**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.**

Студентка гр. 0382 _____ Чегодаева Е.А.

Преподаватель _____ Ефремов М.А.

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.

**Задание.**

***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***. Оформить отчет в соответствии с требованиями. В отчет включить скриншот с запуском программы и результатами.

**Теоретические сведения.**

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:

| **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.**

На основе шаблона .COM модуля, приведённого в методических указаниях, была реализована программа, которая считывает и вывод требуемые данные. Для этого в шаблон были добавлены сообщения, относящиеся к каждому из рассматриваемых параметров. К имеющимся в программе процедурам были добавлены:

- Вспомогательные процедуры, для осуществления вывода:
  PRINT
  PRINT_SYM
- _UNV — определение адреса недоступной памяти, взятого из PSP, в шестнадцатеричном виде. Осуществляется посредством обращения к значению, хранящемуся по адресу DS:[2h].
- _ENV — определение сегментного адреса среды, передаваемой программе, в шестнадцатеричном виде. Реализовано на основе считывания значения по адресу DS:[2Ch].
- _CMD — вывод хвоста командной строки в символьном виде. Посредством обращение к адресу DS:[80h] (длина возможного хвоста) происходит проверка на наличие этого хвоста путём сравнения полученного значения с 0. Если хвост – пуст, то это будет отражено в выводе. В ином случае, при помощи команды loop, реализуется считывание символов (с DS:[81h]) хвоста с одновременным выводом.
- _ENV_Cnt_and_PATH — Вывод содержимого области среды в символьном виде и пути загружаемого модуля. Для осуществления первого этапа реализовано посимвольное считывание строк по адресу DS:[2Ch] (учитывая особенности окончание содержимого области среды), далее – вывод полученных строк. Для получения пути загружаемого модуля пропускаются два байта, и начинается считывание и вывод необходимой информации по тому же принципу.

Результаты запуска программы:

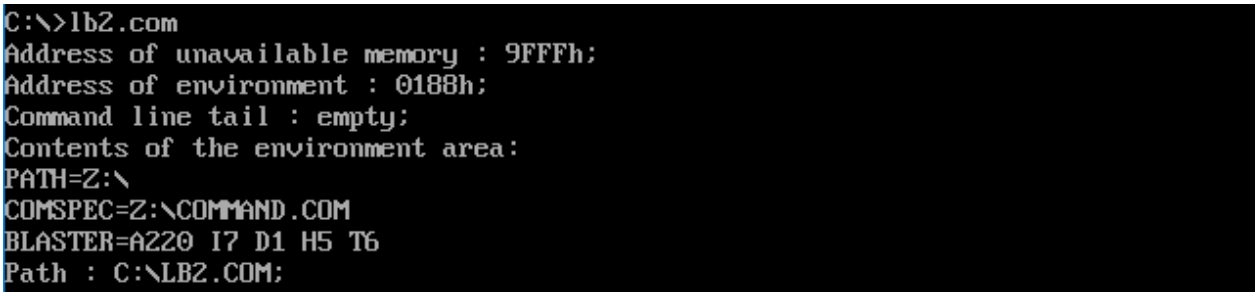


Рис. 1. Результат запуска lb2.com

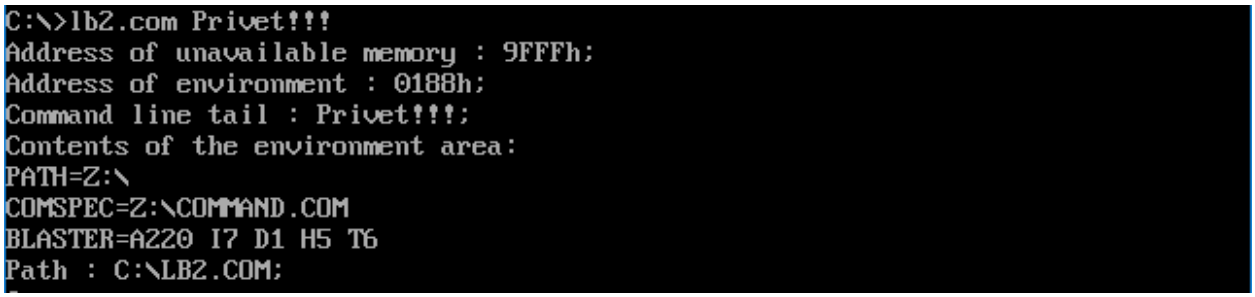


Рис. 2. Результат запуска lb2.com с “хвостом”

Исходный код программы см. в приложении А

**Контрольные вопросы.**

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

- Ответ: Адрес недоступной памяти указывает на сегмент, расположенный следом за тем, что отведён программе.

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

- Ответ: Этот адрес расположен в PSP по смещению 2h — первый байт памяти, идущий за областью, выделенной под программу.

3) Можно ли в эту область памяти писать?

- Ответ: В эту область памяти можно писать, и связанно это с тем, что в MS DOS нет защиты от перезаписи.

Среда, передаваемая программе

1) Что такое среда?

- Ответ: Среда представляет собой область памяти, которая хранит последовательность символьных строк вида: «имя=параметр», которые содержат информацию о системе.

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

- Ответ: Изначально среда создаётся при запуске ОС. Когда запускается приложение — создается копия данной среды, в которую, если в этом есть необходимость, вносятся требуемые приложением дополнительные параметры.

3) Откуда берется информация, записываемая в среду?

- Ответ: Информация, записываемая в среду, берётся из системного пакетного файла AUTOEXEC.BAT, который расположен в корневом каталоге загрузочного устройства.

**Выводы.**

Был исследован интерфейс управляющей программы и загрузочных модулей, также исследован префикс сегмента программы (PSP) и среды, передаваемой программе.

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2.asm

```
TESTPC SEGMENT
     ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
     ORG  100H
START:    jmp BEGIN
; Д А Н Н Ы Е
UNV db 'Address of unavailable memory :     h;', 0DH, 0AH, '$'
ENV db 'Address of environment :     h;', 0DH, 0AH, '$'
CMD db 'Command line tail :', '$'
CMD_Emp db ' empty;', 0DH, 0AH, '$'
END_C db ';', 0DH, 0AH, '$'
ENV_Cnt db 'Contents of the environment area:',0DH, 0AH,'$'
PATH db 'Path :', '$'
END_P db ';', '$'

; П Р О Ц Е Д У Р Ы
;--------------------------------------------------
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
; б а й т  в AL  п е р е в о д и т с я  в  д в а  с и м в о л а  16-г о  ч и
 с л а  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  с т а р ш а я  ц и ф р а
     pop CX          ; в AH  м л а д ш а я
     ret
BYTE_TO_HEX ENDP
;----------------------------
WRD_TO_HEX PROC near
; п е р е в о д  в  16 с/с 16-т и  р а з р я д н о г о  ч и с л а
; в AX - ч и с л о, в DI - а д р е с  п о с л е д н е г о  с и м в о л а
     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 10 с/с, в 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_1
     or AL, 30h
     mov [SI], AL
end_1:  pop DX
     pop CX
     ret
BYTE_TO_DEC ENDP
;----------------------
; КОД

PRINT PROC near
     mov AH, 09h
     int 21h
     ret
PRINT ENDP

PRINT_SYM PROC near
     push AX
     mov AH, 02h
     int 21h
     pop AX
```

```
     ret
PRINT_SYM ENDP

_UNV PROC near
     mov DI, offset UNV
     add DI, 35
     mov AX, DS:[2h]
     call WRD_TO_HEX
     mov DX, offset UNV
     call PRINT
     ret
_UNV ENDP

_ENV PROC near
     mov DI, offset ENV
     add DI, 28
     mov AX, DS:[2Ch]
     call WRD_TO_HEX
     mov DX, offset ENV
     call PRINT
     ret
_ENV ENDP

_CMD PROC near
     mov DX, offset CMD
     call PRINT
     mov CL, DS:[80h]
     cmp CL, 0
     je empty
     mov BX, 81h

lp:
          mov DL, DS:[BX]
          call PRINT_SYM
          inc BX
          loop lp

     mov DX, offset END_C
     call PRINT
     ret
empty:
     mov DX, offset CMD_Emp
     call PRINT
     ret
_CMD ENDP

_ENV_Cnt_and_PATH PROC near

ENV_Contents:
     mov DX, offset ENV_Cnt
     call PRINT
     mov ES, DS:[2Ch]
     xor DI, DI
```

```
     mov DL, ES:[DI]
read_env:
     cmp DL, 0
     je final_env
     call PRINT_SYM
     inc DI
     mov DL, ES:[DI]
     jmp read_env

final_env:
     mov DL, 0Dh
     call PRINT_SYM
     mov DL, 0Ah
     call PRINT_SYM
     inc DI
     mov DL, ES:[DI]
     cmp DL, 00h
     jne read_env

module_PATH:
     mov DX, offset PATH
     call PRINT
     add DI, 2
     mov DL, ES:[DI]
     inc DI

read_path:
     call PRINT_SYM
     mov DL, ES:[DI]
     inc DI
     cmp DL, 0
     jne read_path

     mov DX, offset END_P
     call PRINT
     ret
_ENV_Cnt_and_PATH ENDP

BEGIN:
     call _UNV
     call _ENV
     call _CMD
     call _ENV_Cnt_and_PATH
; Выход в DOS
     xor AL, AL
     mov AH, 4Ch
     int 21h
TESTPC  ENDS
     END START
```