

**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.**

Факультет: КТИ

Дата выполнения работы: 25.02.2021

Студент гр. 9381

Семенов А. Н.

Преподаватель

Ефремов М. А.

Санкт-Петербург
2021

Цель работы.

Исследование интерфейсов управляющей программы и загрузочных модулей. Этот интерфейс состоит в передачи запускаемой программе управляющего блока, содержащего адреса и системные данные. Так загрузчик строит префикс сегмента программы и помещает его адрес в системный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Задание.

Написать и отладить программный модуль **.COM**, который выбирает и распечатывает следующую информацию:

- 1) Сегментный адрес недопустимой 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) Пусть загружаемого модуля.

Сохранить результаты, полученные программой, и включить их в отчет.

Функции и структуры данных.

TETR_TO_HEX – переводит значение 4 младших битов регистра al в цифру 16-ричной системы счисления в виде символа, которая кладется в регистр al.

BYTE_TO_HEX – переводит значение байта, содержащегося в регистре al в двухсимвольное число в шестнадцатеричной системе счисления, которые кладется в регистр ah: код первого символа в al, второго в ah.

WRD_TO_HEX – переводит значение регистра AX в шестнадцатеричное число в виде 4 символов, которые кладутся в память по адресу di.

PRINT_SYMBOL – кладет символ из памяти ES:[DI] в данные программы и печатает его.

PRINT_ENTER – печатает символы перевода строки.

PRINT_TAB – печатает символ табуляции.

Последовательность действий, выполняемых программой.

1. В регистр AX записывается сегментный адрес недопустимой памяти, взятый из PSP, переводится в символьный вид с помощью вызова процедуры WRD_TO_HEX и кладется в память по метке Adress_1.

2. В регистр AX записывается сегментный адрес среды, передаваемой программе, взятый из PSP, переводится в символьный вид с помощью вызова процедуры WRD_TO_HEX и кладется в память по метке Adress_2.

3. Производится печать сегментных адресов недопустимой памяти и среды на экран соответствующим сообщением.

4. В CX записывается число символов в хвосте командной строки. Запускается цикл loop, печатающий посимвольно с помощью вызовов в каждой итерации процедуры PRINT_SYMBOL хвост командной строки.

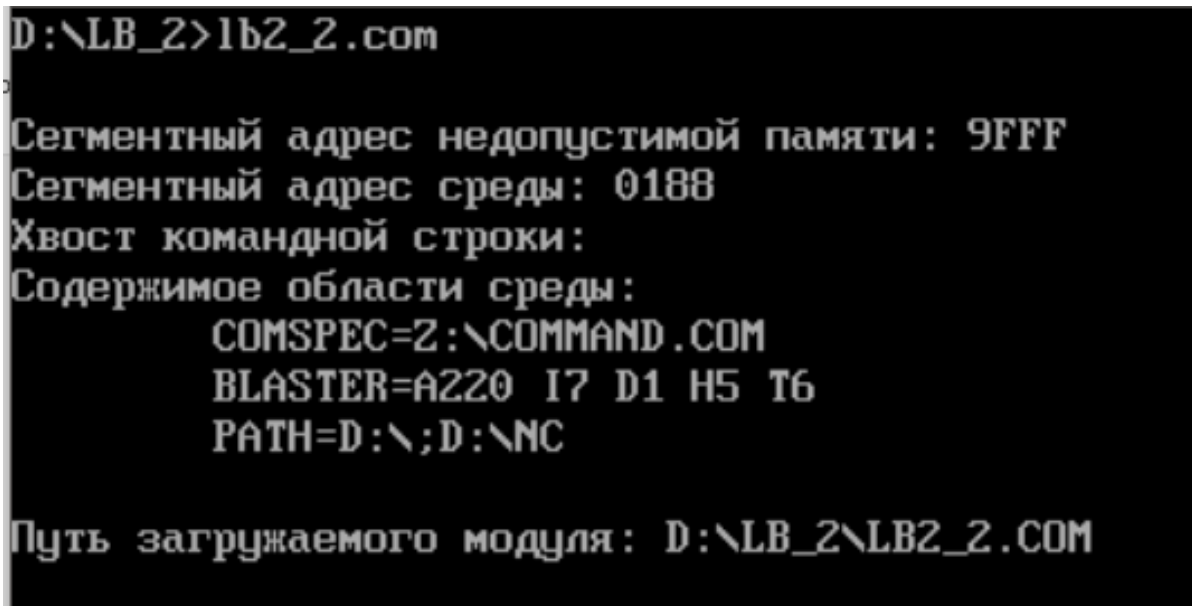
5. Производится печать информации о выводе содержимого среды. Далее печать содержимого среды в цикле посимвольно с помощью вызовов в каждой итерации процедуры PRINT_SYMBOL, при этом разделяя строки информации символами перевода строки с помощью вызовов между строками процедуры PRINT_ENTER.

6. Производится печать информации о выводе пути загружаемого файла, после чего собственно печать пути в цикле посимвольно.

Ход работы

1. Написание исходного текста исходного .COM модуля на языке ассемблера, файл: LB2_2.ASM (см. в приложении).

2. Трансляция исходного кода командой: *masm LB2_2.ASM*, отладка и получение объектного модуля *LB2_2.OBJ*.
3. Сборка объектного модуля командой: *link LB2_2.OBJ*, и получение загрузочного модуля *LB2_2.EXE*.
4. Получение **.COM** модуля: *LB2_2.COM*, с помощью команды: *exe2bin LB2_2.EXE*.
5. Запуск программы в терминале ДОС (рис. 1, рис. 2):

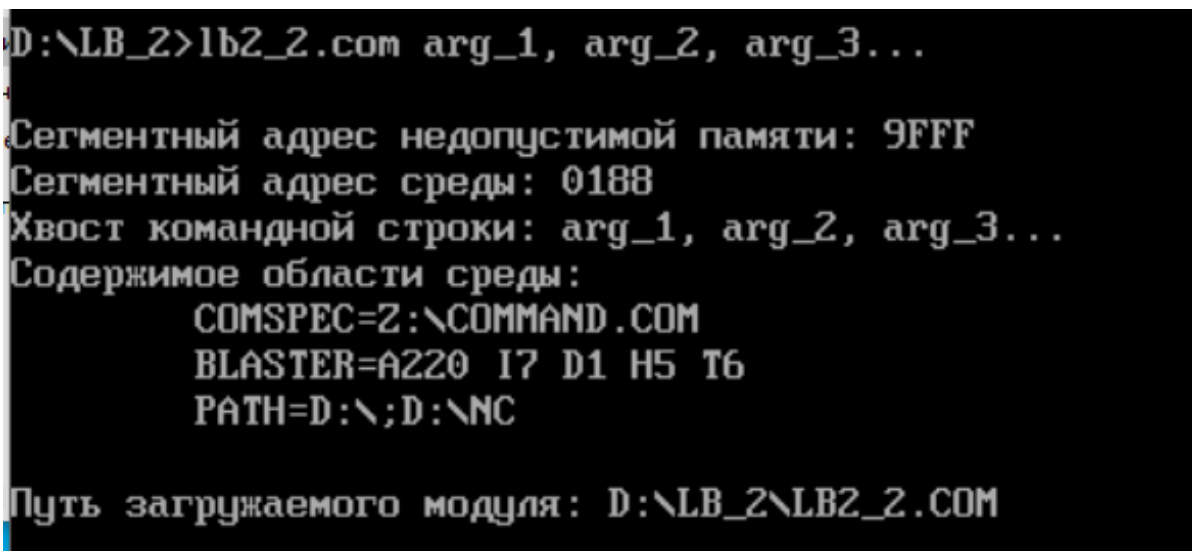


```
D:\LB_2>lb2_2.com

Сегментный адрес недопустимой памяти: 9FFF
Сегментный адрес среды: 0188
Хвост командной строки:
Содержимое области среды:
      COMSPEC=Z:\COMMAND.COM
      BLASTER=A220 I7 D1 H5 T6
      PATH=D:\;D:\NC

Путь загружаемого модуля: D:\LB_2\LB2_2.COM
```

Рис. 1. Запуск программы LB2_2.COM в терминале ДОС без аргументов



```
D:\LB_2>lb2_2.com arg_1, arg_2, arg_3...

Сегментный адрес недопустимой памяти: 9FFF
Сегментный адрес среды: 0188
Хвост командной строки: arg_1, arg_2, arg_3...
Содержимое области среды:
      COMSPEC=Z:\COMMAND.COM
      BLASTER=A220 I7 D1 H5 T6
      PATH=D:\;D:\NC

Путь загружаемого модуля: D:\LB_2\LB2_2.COM
```

Рис. 2. Запуск программы LB2_2.COM в терминале ДОС с аргументами

Ответы на вопросы.

Сегментный адрес недопустимой памяти:

1) На какую область памяти указывает адрес недопустимой памяти? *На область памяти, следующую за областью памяти, отведенной программе, т. е. на ее первый байт.*

2) Где расположен этот адрес по отношению области памяти, отведенной программе? *Этот двухбайтовый адрес расположен в сегменте PSP по смещению – 2 байта от его начала, ровно как и от начала памяти, отведенной программе.*

3) Можно ли в эту область памяти писать? *Можно, потому что в DOS нет контроля доступа к памяти, однако программа не должна модифицировать содержимое этой области памяти.*

Среда, передаваемая программе:

1) Что такое среда? *Окружение ОС, некая совокупность процессов, в которой запускается та или иная программа. Область среды – это сегментная область памяти, хранящая переменные среды – именованные переменные, содержащие настройки и параметры текущей операционной системы (в данном случае DOS), а именно символьные строки вида: «имя»=«параметр», содержащие служебную информацию, такую как имя COMSPEC, которое определяет используемый командный процессор и путь к COMMAND.COM, информацию, задаваемую командами PATH, PROMPT, SET. Запускаемые в ОС программы (модули), при необходимости, могут получить значения переменных среды ОС и использовать их при работе программы.*

2) Когда создается среда? *Перед запуском приложения или в другое время? Среда создается перед запуском приложения, который в DOS осуществляется командным интерпретатором (COMMAND.COM), имеющим свою среду. Запуск интерпретатора производится при загрузке DOS, а его команды получают копию блока его среды. То есть среда создается при загрузке операционной*

системы (среда COMMAND.COM), и копируется каждой из запускаемых программ.

3) Откуда берется информация, записываемая в среду? Информация, записываемая в среду, берется из системного файла AUTOEXEC.BAT, который выполняется командным интерпретатором и производит установку переменных среды.

Вывод.

В ходе лабораторной работы было проведено исследование интерфейсов управляющей программы и загрузочных модулей, а также префикса сегмента программы (PSP) и среды, передаваемой программе. Была разработана, протранслирована и протестирована собственная программа на языке Ассемблера, печатающая информацию об адресах недопустимой памяти, среды, а также содержимое среды, обращаясь к PSP-префиксу.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл LB2_2.ASM:

```
LB_2 SEGMENT
    ASSUME CS:LB_2, DS:LB_2, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN
; Данные:
Message db 0DH, 0AH, 'Сегментный адрес недопустимой памяти: '
Address_1 dw 0, 0
            db 0DH, 0AH

            db 'Сегментный адрес среды: '
Address_2 dw 0, 0
            db 0DH, 0AH

            db 'Хвост командной строки:$'
Symbol db 0, '$'
Content db 'Содержимое области среды:', 0DH, 0AH, '$'
Puth db 'Путь загружаемого модуля: $'
Enter_ db 0DH, 0AH, '$'

; Код (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
    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

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

PRINT_SYMBOL PROC near
        mov BL, ES:[DI]
        mov Symbol, BL
        int 21h
        inc DI
        ret
PRINT_SYMBOL ENDP

PRINT_ENTER PROC near
        push DX
        mov DX, offset Enter_
        int 21h
        pop DX
        ret
PRINT_ENTER ENDP

```



```

PRINT_TAB PROC near
    mov Symbol, 09h
    int 21h
    ret
PRINT_TAB ENDP

```

; Код (программа):

```

BEGIN:

    mov AX, ES:0002h
    mov DI, offset Adress_1
    add DI, 3
    call WRD_TO_HEX

    mov AX, ES:002Ch
    mov DI, offset Adress_2
    add DI, 3
    call WRD_TO_HEX

    mov DX, offset Message
    mov AH, 09h
    int 21h

    mov CH, 0
    mov CL, ES:0080h
    mov DX, offset Symbol
    mov AH, 09h
    mov DI, 81h
    cmp CX, 0
    je End_loop

Write_tail:
    call PRINT_SYMBOL
    loop Write_tail

End_loop:
    call PRINT_ENTER

    push DX
    mov DX, offset Content
    int 21h
    pop DX

    mov ES, DS:002Ch

```

```

        mov DI, 0
        mov CL, 0
        call PRINT_TAB
Writing:
        cmp ES:[DI], CL
        jne Further
        call PRINT_ENTER
        call PRINT_TAB
        inc DI
Further:
        cmp ES:[DI], CL
        je Finish
        call PRINT_SYMBOL
        jmp Writing
Finish:
        call PRINT_ENTER
        push DX
        mov DX, offset Puth
        int 21h
        POP DX

        add DI, 3
Writing_puth:
        cmp ES:[DI], CL
        je Main_Finish
        call PRINT_SYMBOL
        jmp Writing_puth
Main_Finish:
        call PRINT_ENTER

        xor AL,AL
        mov AH,4Ch
        int 21H
LB_2 ENDS
END START

```