

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. 8383

Мололкин К.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

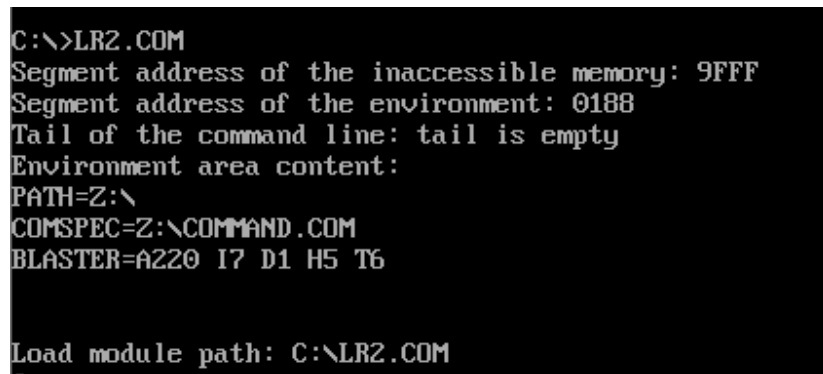
2020

Цель работы

Исследование интерфейса управляющей программы и загрузочных модулей. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Ход работы.

Во время выполнения работы был написан и отлажен код .COM модуля, код модуля представлен в приложении А. Данная программа читает и распечатывает сегментный адрес недоступной памяти, сегментный адрес среды, хвост командной строки, содержимое области среды, а также путь загружаемого модуля. Результат выполнения программы показан на рис. 1.



```
C:\>LR2.COM
Segment address of the inaccessible memory: 9FFF
Segment address of the environment: 0188
Tail of the command line: tail is empty
Environment area content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Load module path: C:\LR2.COM
```

Рисунок 1 – Результат выполнения программы

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на область памяти, за которым программа не должна модифицировать содержимое.

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

Адрес расположен после памяти, отведенной программе.

3) Можно ли в эту область памяти писать?

В эту область памяти можно писать, так DOS не препятствует этому.

Среда, передаваемая программе

1) Что такое среда?

Область среды содержит последовательность строк вида:

имя=параметр

Каждая строка завершается байтом нулей. Под строки выделяется 32 Кбайта памяти.

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

Среда создается перед запуском приложения. Она копируется из COMMAND.COM, которая создается при запуске DOS.

3) Откуда берется информация, записываемая в среду

Из родительской среды.

Выводы.

В ходе выполнения лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей, а также префикс сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
LR2 SEGMENT
    ASSUME CS:LR2, DS:LR2, ES:NOTHING, SS:NOTHING
    ORG 100H
    START: JMP BEGIN

    LOCKED_MEMORY_ADDRESS db "Segment address of the inaccessible
memory:      $"
    ENVIRONMENT_ADDRESS  db 13, 10, "Segment address of the
environment:      $"
    COMAND_LINE_TAIL db 13, 10, "Tail of the command line: $"
    EMPTY_TAIL db "tail is empty$"
    ENIRONMENT_CONTENT db 13, 10, "Environment area content:",
13, 10, '$'
    LOAD_PATH db 13, 10, "Load module path: $"
    ENTER_SYMB db 13, 10, '$'

WRD_TO_HEX PROC near
    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

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
    push CX
    mov AH,AL
    call TETR_TO_HEX
    xchg AL,AH
```

```

        mov CL,4
        shr AL,CL
        call TETR_TO_HEX
        pop CX
        ret
BYTE_TO_HEX ENDP

BYTE_TO_DEC PROC near
        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

PRINT_STRING PROC near
        mov ah, 09h
        int 21h
        ret
PRINT_STRING ENDP

PRINT_SYMBOL PROC near
        mov dl, al
        mov ah, 02h
        int 21h
        ret
PRINT_SYMBOL ENDP

LM_ADRESS PROC near
        mov ax, ds:[02h]
        mov cx, offset LOCKED_MEMORY_ADRESS
        mov di, cx
        add di, 47
        call WRD_TO_HEX
        mov dx, cx
        call PRINT_STRING
        ret

```

```

LM_ADDRESS ENDP

E_ADDRESS PROC near
    mov ax, ds:[2Ch]
    mov cx, offset ENVIRONMENT_ADDRESS
    mov di, cx
    add di, 41
    call WRD_TO_HEX
    mov dx, cx
    call PRINT_STRING
    ret
E_ADDRESS ENDP

TAIL PROC near
    mov dx, offset COMAND_LINE_TAIL
    call PRINT_STRING
    xor cx, cx
    mov cl, ds:[80h]
    cmp cl, 0
    jne line_not_empty
    mov dx, offset EMPTY_TAIL
    call PRINT_STRING
    jmp p_end
line_not_empty:
    xor si, si
    xor dx, dx
cycle:
    mov al, DS:[81h + si]
    call PRINT_SYMBOL
    inc si
    loop cycle
p_end:
    ret
Tail ENDP

E_CONTENT PROC near
    mov dx, offset ENIRONMENT_CONTENT
    call PRINT_STRING
    xor si, si
    mov bx, 2Ch
    mov es, [bx]

print_content_cycle:
    cmp BYTE PTR es:[si], 0h
    je print_enter
    mov al, es:[si]
    call PRINT_SYMBOL
    jmp end_reading
print_enter:
    mov dx, offset ENTER_SYMB
    call PRINT_STRING
end_reading:
    inc si

```

```

        cmp WORD PTR es:[si], 0001h
jne print_content_cycle

        mov DX, offset LOAD_PATH
        call PRINT_STRING
        add SI, 2
print_path:
        cmp BYTE PTR es:[si], 00h
        je p_endd
        mov al, es:[si]
        call PRINT_SYMBOL
        inc si
        jmp print_path
p_endd:
        ret
E_CONTENT ENDP
BEGIN:
        call LM_ADRESS
        call E_ADRESS
        call TAIL
        call E_CONTENT
        xor AL, AL
        mov AH, 4Ch
        int 21h
LR2 ENDS
END START

```