

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. 9381

Преподаватель

Фоминенко А.Н.

Ефремов М. А.

Санкт-Петербург

2021

Цель работы.

Изучение интерфейса управляющей программы и загрузочных модулей. Исследование префикса программного сегмента PSP и среды, передаваемой программе.

Последовательность действий программы.

Программа находит информацию из PSP и выводит её на экран в следующем виде:

- Сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде.
- Сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.
- Хвост командной строки в символьном виде.
- Содержимое области среды в символьном виде.
- Путь загружаемого модуля.

Ход работы.

- 1) Был написан текст исходного .COM модуля lab2.asm
- 2) Далее с помощью транслятора `masm.exe` и компоновщика `link.exe` был скомпилирован плохой .EXE модуль. При помощи `exe2bin.exe` по плохому .EXE модулю был построен хороший .COM модуль.
- 3) Далее загрузочный модуль был протестирован и отлажен.

Результаты работы программы:

```
C:\>lab2.com
Segment address of the unavailable memory:9FFF
Segment address of the environment:0188
The tail of the command line: Empty
Environment area content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Path to loaded module: C:\LAB2.COM

C:\>lab2.com some_argument
Segment address of the unavailable memory:9FFF
Segment address of the environment:0188
The tail of the command line: some_argument
Environment area content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Path to loaded module: C:\LAB2.COM

C:\>
```

Функции программ.

Названия функций	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа.
BYTE_TO_HEX	Перевод байта 16-ной с.с. в символьный код.
WRD_TO_HEX	Перевод слова 16-ной с.с. в символьный код.
PRINT	Вывод строки.

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти.

- 1) На какую область памяти указывает адрес недоступной памяти?

Он указывает на первый байт после памяти, отведённой программе.

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

Адрес хранится сразу после памяти, выделенной программе. Расположен в сторону увеличения адресов.

3) Можно ли в эту область памяти писать?

Можно, так как в DOS нет защиты памяти.

Среда передаваемая программе.

1) Что такое среда?

Среда - это массив символов, состоящий из последовательности символьных строк вида:

<имя> = <параметр>, 00h

Где 00h - нулевой байт, обозначающий конец строки. Конец среды - также байт нулей. Среда хранит такую служебную информацию, как данные об используемом командном процессоре, путь к модулю COMMAND.COM и др.

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

Среда создаётся перед запуском приложения. копирование всех переменных среды осуществляется для каждой запускаемой программы.

3) Откуда берётся информация, записываемая в среду?

С помощью командного интерпретатора COMMAND.COM выполняется запуск файла AUTOEXEC.BAT, из которого и берётся вся необходимая информация.

Вывод.

Был изучен интерфейс управляющей программы и загрузочных модулей.
Был изучен PSP и среда, передаваемая программе.

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
Unavailable_Mem_Address      db 'Segment address of the unavailable memory:
',0DH,0AH,'$'
Environment_Seg_Address      db 'Segment address of the environment:
',0DH,0AH,'$'
Command_Line_Tail            db 'The tail of the command line: ','$'
Tail_Empty                   db 'Empty',0DH,0AH,'$'
Environment_Area_Content     db 'Environment area content: ',0DH,0AH,'$'
Module_Loaded_Path           db 'Path to loaded module: ','$'
Tail                          db 128 DUP('$')
Content                      db 128 DUP('$')
Path                          db 128 DUP('$')

; 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 АН 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
```

```
;-----
```

```
PRINT PROC NEAR ; вывод строки на экран
```

```
    push ax
    mov ah, 9h
    int 21h
    pop ax
    ret
```

```
PRINT ENDP
```

```
;-----
```

```
; КОД
```

```
BEGIN:
```

```
;Сегментный адрес недоступной памяти
```

```
    mov ax, ds:[0002h]
    mov dx, offset Unavailable_Mem_Address
    mov di, dx
    add di, 45
    call WRD_TO_HEX
    call PRINT
```

;Сегментный адрес среды

```
mov ax, ds:[002Ch]
mov dx, offset Environment_Seg_Address
mov di, dx
add di, 38
call WRD_TO_HEX
call PRINT
```

;Хвост командной строки

```
mov cl, ds:[0080h]
mov dx, offset Command_Line_Tail
call PRINT
cmp cl, 0
je Empty_Tail
mov si, 0
```

Tail_Loop:

```
mov al, ds:[81h + si]
mov Tail[si], al
inc si
loop Tail_Loop
```

Print_Tail:

```
mov al, 0Ah
mov Tail[si], al
mov dx, offset Tail
call PRINT
jmp Environment
```

Empty_Tail:

```
mov dx, offset Tail_Empty ;сообщение об отсутствии аргументов
call PRINT
```

;Содержимое области среды

Environment:

```
mov dx, offset Environment_Area_Content
call PRINT
mov ax, ds:[2Ch]
mov ds, ax
mov di, offset Content
mov si, 0
```

Read_content:

```
lodsb
cmp al, 0
je EOL_OR_EXIT
stosb
```

```

        jmp Read_content
EOL_OR_EXIT:
        mov al, 0Ah
        stosb
        lodsb
        cmp al, 0
        je End_Environment
        stosb
        jmp Read_content
End_Environment:
        mov al, 0Dh
        stosb
        mov al, 36
        stosb
        push ds
        mov ax, es
        mov ds, ax
        mov dx, offset Content
        call PRINT
;Путь загружаемого модуля
        mov dx, offset Module_Loaded_Path
        call PRINT
        add si, 2
        mov ds, ds:[2Ch]
        mov di, offset Path
Read_Path:
        lodsb
        cmp al, 0
        je End_Path
        stosb
        jmp Read_Path
End_Path:
        mov al, 0Ah
        stosb
        mov al, 0Dh
        stosb
        mov al, 36
        stosb
        mov ax, es
        mov ds, ax
        mov dx, offset Path

```



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