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. 9383 _____ Нистратов Д.Г.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Постановка задачи.

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.

Функции и структуры данных

Таблица 1 – Функции и структуры данных

Название функций	Описание функций
TETR_TO_HEX	Перевод из четверичной с/с в шестнадцатеричную с/с
BYTE_TO_HEX	Перевод из двоичной с/с в шестнадцатеричную с/с
WRD_TO_HEX	Перевод 2 байтов в шестнадцатеричную с/с
BYTE_TO_DEC	Перевод из двоичной с/с в десятичную с/с
WRITE	Вывод строки на экран
PSP_MEMORY	Вывод сегментного адреса недоступной памяти, взятый из PSP, в шестнадцатеричном виде
PSP_ENVIROMENT	Вывод сегментного адреса среды, передаваемой программе, в шестнадцатеричном виде
PSP_TAIL	Вывод хвоста командной строки в символьном виде

PSP_CONTENT	Вывод содержимого области среды в символьном виде и путь загружаемого модуля

Последовательность действий

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 включите скриншот с запуском программы и результатами.

Выполнение работы.

При выполнении лабораторной работы был написан .COM модуль, см. Изображение 1 и Изображение 2, который выбирает и распечатывает информацию из префикса сегмента программы (PSP) и среды, передаваемой программе.

Изображение 1 - Результат работы программы без аргументов

Изображение 2- Результат работы программы с аргументами

Сегментный адрес недоступной памяти:

1) На какую область памяти указывает адрес недоступной памяти?

   Ответ: на адрес сегмента памяти, после программы.

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

   Ответ: расположен в стороне бОльших адресов.

3) Можно ли в эту область памяти писать?

   Ответ: можно, так как DOS не имеет механизма защиты от перезаписи памяти программ.

Среда, передаваемая программе

1) Что такое среда?

   Ответ: среда – это участок памяти, содержащий в себе переменные среды, путей и другие данные операционной системы.

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

Ответ: среда создается при загрузке ОС. При запуске программы среда копируется в адресное пространство.

3) Откуда берется информация, записываемая в среду?

   Ответ: из системного файла AUTOEXEC.BAT, расположенного в корневом каталоге устройства.

Заключение.

В ходе лабораторный работы был исследован интерфейс управляющей программы и загрузочного модуля, а также префикс сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab2.asm

```
; Исследование интерфейсов программных модулей
; 18.02.2021
; Nistratov Dmitry

; Шаблон текста программы на ассемблере для модуля типа .COM
TESTPC  SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG 100H
START:  JMP BEGIN
; Данные
MEM_ADRESS db 'Locked memory address:     ', 0DH, 0AH, '$' ; 22 + s
pace + hex
ENV_ADRESS db 'Environment address:      ', 0DH, 0AH, '$' ; 20 + sp
ace + hex
TAIL db 'Command line tail:$'
EMPTY_TAIL db 'In Command tail no sybmols', 0DH, 0AH, '$'
CONTENT db 'Content:', 0DH, 0AH, '$'
END_STRING db 0DH, 0AH, '$'
PATH db 'Path:  ', 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 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 цифра
            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
WRITE PROC NEAR
        push ax
        mov ah, 09h
        int 21h
        pop ax
        ret
WRITE ENDP

WRITEBYTE  PROC  NEAR
        push ax
        mov ah, 02h
        int 21h
        pop ax
        ret
WRITEBYTE  ENDP

PSP_MEMORY PROC near
; Сегментный адрес недоступной памяти
        push ax
        push dx
        push di

        mov ax,ds:[02h]
```

```
        mov di, offset MEM_ADRESS
        add di, 26
        call WRD_TO_HEX
        mov dx, offset MEM_ADRESS
        call WRITE

        pop ax
        pop dx
        pop di
        ret
PSP_MEMORY ENDP

PSP_ENVIROMENT  PROC near
; Сегментный адрес среды
        push ax
        push dx
        push di

        mov ax,ds:[2Ch]
        mov di, offset ENV_ADRESS
        add di, 24
        call WRD_TO_HEX
        mov dx, offset ENV_ADRESS
        call WRITE

        pop ax
        pop dx
        pop di
        ret
PSP_ENVIROMENT ENDP

PSP_TAIL PROC near
; хвост командной строки
        push ax
        push cx
        push dx
        push di

        xor cx, cx
        xor di, di

        mov cl, ds:[80h]
        cmp cl, 0h
        je empty

        mov dx, offset TAIL
        CALL WRITE
read:
        mov dl, ds:[81h+di]
        call WRITEBYTE
        inc di

        loop read

        mov dx, 0dh
        call WRITEBYTE

        mov dl, 0ah
```

```
        call WRITEBYTE
        jmp end_pop
empty:
        mov dx, offset EMPTY_TAIL
        call WRITE
end_pop:

        pop ax
        pop cx
        pop dx
        pop di
        ret
PSP_TAIL ENDP

PSP_CONTENT PROC near
;Содержимое области среды и путь загрузочного модуля
        push ax
        push cx
        push dx
        push di

        mov dx, offset CONTENT
        call WRITE
        xor di,di
        mov ds, ds:[2ch]
read_string:
        cmp byte ptr [di], 00h
        jz end_str
        mov dl, [di]
        mov ah, 02h
        int 21h
        jmp find_end
end_str:
        cmp byte ptr [di+1],00h
        jz find_end
        push ds
        mov cx, cs
        mov ds, cx
        mov dx, offset END_STRING
        call WRITE
        pop ds
find_end:
        inc di
        cmp word ptr [di], 0001h
        jz read_path
        jmp read_string
read_path:
        push ds
        mov ax, cs
        mov ds, ax
        mov dx, offset PATH
        call WRITE
        pop ds
        add di, 2
loop_path:
        cmp byte ptr [di], 00h
        jz end_pop_content
        mov dl, [di]
```

```
        mov ah, 02h
        int 21h
        inc di
        jmp loop_path
end_pop_content:
        pop ax
        pop cx
        pop dx
        pop di

        ret
PSP_CONTENT ENDP

BEGIN:
   call PSP_MEMORY
   call PSP_ENVIROMENT
   call PSP_TAIL
   call PSP_CONTENT

; Выход в DOS
        xor AL,AL
        mov AH,4Ch
        int 21H
TESTPC  ENDS
        END START ; Конец модуля, START - точка входа
```