**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. 0304 |  | Шквиря Е. В. |
| Преподаватель |  | Губкин А. Ф. |

Санкт-Петербург

2022

**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.

**Порядок выполнения работы.**

**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.

**Необходимые сведения для составления программы**

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.

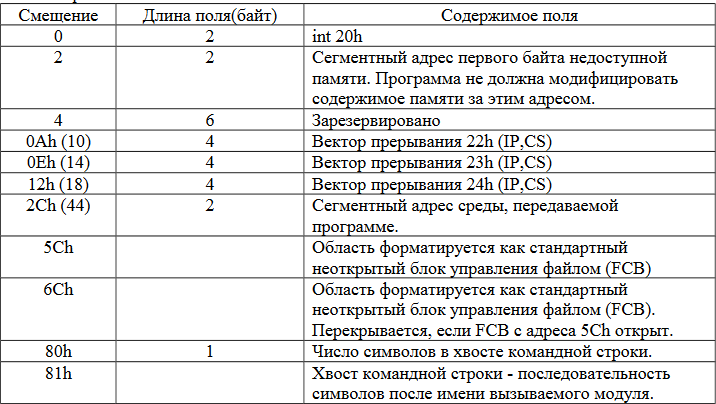
Формат PSP:  


Рис.1 – Таблица данных, хранящихся в PSP

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.

**Контрольные вопросы по лабораторной работе No2**

**Сегментный адрес недоступной памяти**

1) На какую область памяти указывает адрес недоступной памяти?

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

3) Можно ли в эту область памяти писать?

**Среда передаваемая программе**

1) Что такое среда?

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

3) Откуда берется информация, записываемая в среду?

**Выполнение работы.**

1. Для выполнения поставленной задачи был написан код исходного .COM модуля (см. Приложение А, файл *lab2.asm*), который выводит нужную по заданию информацию. Для удобства вывода в сегмент данных были добавлены строки, добавляющиеся в начало каждого ответа.
2. Процедура *PRINT\_UNREACHABLE\_MEMORY* получает адрес недоступной памяти путём считывания значения по адресу 02h. После данный адрес добавляется в строку, и она выводится в консоль.
3. Процедура *PRINT\_ENVIRONMENT\_MEMORY* получает адрес памяти среды путём считывания значения по адресу 2Сh. После данный адрес добавляется в строку, и она выводится в консоль.
4. Процедура *PRINT\_TAIL\_CMD* получает данные, введённые в консоль после имени вызываемого модуля. По адресу 80h берётся длина хвоста, добавляется в регист *CL* и запускается цикл *loop*. Если длина хвоста нулевая, то считывание пропускается. Иначе в цикле каждый раз будет браться символ строки «хвоста» и выводиться в консоль.
5. Процедура *PRINT\_ENVIRONMENT\_CONTENT\_PATH* осуществляет считывание содержимого области среды и путь загружаемого модуля. Для начала происходит посимвольное считывание всех данных от адреса 2Ch вплоть до подряд идущих 00h и 01h. Если между этим были найдены два нулевых байта – выводится перенос вывода на новую строку. После считывания данных считывается путь к загрузочному модулю до момента, когда найден байт 00h.

Результат работы программы приведён на рис. 2:

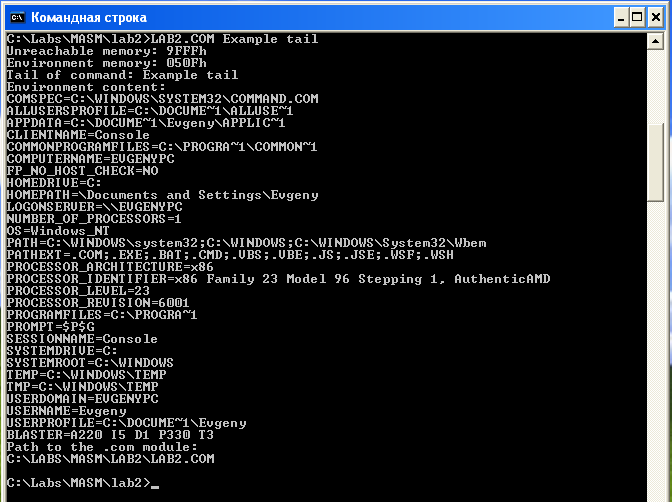


Рис. 2 – Результат работы программы

По рисунку видно, что программа: вывела два адреса – адрес недоступной памяти и адрес памяти среды; вывела переданный хвост «Example tail»; вывела содержимое окружения; вывела путь к загрузочному модулю «C:\Labs\MASM\LAB2\LAB2.COM».

Контрольные вопросы:

**Сегментный адрес недоступной памяти**

1. Адрес недоступной памяти указывает на сегментный адрес области основной оперативной памяти.
2. Данный адрес находится в начале первого сегмента программы (PSP) по адресу 02h.
3. Да. В качестве пример, можно привести данный код:

mov di, ds:[2ch]

mov ds:[di], 12h

mov ds:[di+1], 34h

mov al, ds:[di]

mov ah, ds:[di+1]

После этого в регистре AX будет лежать значение 3412.

**Среда передаваемая программе**

1. Среда является областью памяти, в которой расположены все переменные окружения, на котором программа запускается, в виде строковых переменных по типу Имя=значение. В них может хранится различная информация, например, имя пользователя (на рис.2 – «USERNAME=Evgeny»), или системный диск (на рис.2 – «HOMEDRIVE=C:»).
2. Среда задаётся программе во время запуска, причём для каждой программы она будет зависеть от системы, на которой она запускается. То есть программа получает «локальную копию» содержимого среды, зависящей от машины, на которой данная программа запускается.
3. Существует файл AUTOEXEC.BAT, расположенный в корневом каталоге загрузочного устройства с DOS. Данный файл служит для установки переменных среды и хранит их в себе. При запуске приложения информация об окружении берётся из этого файла.

**Вывод**

В ходе работы был исследован интерфейс управляющей программы, состоящий из блока, содержащего адреса и системные данные. На примере работы с системой WindowsXP был рассмотрен префикс сегмента программы (PSP) и была реализована программа, получающая данные из него и обрабатывающая их.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

Файл: lab2.asm

; 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

    UNREACHABLE\_MEMORY      db 'Unreachable memory: 0000h', 0Dh, 0Ah, '$'

    ENVIRONMENT\_MEMORY      db 'Environment memory: 0000h', 0Dh, 0Ah, '$'

    CMD\_TAIL                db 'Tail of command:$'

    NEXT\_LINE               db 0Dh, 0Ah, '$'

    ENVIRONMENT\_CONTENT     db 'Environment content:', 0Dh, 0Ah, '$'

    PATH\_COM\_MODULE         db 'Path to the .com module:', 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 в 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

        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

    ;-------------------------------

    PRINT\_MESSAGE PROC near

        push ax

        ; сообщение должно быть в dx

        mov ah, 09

        int 21h

        pop ax

        ret

    PRINT\_MESSAGE ENDP

    PRINT\_UNREACHABLE\_MEMORY PROC near

        push ax

        push dx

        mov ax, ds:[02h]

        mov di, offset UNREACHABLE\_MEMORY

        add di, 23

        call WRD\_TO\_HEX

        mov dx, offset UNREACHABLE\_MEMORY

        call PRINT\_MESSAGE

        pop dx

        pop ax

        ret

    PRINT\_UNREACHABLE\_MEMORY ENDP

    PRINT\_ENVIRONMENT\_MEMORY PROC near

        push ax

        push dx

        mov ax, ds:[2ch]

        mov di, offset ENVIRONMENT\_MEMORY

        add di, 23

        call WRD\_TO\_HEX

        mov dx, offset ENVIRONMENT\_MEMORY

        call PRINT\_MESSAGE

        pop dx

        pop ax

        ret

    PRINT\_ENVIRONMENT\_MEMORY ENDP

    PRINT\_TAIL\_CMD PROC near

        push ax

        push cx

        push dx

        mov dx, offset CMD\_TAIL

        call PRINT\_MESSAGE

        mov cx, 0h

        mov cl, ds:[80h]

        mov di, 0h

        cmp cl, 0h

        je EMPTY\_TAIL

READ\_TAIL\_CMD:

        mov al, ds:[81h+di]

        inc di

        mov dl, al

        mov ah, 02h

        int 21h

        loop READ\_TAIL\_CMD

EMPTY\_TAIL:

        mov dx, offset NEXT\_LINE

        call PRINT\_MESSAGE

        pop dx

        pop cx

        pop ax

        ret

    PRINT\_TAIL\_CMD ENDP

    PRINT\_ENVIRONMENT\_CONTENT\_PATH PROC near

        push ax

        push bx

        push cx

        push dx

        mov dx, offset ENVIRONMENT\_CONTENT

        call PRINT\_MESSAGE

        mov di, 00h

        mov dx, 00h

        mov ds, ds:[2Ch]

START\_READ\_CONTENT:

        cmp byte ptr [di], 00h

        je END\_LINE

        mov dl, [di]

        mov ah, 02h

        int 21h

        jmp IS\_END\_CONTENT

END\_LINE:

        cmp byte ptr [di+1], 00h

        je IS\_END\_CONTENT

        push ds

        mov cx, cs

        mov ds, cx

        mov dx, offset NEXT\_LINE

        call PRINT\_MESSAGE

        pop ds

IS\_END\_CONTENT:

        inc di

        mov dx, word ptr [di]

        cmp dx, 0001h

        je READ\_PATH

        jmp START\_READ\_CONTENT

READ\_PATH:

        push ds

        mov ax, cs

        mov ds, ax

        mov dx, offset PATH\_COM\_MODULE

        call PRINT\_MESSAGE

        pop ds

        add di, 2

PRINT\_PATH:

        mov bl, [di]

        cmp bl, 00h

        je END\_READ\_CONTENT

        mov dl, bl

        mov ah, 02h

        int 21h

        inc di

        jmp PRINT\_PATH

END\_READ\_CONTENT:

        mov cx, cs

        mov ds, cx

        mov dx, offset NEXT\_LINE

        call PRINT\_MESSAGE

        pop dx

        pop cx

        pop bx

        pop ax

        ret

    PRINT\_ENVIRONMENT\_CONTENT\_PATH ENDP

    BEGIN:

        call PRINT\_UNREACHABLE\_MEMORY

        call PRINT\_ENVIRONMENT\_MEMORY

        call PRINT\_TAIL\_CMD

        call PRINT\_ENVIRONMENT\_CONTENT\_PATH

    EXIT:

    ; Выход в DOS

        xor AL,AL

        mov AH,4Ch

        int 21H

    TESTPC ENDS

END START ;конец модуля, START - точка входа