МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Операционные системы»
Тема: Исследование интерфейсов программных модулей

Студент гр.8382 _____ Синельников М.Р

Преподаватель _____ Ефремов М.А

Санкт-Петербург
2020

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

Ход работы.

1) Скриншот запуска программы и результатов:

рисунок 1 — результат работы

Контрольные вопросы.

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

На сегментный адрес последнего параграфа памяти, который DOS использует для запуска программ.

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

Сразу за областью памяти, отведённой программе.

3) Можно ли в эту область памяти писать?

Можно.

Среда передаваемая программе

1) Что такое среда?

Среда — это массив строк ASCIIZ, вида параметр = значение. Среда начинается с границы параграфа, после последней строки следует нулевой байт.

Среда содержит как минимум, параметр COMSPEC. Она также содержит значения, установленные командами PROMPT, PATH, SET.

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

В процессе начальной загрузки DOS создает окружение, в котором будут работать активизируемые программы, и, прежде всего, COMMAND.COM.

3) Откуда берётся информация, записываемая в среду?

Во время запуска программы, система выделяет две области памяти - для программы и среды окружения (обычно эта память располагается перед памятью программы). Окружение предыдущей программы копируется в новую область памяти. Программа может менять переменные окружения (команда SET).

Выводы.

Был исследован интерфейс управлющей программы и загрузочных модулей, а также префикс сегмента программы (PSP) и среды, передаваемой программе.

Приложение А. Исходный код программы OS_2.asm

```
TESTPC          SEGMENT
                ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
                ORG 100H ; резервирование места для PSP
START: JMP begin

ADNAM db 'Address not available memory:     ',0DH,0AH,'$'
AENV db 'Environment adress:      ',0DH,0AH,'$'
TAIL db 'Tail: ','$'
NEW_LINE db 0DH,0AH,'$'
ENV_CONTENT db 'Environment content: ',0DH,0AH,'$'
PATH db 'PATH to file:  ',0DH,0AH,'$'

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

DEFINE_ADNAM PROC near
        push ax
        mov ax,es:[2]
        lea di,ADNAM
        add di,33
        call WRD_TO_HEX
        pop ax
        ret

DEFINE_ADNAM ENDP

DEFINE_AENV PROC near
        push ax
        mov ax,es:[2ch]
        lea di,AENV
        add di, 24
        call WRD_TO_HEX
        pop ax
        ret

DEFINE_AENV ENDP

DEFINE_TAIL PROC near
        push ax
        push bx
        push dx
        mov bl,es:[80h]
        lea di,es:[81h]
        mov cl,0h
        cmp cl,bl
        je continue
        mov ah,02h
        do1:
                mov dl,[di]
                int 21h
                inc di
                add cl,1
                cmp cl,bl
                jle do1
```

```
        continue:
        pop dx
        pop bx
        pop ax
        ret

DEFINE_TAIL ENDP

DEFINE_ENV_CONTENT PROC near
        push ax
        push bx
        push cx
        push dx
        push ds
        mov es,es:[2ch]
        mov di,0
        cmp byte ptr es:[di],00h
        je go
        mov ah,02h
        do2:
                mov dl,es:[di]
                int 21h
                inc di
                cmp byte ptr es:[di],0
                jne do2
                cmp byte ptr es:[di + 1],0
                jne do2

        go:
        push dx
        lea dx,NEW_LINE
        call PRINT_STR
        pop dx
        push dx
        lea dx,PATH
        call PRINT_STR
        pop dx
        add di,4
        cmp byte ptr es:[di],00h
        je finish
        do3:
                mov dl,es:[di]
                int 21h
                inc di
                cmp byte ptr es:[di],00h
                jne do3

        finish:
        pop ds
        pop dx
        pop cx
        pop bx
        pop ax
        ret

DEFINE_ENV_CONTENT ENDP

PRINT_STR PROC near
        push ax
        mov ah,09h
        int 21h
```

```
        pop ax
        ret

PRINT_STR ENDP

begin:
        call DEFINE_ADNAM
        lea dx,ADNAM
        call PRINT_STR
        call DEFINE_AENV
        lea dx,AENV
        call PRINT_STR
        lea dx,TAIL
        call PRINT_STR
        call DEFINE_TAIL
        lea dx, NEW_LINE
        call PRINT_STR
        lea dx,ENV_CONTENT
        call PRINT_STR
        call DEFINE_ENV_CONTENT
        ret

TESTPC          ENDS
        END START
```