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. 0382 _____ Тюленев Т.В.

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.
6. 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.

Выполнение работы.

Для выполнения заданий по лабораторной работе был написан файл **`lab2_os.asm`**.

Основные процедуры:

print_unavailable – процедура вывода адреса сегмента первого байта недоступной памяти с помощью чтения соответствующих байтов в PSP.

print_env_address – процедура вывода сегментного адреса среды с помощью чтения соответствующих байтов в PSP.

print_input_string – процедура печати хвоста командной строки. Сначала считывается количество символов в хвосте, а затем происходит их печать с помощью прерывания **21h** с кодом **2**.

print_string – процедура печати всех символов, начиная с адреса в регистре **si** до нулевого.

print_environment_content_and_path – процедура печати содержимого области среды и пути содержимого модуля, используя процедуру выше.

Контрольные вопросы.

Сегментный адрес недоступной памяти.

1) На какую область памяти указывает адрес недоступной памяти?

На сегментный адрес основной оперативной памяти, расположенной после программы, т. е. на первый сегмент после памяти, выделенной программе.

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

Сегментный адрес расположен в PSP по адресу 2Ch.

3) Можно ли в эту область памяти писать?

Да, можно, т. к. в DOS общее адресное пространство.

Среда, передаваемая программе.

1) Что такое среда?

Среда – область памяти, где хранятся определенные параметры системы,

называемые переменными среды.

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

Среда создается при загрузке ОС, но может меняться перед запуском приложения, в соответствии с его требованиями.

3) Откуда берется информация, записываемая в среду?

Из системного пакетного файла AUTOEXEC.BAT, который расположен в корневом каталоге загрузочного устройства.

Выводы.

Был исследован интерфейс управляющей программы и загрузочных модулей. Была написана программа печатающая сегментный адрес недоступной памяти, сегментный адрес среды, хвост командной строки, содержимое области среды и путь загружаемого модуля.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
LAB2  Segment
        Assume CS:LAB2, DS:LAB2,
ES:NOTHING, SS:NOTHING
        ORG 100H
START: JMP BEGIN

unavailable_memory_msg db 'Unavailable
memory address:     ', 0ah, '$'
segment_env_addres_msg db 'Environment
address:     ', 0ah, '$'
input_string db 'Input string:', '$'

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

print PROC NEAR
  push ax
  mov ah, 09h
  int 21h
  pop ax
  ret
print ENDP

print_unavailable proc near
  push ax
  push di
  push dx

  mov ax, es:[02h]
  mov di, offset unavailable_memory_msg
  add di, 31
  call wrd_to_hex
  mov dx, offset unavailable_memory_msg
  call print

  pop dx
  pop di
  pop ax
  ret
print_unavailable endp

print_env_address proc near
  push ax
  push di
  push dx

  mov ax, es:[02Ch]
  mov di, offset segment_env_addres_msg
  add di, 24
  call wrd_to_hex
  mov dx, offset segment_env_addres_msg
  call print
```

```
  pop dx
  pop di
  pop ax
  ret
print_env_address endp

print_input_string proc near
  push dx
  push cx
  push si
  push ax

  mov dx, offset input_string
  call print
  mov cl, ds:[80h]
  mov si, 081h
  mov ah, 02h
  cmp cl, 0
  je end_


  print_symbol:
    mov dl, [si]
    int 21h
    inc si
    loop print_symbol


  end_:
    mov dl, 0ah
    int 21h

  pop ax
  pop si
  pop cx
  pop dx

  ret

print_input_string endp

print_string proc near
  push dx
  push ax
  mov ah, 02h

  print_sym:
    mov dl, ds:[si]
    inc si

    cmp dl, 0
    jz end_of_string

    int 21h
    jmp print_sym

  end_of_string:
```

```
    mov dl, 0ah
    int 21h

  pop ax
  pop dx
  ret
print_string endp

print_environment_content_and_path proc
near
  push ds
  push si
  push ax
  push cx
  push dx

  mov ds, es:[2ch]
  mov si, 0

  print_strings:
    call print_string
    mov bl, ds:[si]
    cmp bl, 0
    jz print_path
    jmp print_strings

  print_path:
    add si, 3
    call print_string

  the_end:
    pop dx
    pop cx
    pop ax
    pop si
    pop ds

  ret

print_environment_content_and_path endp

BEGIN:

  call print_unavailable
  call print_env_address
  call print_input_string
  call
print_environment_content_and_path

  xor al, al
  mov ah, 4Ch
  int 21H

LAB2  ENDS
        END     START
```