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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов командных модулей

Студент гр. 8383 _____ Бессуднов Г. И.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург

2020

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Ход работы.

В ходе выполнения лабораторной работы был написан и отлажен программный модуль типа **.COM**, который выбирает и распечатывает на экран следующую информацию:

1) Сегментный адрес недоступной памяти 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.

Результат работы представлен на рис. 1 и на рис. 2, исходный код программы содержится в Приложении А.

Рисунок 1 – Результат работы программы без входных данных

Рисунок 2 – Результат работы программы с входными данными

Контрольные вопросы

Ниже приведены ответы на контрольные вопросы:

1. Сегментный адрес недоступной памяти
   1.1. Адрес недоступной памяти указывает на служебную часть памяти. Она не может быть выделена DOS под программу.
   1.2. За областью памяти, отведенной пользовательским программам.
   1.3. Да, можно. В DOS отсутствуют механизмы защиты памяти.
2. Среда, передаваемая программе
   2.1. Это последовательность символьных строк вида [имя] = [параметр]. Строка заканчивается байтом нулей.
   2.2. При запуске DOS создается корневая среда. Когда же происходит запуск пользовательской программы, или же одна программа запускает другую, то запущенная программа получает свой экземпляр блока среды, который копирует блок среды родителя, но может быть изменен в соответствии с нуждами программы. Таким образом среда создается при загрузке ОС, но каждая программа может изменить свою копию.
   2.3. Из файла AUTOEXEC.BAT.

Выводы.

В ходе выполнения работы был исследован интерфейс управляющей программы и загрузочных модулей, а так же интерфейс префикса сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
TESTPC SEGMENT
ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
ORG 100H
START: JMP BEGIN
NEW_LINE db 13, 10, '$'
LOCK_MEM_MES db 'Locked memory: $'
ENV_MES db 'Environment: $'
TAIL_MES db 'Command line tail: $'
NO_TAIL_MES db 'empty as always $'
ENV_CON_MES db 'Enviroment content: $'
PATH_MES db 'Path: $'

;-----------------------------------------
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
;-----------------------------------------
PRINT_SYMBOL PROC near
    push AX
    mov AH, 02H
    int 21H
    pop AX
    ret
PRINT_SYMBOL ENDP
;-----------------------------------------
PRINT_MESSAGE PROC near
    push AX
    mov AH, 09H
    int 21H
```

```
    pop AX
    ret
PRINT_MESSAGE ENDP
;-------------------------------------
PRINT_HEX PROC near
    push AX
    mov AL, AH
    call BYTE_TO_HEX
    mov DL, AH
    call PRINT_SYMBOL
    mov DL, AL
    call PRINT_SYMBOL
    pop AX
    call BYTE_TO_HEX
    mov DL, AH
    call PRINT_SYMBOL
    mov DL, AL
    call PRINT_SYMBOL
    ret
PRINT_HEX ENDP
;-------------------------------------

BEGIN:
;--------LOCK_MEM--------
    mov DX, offset LOCK_MEM_MES
    call PRINT_MESSAGE
    mov AX, DS:[02H]
    call PRINT_HEX
    mov DX, offset NEW_LINE
    call PRINT_MESSAGE
;--------ENV--------
    mov DX, offset ENV_MES
    call PRINT_MESSAGE
    mov AX, DS:[2CH]
    call PRINT_HEX
    mov DX, offset NEW_LINE
    call PRINT_MESSAGE
;--------TAIL--------
    mov DX, offset TAIL_MES
    call PRINT_MESSAGE
    mov CX, 0
    mov CL, DS:[80H]
    cmp CL, 0
    je TAIL_EMPTY
    mov DI, 0
     xor DX, DX
TAIL_LOOP:
     mov DL, DS:[81H + DI]
     call PRINT_SYMBOL
     inc DI
```

```
      loop TAIL_LOOP
      jmp TAIL_END
TAIL_EMPTY:
      mov DX, OFFSET NO_TAIL_MES
      call PRINT_MESSAGE
TAIL_END:
    mov DX, offset NEW_LINE
    call PRINT_MESSAGE
;--------ENV_PATH_MES--------
    mov DX, offset ENV_CON_MES
    call PRINT_MESSAGE
    xor SI, SI
      mov BX, 2CH
      mov ES, [BX]
PRINT_CONTENT:
      cmp BYTE PTR ES:[SI], 0H
      je NEXT_CONTENT
      mov AL, ES:[SI]
    mov DL, AL
      call PRINT_SYMBOL
      jmp CHECK
NEXT_CONTENT:
      mov DX, offset NEW_LINE
      call PRINT_MESSAGE
CHECK:
      inc SI
      cmp WORD PTR ES:[SI], 0001H
    je PRINT_PATH
      jmp PRINT_CONTENT
PRINT_PATH:
      mov DX, offset PATH_MES
      call PRINT_MESSAGE
      add SI, 2
PATH_LOOP:
      cmp BYTE PTR ES:[SI], 00H
      je CONTENT_END
      mov AL, ES:[SI]
    mov DL, AL
      call PRINT_SYMBOL
      inc SI
      jmp PATH_LOOP
CONTENT_END:

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