop:;cx = cx - 1
    mov al, ds:[81h + di]
    inc di
    mov [si], al
    inc si
loop string_loop
```

```
    mov dx, offset tail
    call print_string
    jmp content_of_environment

if_zero:
    mov dx, offset no_tail
    call print_string
    jmp content_of_environment

content_of_environment:
    mov dx, offset content_of_environment_area
    call print_string
    sub di, di
    mov bx, 2Ch
    mov ds, [bx]
loop_env_string:
    cmp byte ptr [di], 00h ;проверка на конец строки
    je next_string
    mov dl, [di];вывод
    mov ah, 02h
    int 21h
    jmp check_path
next_string:
    push ds
       mov cx, cs
       mov ds, cx
       mov dx, offset new_string ;перенос на новую строку
       call print_string
       pop ds
check_path:
    inc di
    cmp word ptr [di], 0001h ;начался путь
    je print_path
    jmp loop_env_string
print_path:
    push ds
    mov ax, cs
    mov ds, ax
    mov dx, offset path_of_loaded_module
    call print_string
    pop ds
    add di, 2 ;на начало пути
loop_path:
    cmp byte ptr [di], 00h;проверка на конец пути
    je to_end
    mov dl, [di]
    mov ah, 02h
    int 21h
    inc di
    jmp loop_path
to_end:
    ret
print_information endp

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