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 _____ Кузьмин Д. И.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург

2021

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. 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.

1. Были созданы строки, содержащие начальную информацию.
2. Далее были созданы процедуры, извлекающие данные из PSP и выводящие их на экран.
3. Затем в главной процедуре были вызваны все остальные, а затем осуществлен выход из программы.

Контрольные вопросы.

1.1 На какую область память указывает адрес недоступной памяти?

На первый байт после памяти, отведенной программе.

1.2 Где расположен этот адрес по отношению к области памяти, отведенной программе?

Он расположен после области памяти, отведенной программе, в сторону увеличивающихся адресов

1.3 Можно ли в эту область памяти писать?

Можно, так как в DOS нет механизмов защиты.

2.1 Что такое среда?

Множество переменных, хранящих информацию о системе и передаваемых программе

2.2 Когда создается среда? Перед запуском приложения или в другое время?

Создается во время запуска ОС, но во время запуска приложения может меняться.

2.3 Откуда берется информация, записываемая в среду?

Из файла autoexec.bat

Выводы.

Был изучен интерфейса управляющей программы и загрузочных модулей. Разработана программа, извлекающая информацию из PSP и выводящая ее на экран.

ПРИЛОЖЕНИЕА. ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ

```
H:\>lb2.com -a -b -c -d -f -g
Unavailable memory adress:9FFF
Environment adress:0188
Tail:
 -a -b -c -d -f -g
Environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Path to COM:
H:\LB2.COM
H:\>
```

Выполнение программы

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД

Файл lb2.asm

```
TESTPC SEGMENT
 ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
 ORG 100H
START: JMP BEGIN
; Данные
mem_adress db 'Unavailable memory adress:         ', 0DH, 0AH, '$'
environment_adress db 'Environment adress:         ', 0DH, 0AH, '$'
tail db 'Tail:',0DH, 0AH, '$'
env_content db 0DH, 0AH ,'Environment content:', 0DH, '$'
path db 0DH, 0AH,'Path to COM:', 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
;--------------------------------
WRITE_MSG PROC near
                mov AH, 09h
                int 21h
                ret
WRITE_MSG 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, в AH - младшая
                pop CX
                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 10чную с/с, SI - адрес 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_MEM_ADRESS PROC near
                mov DI, offset mem_adress
                add DI, 29
                mov AX, DS:[02h]
                call WRD_TO_HEX
                mov DX, offset mem_adress
                call WRITE_MSG
                ret
PRINT_MEM_ADRESS ENDP
;-------------------------------
PRINT_ENV_ADRESS PROC near
                mov DI, offset environment_adress
                add DI, 22
                mov AX, DS:[2ch]
                call WRD_TO_HEX
                mov DX, offset environment_adress
                call WRITE_MSG
                ret
PRINT_ENV_ADRESS ENDP
;-------------------------------
PRINT_TAIL PROC near
                mov DX, offset tail
                call WRITE_MSG
                mov AH, DS:[80h]
                xor DI, DI
                check_num:
                cmp AH, 0
                jle exit
                mov AL, DS:[81h + DI]
```

```
                int 29h
                dec AH
                inc DI
                jmp check_num
                exit:
                ret
PRINT_TAIL ENDP
;---------------------------------
PRINT_ENV_CONTENT PROC near
                mov DX, offset env_content
                call WRITE_MSG
               mov     AX, DS:[2ch]
                mov     DS,AX
                xor   SI, SI
                mov   DX,SI

newstr:    call newline
                call print_str

nxt:
                cmp byte ptr [SI+1],0
                jne newstr

                push DS
                mov CX, CS
                mov DS, CX
                mov DX, offset path
                call WRITE_MSG
                pop DS
                lodsw
last:
                mov CX, AX
                call    print_str
                loop    last
                ret
print_str:
                lodsb
                cmp AL, 0
                jz ex
                int 29h
                jmp print_str
          ex:
                ret
newline:
                mov AX, 0D0Ah
                call  print_wrd
                ret
print_wrd:
                int 29h
                xchg    AH, AL
                int 29h
                ret
PRINT_ENV_CONTENT ENDP
;---------------------------------
; КОД
BEGIN:
                call PRINT_MEM_ADRESS
```

```
                call PRINT_ENV_ADRESS
                call PRINT_TAIL
                call PRINT_ENV_CONTENT
;Выход в DOS
                xor AL,AL
                mov AH,4Ch
                int 21H
TESTPC ENDS
 END START ; Конец модуля, start - точка входа
```