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 _____ Охотникова Г.С.

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.

Выполнение работы.

При выполнении данной лабораторной работы в шаблон из методических указаний были добавлены следующие процедуры:

- PRINT_STRING — выводит на экран строку.
- PRINT_SYMBOL — выводит на экран символ, содержащийся в регистре DL.
- PRINT_NMEM — выводит на экран сегментный адрес недоступной памяти в шестнадцатеричном виде.

- PRINT_ENV — выводит на экран сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.
- PRINT_TAIL — выводит на экран хвост командной строки в символьном виде.
- PRINT_CONT — выводит на экран содержимое области среды в символьном виде.
- PRINT_PATH — выводит на экран путь загружаемого модуля.

Пример работы программы:

Рисунок 1 — пример работы, если нет хвоста

Рисунок 2 — пример работы, если есть хвост

Исходный программный код см. в приложении А.

Контрольные вопросы.

Сегментный адрес недоступной памяти.

1) На какую область памяти указывает адрес недоступной памяти?

Ответ: Указывает на сегмент, который расположен сразу после памяти, которая выделена под программу.

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

Ответ: Это первый байт памяти, который расположен после памяти, которая выделена под программу, то есть начиная с адреса 9FFF.

3) Можно ли в эту область памяти писать?

Ответ: Можно, потому что DOS не имеет защиты памяти от перезаписи.

Среда, передаваемая программе.

1) Что такое среда?

Ответ: Среда — это область памяти, которая хранит информацию о состоянии системы: «имя = параметр».

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

Ответ: Среда создается при запуске операционной системы. Затем, при запуске программы, создается копия среды в адресном пространстве программы.

3) Откуда берется информация, записываемая в среду?

Ответ: В корневом каталоге расположен системный файл autoexec.bat, информация берется из него.

Выводы.

При выполнении данной лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей. Была написана программа, которая выводит на экран информацию об адресе недоступной памяти, среде и пути.

ПРИЛОЖЕНИЕ А.

Название файла: lab2.asm

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

NMEM db 'Not available memory:     h', 0Dh, 0Ah,'$'
ENVIROMENT db 'Enviroment:     h', 0Dh, 0Ah, '$'
TAIL db 'Tail: ', '$'
ENV_CT db 'Enviroment content: ', '$'
PATH db 'Path: ', '$'

;STRING db 'Значение регистра AX= ',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 шестн. 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_SYMBOL PROC near
push AX
mov AH, 02h
int 21h
pop AX
ret
PRINT_SYMBOL ENDP

PRINT_NMEM PROC near
mov AX, DS:[2h]
mov DI, offset NMEM
add DI, 25
call WRD_TO_HEX
mov DX, offset NMEM
call PRINT_STRING
ret
PRINT_NMEM ENDP

PRINT_ENV PROC near
mov AX, DS:[2Ch]
mov DI, offset ENVIROMENT
add DI, 15
call WRD_TO_HEX
mov DX, offset ENVIROMENT
call PRINT_STRING
ret
```

```
PRINT_ENV ENDP

PRINT_TAIL PROC near
mov CL, DS:[80h]
mov DX, offset TAIL
call PRINT_STRING
cmp CL, 0
je end_proc
mov DI, 81h

loop_proc:
mov DL, DS:[DI]
call PRINT_SYMBOL
inc DI
loop loop_proc

end_proc:
mov DL, 0DH
call PRINT_SYMBOL
mov DL, 0AH
call PRINT_SYMBOL
ret
PRINT_TAIL ENDP

PRINT_CONT PROC near
   mov ES, DS:[2Ch]
   xor DI, DI
   mov DX, offset ENV_CT
   call PRINT_STRING

label1:
   mov DL, ES:[DI]
   cmp DL, 0
   je end_proc2
   call PRINT_SYMBOL
   inc DI
   jmp label1

end_proc2:
```

```
   mov DL, 0DH
   call PRINT_SYMBOL
   mov DL, 0AH
   call PRINT_SYMBOL
   inc DI
   mov DL, ES:[DI]
   cmp DL, 0
   jne label1
   ret
PRINT_CONT ENDP

PRINT_PATH PROC near
add DI, 3
mov DX, offset PATH
call PRINT_STRING

label_path:
mov DL, ES:[DI]
cmp DL, 0
je end_path
call PRINT_SYMBOL
inc DI
jmp label_path

end_path:
mov DL, 0DH
call PRINT_SYMBOL
mov DL, 0AH
call PRINT_SYMBOL
ret

PRINT_PATH ENDP

; КОД
BEGIN:
call PRINT_NMEM
call PRINT_ENV
call PRINT_TAIL
call PRINT_CONT
```

```
call PRINT_PATH
 xor AL,AL
 mov AH,4Ch
 int 21H
TESTPC ENDS
 END START ;конец модуля, START - точка входа
```