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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Организация ЭВМ и систем»

Тема: «Исследование интерфейсов программных модулей»

Студент гр. 8381 _____ Переверзев Д.Е.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2019

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.

Основные теоретические положения.

При начальной загрузке программы формируется PSP, который размещается в начале первого сегмента программы. PSP занимает 256 байт и располагается с адреса, кратного границе сегмента. При загрузке модулей типа .COM все сегментные регистры указывают на адрес PSP. При загрузке модуля типа .EXE сегментные регистры DS и ES указывают на PSP. Именно по этой причине значения этих регистров в модуле .EXE следует переопределять.

Таблица 1 – Формат PSP

Смещение	Длина поля(байт)	Содержимое поля
0	2	int 20h
2	2	Сегментный адрес первого байта недоступной памяти. Программа не должна модифицировать содержимое памяти за этим адресом.
4	6	Зарезервировано
0Ah (10)	4	Вектор прерывания 22h (IP,CS)
0Eh (14)	4	Вектор прерывания 23h (IP,CS)
12h (18)	4	Вектор прерывания 24h (IP,CS)
2Ch (44)	2	Сегментный адрес среды, передаваемой программе.
5Ch		Область форматируется как стандартный неоткрытый блок управления файлом (FCB)
6Ch		Область форматируется как стандартный неоткрытый блок управления файлом (FCB). Перекрывается, если FCB с адреса 5Ch открыт.
80h	1	Число символов в хвосте командной строки.
81h		Хвост командной строки - последовательность символов после имени вызываемого модуля.

Область среды содержит последовательность символьных строк вида: имя=параметр Каждая строка завершается байтом нулей. В первой строке указывается имя COMSPEC, которая определяет используемый командный

процессор и путь к COMMAND.COM. Следующие строки содержат информацию, задаваемую командами PATH, PROMPT, SET. Среда заканчивается также байтом нулей. Таким образом, два нулевых байта являются признаком конца переменных среды. Затем идут два байта, содержащих 00h, 01h, после которых располагается маршрут загруженной программы. Маршрут также заканчивается байтом 00h.

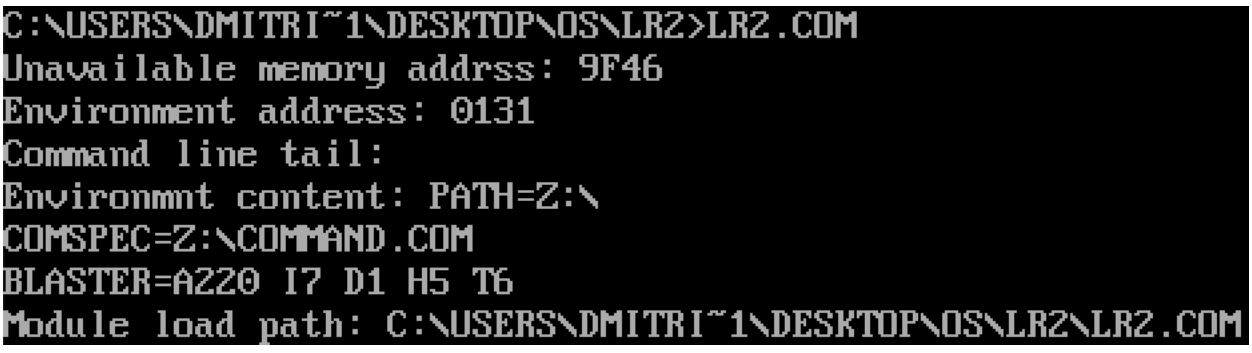
Выполнение работы.

Написан текст исходного .COM модуля, который выбирает и распечатывает следующую информацию:

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

Полученный исходный модуль был отлажен. Результаты выполнения программы представлены на рис. 1.

Рисунок 1 – Вывод программы



Контрольные вопросы

Сегментный адрес недоступной памяти:

1. На какую область памяти указывает адрес недоступной памяти?

Область недоступной памяти начинается с 9FFF. Ее адрес указывает на служебную часть памяти, которую DOS не может выделить под программу.

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

Адрес недоступной памяти превышает адрес начала области памяти, отведённой программе.

3. Можно ли в эту область памяти писать?

Можно, потому что память DOS не имеет защиты.

Среда, передаваемая программе:

1. Что такое среда?

Среда – это последовательность символьных строк вида: имя=параметр (они называются переменные среды), которые содержат данные о некоторых директориях операционной системы и конфигурации компьютера, которые передаются программе, когда она запускается.

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

В процессе начальной загрузки DOS создает начальную среду, в которой будут работать активизируемые программы, и прежде всего командный процессор command.com .

1. Откуда берется информация, записываемая в среду?

Информация, берётся из системного файла autoexec.bat. Эта информация может быть изменена пользователем во время работы системы.

Вывод

В результате выполнения данной лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей. Была написана программа, которая выводит на экран сегментный адрес недоступной памяти, взятый из PSP, сегментный адрес среды, передаваемой программе, хвост командной строки и путь загружаемого модуля.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR2.ASM

```
VERSER SEGMENT
        ASSUME CS:VERSER, DS:VERSER, ES:NOTHING, SS:NOTHING
        org 100H
START:  jmp BEGIN
UNAVAILABLE_ADDRESS         db  'Unavailable memory addrss: $'
ENVIRONMENT_ADDRESS         db  'Environment address: $'
COMMAND_LINE_TAIL           db  'Command line tail: $'
ENVIRONMENT_CONTENT         db  'Environmnt content: $'
LOAD_PATH                   db  'Module load path: $'
ENDL_LINE                   db  0dh, 0ah, '$'


BYTE_IN_HEX PROC near
        push    cx
        push    si
        mov     al, ah
        call    TETR_TO_HEX
        xchg    al, ah
        mov     cl, 4
        shr     al, cl
        call    TETR_TO_HEX
        lea     si, db  '  $'
        mov     [si], al
        mov     [si+1], ah
        mov     dx, offset db   '  $'
        call    PRINT_STRING
        pop     si
        pop     cx
        ret
```

```
BYTE_IN_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
PRINT_STRING PROC near
        push    ax
        mov     ah, 09h
        int     21h
        pop     ax
        ret
PRINT_STRING ENDP
PRINT_REG PROC  near
        push    cx
        push    dx
        mov     cx, ax
        call    BYTE_IN_HEX
        mov     al, ch
        call    BYTE_IN_HEX
        mov     dx, offset ENDL_LINE
        call    PRINT_STRING
        pop     dx
        pop     cx
        ret
PRINT_REG ENDP

PRINT_TAIL PROC near
```

```
        push ax
        push cx
        push si
        mov cx, 0
        mov cl, ds:[80h]
        cmp CL, 0
        je PT_END
        mov si, 0
        mov ax, 0
CYCLE:  mov al, ds:[81h+si]
        call PRINT_BYTE
        add si, 1
        loop CYCLE
PT_END: mov dx, offset ENDL_LINE
        call PRINT_STRING
        pop si
        pop cx
        pop ax
        ret
PRINT_TAIL ENDP

PRINT_BYTE PROC near
        push ax
        push dx
        mov dx, 0h
        mov dl, al
        mov ah, 02h
        int 21h
        pop dx
        pop ax
        ret
PRINT_BYTE ENDP
```

```
PRINT_ENV PROC near
        push ax
    push es
        push si
        push dx
        push bx
        mov bx, 2Ch
        mov es, [bx]
        pop bx
        mov si, 0
        mov al, es:[si]
SYMBOL: cmp al, 0
        jne NOLINE
        mov dx, offset ENDL_LINE
        call PRINT_STRING
        jmp LOOPER
NOLINE: call PRINT_BYTE
LOOPER: add si, 1
        mov al, es:[si]
        mov ah, es:[si+1]
        cmp ax, 0
        jne SYMBOL
        mov dx, offset ENDL_LINE
        call PRINT_STRING
        call PRINT_PATH
        pop dx
        pop si
        pop es
        pop ax
        ret
PRINT_ENV ENDP
```

```
PRINT_PATH PROC near
        mov dx, offset LOAD_PATH
        call PRINT_STRING
        add si, 4
SYMB:   mov al, es:[si]
        cmp al, 0
        je P_END
        call PRINT_BYTE
        add si, 1
        jmp SYMB
P_END:  ret
PRINT_PATH ENDP
BEGIN:
        push    dx
        push    ax

        mov     dx, offset UNAVAILABLE_ADDRESS
        call    PRINT_STRING
        mov     ax, ds:[02h]
        call    PRINT_REG

        mov     dx, offset ENVIRONMENT_ADDRESS
        call    PRINT_STRING
        mov     ax, ds:[2Ch]
        call    PRINT_REG

        mov     dx, offset COMMAND_LINE_TAIL
        call    PRINT_STRING
        call    PRINT_TAIL

        mov     dx, offset ENVIRONMENT_CONTENT
```

```
        call    PRINT_STRING
        call    PRINT_ENV

ENDING:
        pop     ax
        pop     dx

        xor     al, al
        mov     ah, 4ch
        int     21h
        ret
VERSER  ENDS
        END     START
```