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 _____ Михайлова О.Д.

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.

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.

Оформление отчета в соответствии с требованиями. В отчет включите скриншот с запуском программы и результатами.

Выполнение работы.

Для выполнения задания был использован шаблон из методических указаний с процедурами перевода двоичных кодов в символы шестнадцатеричных чисел и десятичное число: TETR_TO_HEX, BYTE_TO_HEX, WRD_TO_HEX, BYTE_TO_DEC.

Для получения необходимой информации была использована адресация вида ds:[<смещение>], так как данный сегментный регистр указывает на начало PSP.

Также в программу были добавлены следующие процедуры:

- PRINT_STRING – процедура вывода строки на экран.
- PRINT_SYMB – процедура вывода символа на экран.
- MemAddr – получение сегментного адреса недоступной памяти в шестнадцатеричном виде и вывод его на экран.
- EnvAddr – получение сегментного адреса среды, передаваемой программе, и вывод его на экран.
- CTail – получение хвоста командной строки в символьном виде. Для этого сначала в регистр cx записывается число символов в хвосте командной строки, а затем посимвольно считывается хвост и выводится на экран.
- EnvCont – получение содержимого области среды в символьном виде. Информация так же считывается и выводится посимвольно.
- MPath – получение пути загружаемого модуля. Путь так же выводится посимвольно.

Результаты работы программы представлены на рисунках 1, 2.

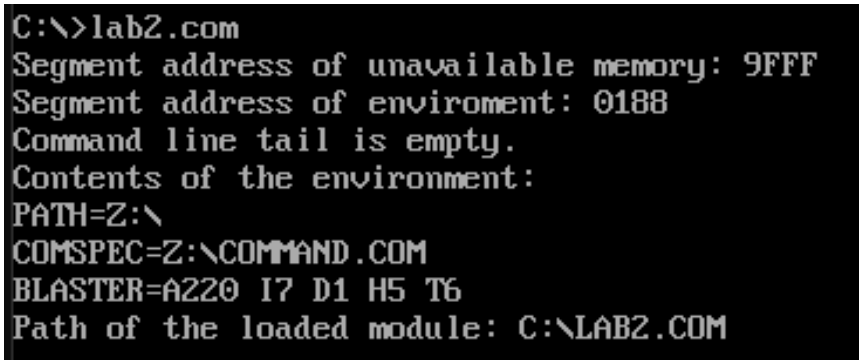

Рисунок 1 - результат работы программы без хвоста

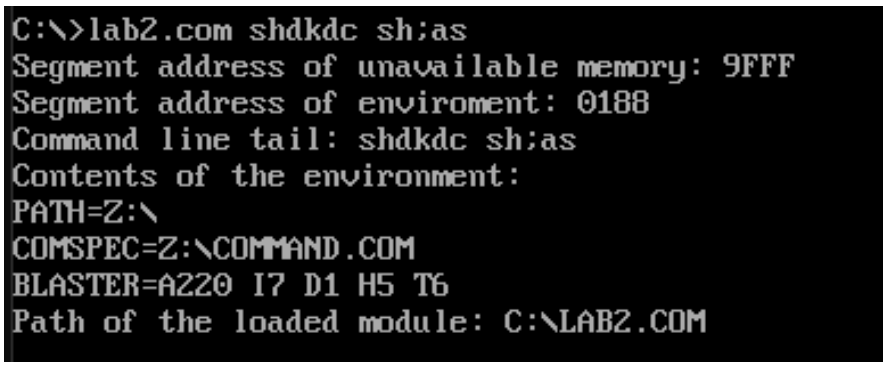

Рисунок 2 - результат работы программы с хвостом

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

На значение сегментного адреса памяти, следующей за памятью, выделенной программе.

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

Сразу после памяти, выделенной программе, в PSP по смещению 2.

3. Можно ли в эту область памяти писать?

Да, можно, так как в MS DOS нет защиты от перезаписи памяти.

Среда передаваемая программе

1. Что такое среда?

Среда – это область памяти, которая хранит последовательность строк, содержащих информацию в виде: имя=параметр

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

Среда создается при загрузке операционной системы. Эта среда копируется в адресное пространство запущенной программы.

3. Откуда берется информация, записываемая в среду?

Из системного пакетного файла AUTOEXEC.BAT.

Выводы.

В ходе работы были изучены интерфейс управляющей программы и загрузочных модулей, а также PSP и среда, передаваемая программе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab2.asm

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

; Данные
MEM_ADDR db 'Segment address of unavailable memory:
',0DH,0AH,'$'
ENV_ADDR db 'Segment address of enviroment:    ',0DH,0AH,'$'
TAIL db 'Command line tail:','$'
EMPTY_TAIL db 'Command line tail is empty.',0Dh,0Ah,'$'
ENV_CONT db 'Contents of the environment:',0Dh,0Ah,'$'
PATH db 'Path of the loaded module: ','$'

; 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 шест. 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_STRING PROC near
    mov AH, 09h
    int 21h
    ret
PRINT_STRING ENDP

PRINT_SYMB PROC near
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
PRINT_SYMB ENDP

MemAddr PROC near
    mov ax, ds:[2h]
    mov di, OFFSET MEM_ADDR + 42
    call WRD_TO_HEX
    mov dx, OFFSET MEM_ADDR
    call PRINT_STRING
    ret
MemAddr ENDP

EnvAddr PROC near
```

```
    mov ax, ds:[2Ch]
    mov di, OFFSET ENV_ADDR+34
    call WRD_TO_HEX
    mov dx, OFFSET ENV_ADDR
    call PRINT_STRING
    ret
EnvAddr ENDP

CTail PROC near
    mov cl, ds:[80h]
    cmp cl, 0h
    je empty_t
    mov dx,offset TAIL
    call PRINT_STRING
    mov di,81h
loop_tail:
    mov dl, ds:[di]
    call PRINT_SYMB
    inc di
    loop loop_tail
    mov dl, 0Dh
    call PRINT_SYMB
    mov dl, 0Ah
    call PRINT_SYMB
    jmp final_tail
empty_t:
    mov dx, OFFSET EMPTY_TAIL
    call PRINT_STRING
final_tail:
    ret
CTail ENDP

EnvCont PROC near
    mov dx, OFFSET ENV_CONT
    call PRINT_STRING
    mov es, ds:[2Ch]
    xor di, di
loop_env:
    mov dl, es:[di]
    cmp dl, 0h
    je final_env
    call PRINT_SYMB
    inc di
    jmp loop_env
final_env:
    mov dl, 0Dh
    call PRINT_SYMB
    mov dl, 0Ah
    call PRINT_SYMB
    inc di
    mov dl, es:[di]
    cmp dl, 0h
    jne loop_env
    ret
EnvCont ENDP
```

```
MPath PROC near
    add di, 3
    mov dx, OFFSET PATH
    call PRINT_STRING
loop_path:
    mov dl, es:[di]
    cmp dl, 0h
    je final_path
    call PRINT_SYMB
    inc di
    jmp loop_path
final_path:
    mov dl, 0Dh
    call PRINT_SYMB
    mov dl, 0Ah
    call PRINT_SYMB
    ret
MPath ENDP

; Код
BEGIN:
    call MemAddr
    call EnvAddr
    call CTail
    call EnvCont
    call MPath
    xor AL,AL
    mov AH,4Ch
    int 21H
TESTPC ENDS
END START
```