МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе № 2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студент гр. 8383 _____ Аверина О.С.

Преподаватель _____ Губкин А. Ф.

Санкт-Петербург

2020

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

Постановка задачи.

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 стоки в символьном виде.
4. Содержимое области среды в символьном виде.
5. Путь загружаемого модуля.

Выполнение работы.

Был написан код .COM модуля, который читает из PSP сегментный адрес недоступной памяти, сегментный адрес среды, передаваемой программе и выводит на экран вместе с хвостом командной строки, содержимым области среды и путем загружаемого модуля. Данный код был собран в .COM модуль.

В результате выполнения были получены следующие значения(рис.1):

Рисунок 1 – результат работы программы

Выводы.

В ходе лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей, а также исследован префикс сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А
ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

Сегментный адрес недоступной памяти:

1) На какую область памяти указывает адрес недоступной памяти?

   Адрес указывает на окончание основной оперативной памяти, которая расположена с 0000h по 9FFFh.

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

   Адрес расположен сразу за окончанием выделенной памяти.

3) Можно ли в эту область памяти писать?

   Можно, так как DOS не имеет возможности защищать память от изменений.

Среда, передаваемая программе:

1) Что такое среда?

   Это текстовая переменная операционной системы, хранящая какую-либо информацию — например, данные о настройках системы. Область среды содержит последовательность символьных строк вида: имя = параметр. Каждая строка завершается байтом нулей. В первой строке указывается имя COMSPEC, которая определяет используемый командный процессор и путь к COMMAND.COM. Следующие строки содержат информацию, задаваемую командами PATH, PROMPT, SET.

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

   Среда создается при загрузке программы в память.

3) Откуда берётся информация, записываемая в среду?

   Переменные среды устанавливаются пользователем или сценариями оболочки. В Windows переменные среды задаются в реестре Windows и программным обеспечением.

ПРИЛОЖЕНИЕ А

Исходный код программы.

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

; Данные
MEM_ADRESS db 'Inaccessible memory address:     h',13,10,'$'
ENV_ADRESS db 'Environment address:     h',13,10,'$'
TAIL db 'Command line tail:        ',13,10,'$'
NULL_TAIL db 'In Command tail no sybmols',13,10,'$'
CONTENT db 'Content:',13,10, '$'
END_STRING db 13, 10, '$'


;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
;
          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
;--------------------------------
WRITE_PROC PROC near
          mov AH,09h
          int 21h
          ret
```

```
WRITE_PROC ENDP

PROC_MEMORY PROC near   ; вывод сегментного адреса недоступной
памяти

        mov ax,ds:[02h]

        mov di, offset MEM_ADRESS
        add di, 32

        call WRD_TO_HEX
        mov    dx, offset MEM_ADRESS
        call WRITE_PROC
        ret
PROC_MEMORY ENDP



PROC_ENVIROMENT  PROC near   ; вывод сегментного адреса среды,
передаваемой программе
          mov ax,ds:[2Ch]

          mov di, offset ENV_ADRESS
          add di, 24

          call WRD_TO_HEX
          mov dx, offset ENV_ADRESS
          call WRITE_PROC
          ret
PROC_ENVIROMENT ENDP
```

```
PROC_TAIL PROC near

        xor cx, cx
          mov cl, ds:[80h]
          mov si, offset TAIL
          add si, 19
        cmp cl, 0h
        je EMPTY_TAIL   ; считываем число символов в хвосте ком.
строки,
          xor di, di          ; если не пустой, выводим
          xor ax, ax

     READ_TAIL:
          mov al, ds:[81h+di]
          inc di
          mov [si], al
          inc si
          loop READ_TAIL
          mov dx, offset TAIL
          jmp END_TAIL

     EMPTY_TAIL:
          mov dx, offset NULL_TAIL

     END_TAIL:
          call WRITE_PROC
          ret
PROC_TAIL ENDP
```

```
PROC_CONTENT PROC near

        mov dx, offset CONTENT
        call WRITE_PROC

        xor di,di
        mov ds, ds:[2Ch]        ;; вывод сегментного

     READ_STRING:
          cmp byte ptr [di], 00h
          jz END_STR

          mov dl, [di]
          mov ah, 02h
          int 21h
          jmp DETECT_END

     END_STR:
         cmp byte ptr [di+1],00h
         jz DETECT_END
         push ds
         mov cx, cs
          mov ds, cx
          mov dx, offset END_STRING
          call WRITE_PROC
          pop ds

     DETECT_END:
```

```
          inc di
          cmp word ptr [di], 0001h
          jz READ_PATH

          jmp READ_STRING

     READ_PATH:
          add di, 2

     LOOP_PATH:
          cmp byte ptr [di], 00h
          jz  DONE
          mov dl, [di]
          mov ah, 02h
          int 21h
          inc di
          jmp LOOP_PATH

     DONE:
          ret
PROC_CONTENT ENDP

BEGIN:

call PROC_MEMORY
call PROC_ENVIROMENT
call PROC_TAIL
call PROC_CONTENT
```

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