МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студент гр. 0382 _____ Корсунов А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Постановка задачи

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.

Сведения о функциях и структурах данных управляющей программы.

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа ASCII (результат записывается в AL)
BYTE_TO_HEX	Перевод байта из AL переводится в два символа шестн. числа в AX (в AL старшая цифра в AH младшая цифра)
WRD_TO_HEX	Перевод в 16 с/с 16-ти разрядного числа (в AX – число, DI – адрес последнего символа)
BYTE_TO_DEC	Перевод в 10 с/с (SI – адрес поля младшей цифры)
WRITE_MESSAGE_WORD	Вывод строки на экран
WRITE_MESSAGE_BYTE	Вывод символа на экран
PRINT_UNAVAILABLE_MEMORY	Вывод сегментного адреса недоступной памяти из PSP в шестнадцатеричном виде на экран
PRINT_ENVIRONMENT	Вывод сегментного адреса среды, передаваемой программе, в шестнадцатеричном виде на экран
PRINT_COMMAND_LINE_END	Вывод хвоста командной строки в символьном виде на экран

PRINT_CONTENT_ENV_AREA_AND _LOADED_MODULE_PATH	Выводы содержимого области среды в символьном виде и пути загружаемого модуля на экран

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.

Сохранить результаты, полученные программой, и включить их в отчет.

Шаг 2. Оформить отчет в соответствии с требованиями. В отчет включить скриншот с запуском программы и результатами. Ответить на контрольные вопросы.

Выполнение работы

Шаг 1. Был написан и отлажен программный модуль типа **.COM**, который выбирает распечатывает необходимую в задании информацию.

Рисунок 1. Демонстрация работы программного модуля типа .COM (хвост командной строки — пустой)

Рисунок 2. Демонстрация работы программного модуля типа .COM (хвост командной строки не пустой)

Шаг 2. На основе написанной программе были даны ответы на контрольные вопросы.

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной среды?

Ответ: адрес недоступной среды указывает на значение адреса первого байта сразу после памяти, выделенную под программу.

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

Ответ: Сразу после памяти, выделенной под программу — в PSP по смещению 2.

3. Можно ли в эту область памяти писать?

Ответ: Да — в DOS нет защиты от перезаписи памяти.

Среда передаваемая программе

1. Что такое среда?

Ответ: Среда — это область памяти, хранящая переменные среды, хранящие информацию о состоянии системы в формате имя=значение.

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

Ответ: Среда создается при запуске ОС (соответственно, в DOS – при запуске DOS). Она же копируется в адресное пространство запущенной программы.

3. Откуда берется информация, записываемая в среду?

Ответ: Эта информация берется из файла AUTOEXEC.BAT.

Вывод.

В ходе выполнения лабораторной работы было проведено исследование интерфейса управляющей программы и загрузочных модулей.. исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Приложение А

Файл com.asm

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

UNAVAILABLE_MEMORY db 'Address of unavailable memory segment:     ', 0DH, 0AH, '$'
ENVIRONMENT db 'Address of environment segment:     ', 0DH, 0AH, '$'
CONTENT_ENV_AREA db 'Contents of environment area:   ', 0DH, 0AH, '$'
COMMAND_LINE_END_EMPTY db 'End of command line: empty', 0DH, 0AH, '$'
COMMAND_LINE_END db 'End of command line:$'
LOADED_MODULE_PATH db 'Path of loaded module:$'

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

WRITE_MESSAGE_WORD PROC near
        push AX
        mov AH, 09h
        int 21h
        pop AX
        ret
WRITE_MESSAGE_WORD ENDP

WRITE_MESSAGE_BYTE PROC near
        push AX
```

```
        mov AH, 02h
        int 21h
        pop AX
        ret
WRITE_MESSAGE_BYTE ENDP

PRINT_UNAVAILABLE_MEMORY PROC near
        push AX
        push DI
        push DX

        mov AX, DS:[02h]
        mov DI, offset UNAVAILABLE_MEMORY
        add DI, 42
        call WRD_TO_HEX
        mov DX, offset UNAVAILABLE_MEMORY

        call WRITE_MESSAGE_WORD

        pop DX
        pop DI
        pop AX

        ret
PRINT_UNAVAILABLE_MEMORY ENDP

PRINT_ENVIRONMENT PROC near
        push AX
        push DI
        push DX

        mov AX, DS:[02Ch]
        mov DI, offset ENVIRONMENT
        add DI, 35
        call WRD_TO_HEX
        mov DX, offset ENVIRONMENT

        call WRITE_MESSAGE_WORD

        pop DX
        pop DI
        pop AX

        ret
```

```
PRINT_ENVIRONMENT ENDP

PRINT_COMMAND_LINE_END PROC near
        push AX
        push DI
        push CX
        push DX

        xor CX, CX

        mov CL, DS:[80h]
        cmp CL, 0h
        je empty_cont
        xor DI, DI

        mov DX, offset COMMAND_LINE_END
        call WRITE_MESSAGE_WORD

        cycle:
                mov DL, DS:[81h+DI]
                call WRITE_MESSAGE_BYTE
                inc DI
        loop cycle
        mov DL, 0Dh
        call WRITE_MESSAGE_BYTE
        mov DL, 0Ah
        call WRITE_MESSAGE_BYTE
        jmp final

        empty_cont:
                mov DX, offset COMMAND_LINE_END_EMPTY
                call WRITE_MESSAGE_WORD

        final:
                pop DX
                pop CX
                pop DI
                pop AX
        ret
PRINT_COMMAND_LINE_END ENDP

PRINT_CONTENT_ENV_AREA_AND_LOADED_MODULE_PATH PROC near
        push AX
        push DI
```

```
push DX
push ES

mov DX, offset CONTENT_ENV_AREA
call WRITE_MESSAGE_WORD
xor DI, DI
mov AX, DS:[2Ch]
mov ES, AX

cycle_02:
        mov DL, ES:[DI]
        cmp DL, 0h
        je end_word
        call WRITE_MESSAGE_BYTE
        inc DI
        jmp cycle_02

end_word:
        mov DL, 0Ah
        call WRITE_MESSAGE_BYTE
        inc DI
        mov DL, ES:[DI]
        cmp DL, 0h
        je final_02
        call WRITE_MESSAGE_BYTE
        inc DI
        jmp cycle_02

final_02:
        mov DX, offset LOADED_MODULE_PATH
        call WRITE_MESSAGE_WORD
        add DI, 3
        cycle_03:
                mov DL, ES:[DI]
                cmp DL, 0h
                je final_03
                call WRITE_MESSAGE_BYTE
                inc DI
                jmp cycle_03

final_03:
        pop ES
        pop DX
        pop DI
```

```
            pop AX
        ret
PRINT_CONTENT_ENV_AREA_AND_LOADED_MODULE_PATH ENDP

BEGIN:
        call PRINT_UNAVAILABLE_MEMORY
        call PRINT_ENVIRONMENT
        call PRINT_COMMAND_LINE_END
        call PRINT_CONTENT_ENV_AREA_AND_LOADED_MODULE_PATH

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