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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студентка гр. 9383 _____ Карпекина А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

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 в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Сведения о функциях и структурах.

TETR_TO_HEX - перевод десятичной цифры в код символа (записывается в AL)

BYTE_TO_HEX - переводит байт из AL в два символа шестнадцатеричного числа в AX

WRD_TO_HEX - перевод в 16 с/с 16-ти разрядного числа (в AX число, DI - адрес последнего символа)

BYTE_TO_DEC - перевод в 10 с/с (SI - адрес поля младшей цифры)

MEM_AD - выводит сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде

ENV_AD - выводит сегментный адрес среды, передаваемый программе, в шестнадцатеричном виде

TAIL - выводит хвост командной строки в символьном виде

ENV - выводит содержимое области среды в символьном виде и путь загружаемого модуля

Выполнение работы.

Была написана программа, которая выводит всю необходимую пользователю информацию. Последовательно сначала функция MEM_AD выводит сегментный адрес недоступной памяти из PSP, а ENV_AD выводим сегментный адрес среды. TAIL выводит хвост командной строки, если он пуст

то в консоли печатается соответствующее сообщение. В конце ENV печатает содержимое области среды и путь, который проходит загружаемый модуль.

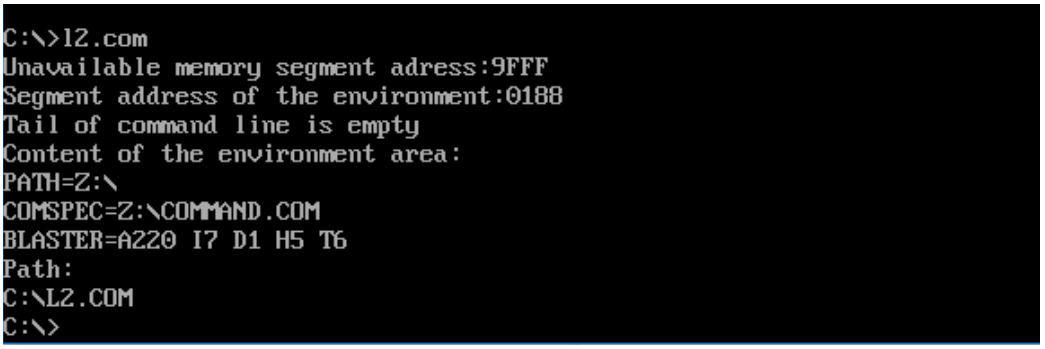

Рисунок 1 - Пример работы программы с пустым хвостом

Рисунок 2 - Пример работы программы №2

Ответы на вопросы.

Сегментный адрес недоступной памяти.

1) На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на значение сегментного адреса первого байта за памятью, отведенной программе.

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

Сразу за памятью, отведенной программе (в PSP по смещению 2ch).

3) Можно ли в эту область памяти написать?

Можно, так как DOS не имеет защиты от перезаписи памяти для сторонних программ, которым эта память не выделялась.

Среда передаваемая программе.

1) Что такое среда?

Совокупность значений системных переменных, путей, открытых файловых хэндлов и других ресурсов операционной системы, передаваемые процессу (программе) при его запуске.

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

Первоначальная среда создается при запуске ОС. Среда копируется в адресное пространство программы и может быть изменена, по требованию программы. Когда одна программа запускается другой, то ей передается родительская среда.

3) Откуда берется информация, записываемая в среду?

Из файла AUTOEXEC.BAT, расположенного в корневом каталоге загрузочного устройства.

Вывод.

В процессе работы была написана программа для ознакомления с PSP, были изучены интерфейсы программных модулей DOS.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

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


UN_MEM_AD db  'Unavailable memory segment adress:     ',0DH,0AH,'$'
SEG_ENV_AD db 'Segment address of the environment:    ',0DH,0AH,'$'
STR_TAIL db 'Tail of command line:          ',0DH,0AH,'$'
TAIL_EMPTY db 'Tail of command line is empty    ',0DH,0AH,'$'
CONTENT_ENV db 'Content of the environment area:',0DH,0AH, '$'
STR_END db 0DH,0AH, '$'
PATH db 'Path:  ',0DH,0AH, '$'



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

BYTE_TO_DEC PROC near
        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

MEM_AD PROC near
   mov ax, ds:[02h]
   mov di, offset UN_MEM_AD
   add di, 37
   call WRD_TO_HEX
   mov dx, offset UN_MEM_AD
```

```
    mov AH,09h
    int 21h
    ret
MEM_AD   ENDP

ENV_AD PROC near
    mov ax, ds:[02Ch]
    mov di, offset SEG_ENV_AD
    add di, 38
    call WRD_TO_HEX
    mov dx, offset SEG_ENV_AD
    mov AH,09h
    int 21h
    ret
ENV_AD ENDP

TAIL PROC near
    xor cx, cx
    mov cl, ds:[80h]
    mov si, offset STR_TAIL
    add si, 21
    cmp cl, 0h
    je empty
    mov di, 0
    mov ax, 0

read:
    mov al, ds:[81h+di]
    inc di
    mov [si], al
    inc si
    loop read
    mov dx, offset STR_TAIL
    jmp print

empty:
    mov dx, offset TAIL_EMPTY

print:
    mov AH,09h
    int 21h
    ret
TAIL ENDP
```

```
ENV PROC near
    mov dx, offset CONTENT_ENV
    mov AH,09h
    int 21h
    xor di, di
    mov ds, ds:[2Ch]
r_str:
    cmp byte ptr [di], 0
    je s_end
    mov dl, [di]
    mov ah, 02h
    int 21h
    jmp find
s_end:
    cmp byte ptr [di+1],00h
    je find
    push ds
    mov cx, cs
    mov ds, cx
    mov dx, offset STR_END
    mov AH,09h
    int 21h
    pop ds
find:
    inc di
    cmp word ptr [di], 0001h
    je r_path
    jmp r_str
r_path:
    push ds
    mov ax, cs
    mov ds, ax
    mov dx, offset PATH
    mov AH,09h
    int 21h
    pop ds
    add di, 2
loop_p:
    cmp byte ptr [di], 0
    je break
    mov dl, [di]
    mov ah, 02h
    int 21h
    inc di
```

```
    jmp loop_p
break:
    ret
ENV ENDP

BEGIN:
    call MEM_AD
    call ENV_AD
    call TAIL
    call ENV

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