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. 9382 _____ Рыжих Р.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Постановка задачи

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 данных.

В данной программе используются следующие функции и структуры данных:

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа, который записывается в AL
BYTE_TO_HEX	Перевод значений байта в число 16-ой СС и его представление в виде двух символов
WRD_TO_HEX	Перевод слова в число 16-ой СС и представление его в виде четырех символов
BYTE_TO_DEC	Перевод значения байта в число 10-ой СС и представляет его в виду символов
WRITESTRING	Вывод строки на экран
WRITE_ADR_1	Печать на экран сегментный адрес недоступной памяти из PSP
WRITE_ADR_2	Печать на экран сегментный адрес среды, передаваемой программе
WRITE_TAIL	Печать на экран хвоста командной строки
WRITE_CONTENT	Печать на экран содержимого области среды
WRITE_PATH	Печать на экран путь загружаемого модуля

Выполнение шагов лабораторной работы:

1 шаг:

1) На экран печатается сегментный адрес недоступной памяти из PSP в шестнадцатеричном виде

2) На экран печатается сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде

3) На экран печатается хвост командной строки в символьном виде

4) На экран печатается содержимое области среды в символьном виде

5) На экран печатается путь загружаемого модуля

Результаты, полученные программой:

```
C:\>LAB2.COM
Segment address of inaccessible memory: 9FFFH
Environment segment address: 0188H
Command line tail is empty
Environment area content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
C:\LAB2.COM
```

Рис. 1. - Пример работы программы

```
C:\>LAB2.COM PUTIN POMOGI
Segment address of inaccessible memory: 9FFFH
Environment segment address: 0188H
Command line tail:
 PUTIN POMOGI
Environment area content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
C:\LAB2.COM
```

Рис. 2. - Пример работы программы

2 шаг:

Был оформлен отчет в соответствии с требованиями. В отчете включены скриншоты с запуском программы и результатами.

Ответы на контрольные вопросы

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на адрес следующего сегмента памяти после участка памяти, отведенного под программу.

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

В PSP по смещнию 02h в сторону бОльших адресов.

3) Можно ли в эту область памяти писать?

Можно, потому что DOS не имеет механизмов защиты перезаписи памяти программ, для которых эта память не выделялась.

Среда передаваемая программе

1) Что такое среда?

Переменные, в которых хранятся некоторые настройки операционной системы. Переменные среды хранят информацию о состоянии системы.

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

Во время загрузки модуля в оперативную память

3) Откуда берется информация, записываемая в среду?

Данная информация берется из файла AUTOEXEC.BAT, который расположен в корневом каталоге загрузочного устройства.

Заключение.

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

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: Lab2.asm

```
TESTPC SEGMENT
   ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
   ORG 100H
START: JMP BEGIN
; Данные
   SEG_ADR_1 db  'Segment address of inaccessible memory:
H',0DH,0AH,'$';41
   SEG_ADR_2 db  'Environment segment address:
H',0DH,0AH,'$';30
   TAIL  db 'Command line tail:',0DH,0AH,'$'
   CONTENT  db 'Environment area content:',0DH,0AH,'$'
   PATH  db 'Loadable module path:',0DH,0AH,'$'
   EMPTY db 'Command line tail is empty',0DH,0AH,'$'
   NEW_STR db 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 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
;-------------------------------
WRITESTRING PROC near
   mov AH,09h
   int 21h
   ret
WRITESTRING ENDP

PRINT_INFO PROC near

write_adr_1:
   mov ax, ds:[02h]
   mov di, offset SEG_ADR_1
   add di, 43
   call WRD_TO_HEX
   mov dx, offset SEG_ADR_1
   call WRITESTRING

write_adr_2:
   mov ax, ds:[2Ch]
   mov di, offset SEG_ADR_2
   add di, 32
   call WRD_TO_HEX
```

```
    mov dx, offset SEG_ADR_2
    call WRITESTRING

write_tail:
    push ax
    push cx
    xor ax, ax
    xor cx, cx
    mov cl, ds:[80h]
    cmp cl,0
    je if_0
    mov dx, offset TAIL
    call WRITESTRING
    mov di, 0

write_tail_symbol:
    mov dl, ds:[81h + di]
    inc di
    mov ah, 02h
    int 21h
    loop write_tail_symbol

    mov dx, offset NEW_STR
    call WRITESTRING
    jmp end_tail

if_0:
    mov dx, offset EMPTY
    call WRITESTRING

end_tail:
    pop cx
    pop ax

write_content:
    push dx
```

```
    push ax
    push si
    push ds
    mov dx, offset CONTENT
    call WRITESTRING
    xor si, si
    mov ds, ds:[2CH]

write_content_symbol:
    mov dl,[si]
    cmp dl,00h
    je endl

    inc si
    mov ah, 02h
    int 21h
    jmp write_content_symbol

endl:
    inc si
    mov dl,[si]
    cmp dl,00h
    je end_content

    pop ds
    mov dx, offset NEW_STR
    call WRITESTRING
    push ds
    mov ds,ds:[2Ch]
    jmp write_content_symbol

end_content:
    pop ds
    mov dx, offset NEW_STR
    call WRITESTRING
    push ds
```

```
   mov ds,ds:[2Ch]
   add si, 3

write_path:
   mov dl,[si]
   cmp dl,0
   je end_path

   mov ah,02h
   int 21h
   inc si
   jmp write_path

end_path:
   pop ds
   pop si
   pop ax
   pop dx
   ret
PRINT_INFO ENDP

; Код
BEGIN:
   call PRINT_INFO

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