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. 9383 _____ Ноздрин В.Я.

Преподаватель _____ Ефремов М.А.

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 сегментный регистр. Исследование префикса сегмента програмы (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 среду, 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.

Основные теоретические положения.

При начальной загрузке программы формируется PSP, который рахмещается в начале первого сегмента программы. PSP занимает 256 байт и располагается с адреса кратного границе сегмента. При загрузке модулей типа .COM все сегментные регистры указывают на адрес PSP. При загрузке модуля типа .EXE сегментные регистры DS и ES указывают на PSP. Именно по этой причине значения этих регистров в модуле .EXE следует переопределять.

Формат PSP:

Смещение	Длина поля (байт)	Содержимое поля
0	2	Int 20h
2	2	Сегментный адрес первого байта недоступной памяти. Программе не должна модифицировать содержимое памяти за этим адресом.
4	6	Зарезервировано
0Ah(10)	4	Вектор прерывания 22h (IP,CS)
0Eh(14)	4	Вектор прерывания 23h (IP,CS)
12h(18)	4	Вектор прерывания 24h (IP,CS)
2Ch(44)	2	Сегментный адрес среды, передаваемой программе.
5Ch		Область формируется как стандартный неоткрытый блок управления файлом (FCB)
6Ch		Область формируется как стандартный неоткрытый блок управления файлом (FCB). Перекрывается, если FCB с адреса 5Ch открыт.
80h	1	Число символов в хвосте командной строки
81h		Хвост командной строки – последовательнос тьсимволов после имени вызываемого модуля.

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, PROMPR, SET.

Среда заканчивается также байтом нулей. Таким образом, два нулевый байта являются признаком конца переменных среды. Затем идут два байта, содержащих 00h, 01h, после которых располагается маршрут загруженной программы. Маршрут также заканчивается байтом 00h.

Выполнение работы.

Шаг 1. Написан текст исходного .COM модуля lab2.asm.

Шаг 2. Модуль скомпилирован и отлажен. Результаты работы сохранены и занесены в отчет.

Используемые функции

TETR_TO_HEX – функция, переводящая десятичную цифру в код символа.

BYTE_TO_HEX – функция, переводящая байт в шестнадцатеричной системе счисления в код символа.

WRD_TO_HEX – функция, переводящая шестнадцатеричное число в символьный код.

BYTE_TO_DEC – функция, переводящая байт в шестнадцатеричной системе счисления в символьныый код десятичной системы счисления.

PRINT_STRING – функция, выводящая строку на экран

UMA – функция, выводящая на экран информацию о недоступной памяти.

ENV_A – функция, выводящая на экран информацию о среде.

C_TAIL – функция, выводящая на экран хвост командной строки в символьном виде.

ENV_C – функция, выводящая на экран содержимое области среды и путь до выполняемого модуля.

Контрольные вопросы.

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на область памяти, которую программа не должна изменять.

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

Адрес недоступной памяти указывает на первый байт следующий сегментом выделенным для программы, в области увеличивающихся адресов.

3) Можно ли в эту область памяти писать?

В область недоступной памяти писать можно, потому что в MS DOS отсутствует контроль доступа к памяти.

Среда передаваемая программе

1) Что такое среда?

Среда это область памяти, хранящая переменные среды в виде строк символов вида «имя=парамметр». В этих переменных записана информация о состоянии

системы, такая как путь к домашней директории, путь к модулю COMMAND.COM, данные об используемом командноим процессоре и другое.

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

Среда создается при загрузке операционной системы. При запуске программы среда копируется в его адресное пространство.

3) Откуда берется инфомация, записываемая в среду?

Информация, записываемая в среду берется из системного файла AUTOEXEC.BAT.

Выводы.

В процессе выполнения лабораторной работы был изучен интерфейс управляющей программы и загрузочных модулей, порефикс сегмента программы PSP и передаваемая программе среда.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД.

Название файла: lab2.asm

```
TESTPC  SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG 100H
START:  JMP BEGIN
; DATA
UNAVAILABLE_MEMORY_ADDRESS   DB  'Unavailable  memory  address:
'      ,0DH,0AH,'$'
ENV_ADDRESS                  DB 'Environment segment address:
'     ,0DH,0AH,'$'
COMMAND_TAIL                                DB  'Command  tail:
',0DH,0AH,'$'
COMMAND_TAIL_EMPTY                  DB  'Command  tail:  empty'
,0DH,0AH,'$'
ENV_CONTENT                 DB 'Environment segment content:'
,0DH,0AH,'$'
MODULE_PATH                                 DB  'Module  path:'
,0DH,0AH,'$'
END_STRING                  DB 0DH, 0AH, '$'

; PROCEDURES
;-------------------------------------------------------------
------------------
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_1
        OR      AL, 30h
        MOV     [SI], AL
    END_1:
        POP     DX
        POP     CX
        RET
    BYTE_TO_DEC ENDP
    ;-------------------------------------------------------------
------------------
    ;-------------------------------------------------------------
------------------
    PRINT_STRING PROC near
        PUSH    AX
        MOV     ah, 09h
        INT     21h
        POP     AX
        RET
    PRINT_STRING ENDP
```

```
    ;-------------------------------------------------------------
------------------
    UMA PROC near
        PUSH    AX
        PUSH    DI
        MOV     AX, DS:[02h]
        MOV     DI, offset UNAVAILABLE_MEMORY_ADDRESS
        ADD     DI, 30
        CALL    WRD_TO_HEX
        MOV     DX, offset UNAVAILABLE_MEMORY_ADDRESS
        CALL    PRINT_STRING
        POP     DI
        POP     AX
        RET
    UMA ENDP
    ;-------------------------------------------------------------
------------------
    ENV_A PROC near
        PUSH    AX
        PUSH    CX
        PUSH    DI
        MOV     AX, DS:[2Ch]
        MOV     DI, offset ENV_ADDRESS
        ADD     DI, 32
        CALL    WRD_TO_HEX
        MOV     DX, offset ENV_ADDRESS
        CALL    PRINT_STRING
        POP     DI
        POP     CX
        POP     AX
        RET
    ENV_A ENDP
    ;-------------------------------------------------------------
------------------
```

```
C_TAIL PROC near
    XOR     CX, CX
    MOV     CL, DS:[80h]
    MOV     SI, offset COMMAND_TAIL
    ADD     SI, 15
    CMP     CL, 0h
    je      EMPTY
    XOR     DI, DI
    XOR     AX, AX
READ:
    MOV     AL, DS:[81h+DI]
    INC     DI
    MOV     [SI], AL
    INC     SI
    LOOP    READ
    MOV     DX, offset COMMAND_TAIL
    jmp     PRINT_TAIL
EMPTY:
    MOV     DX, offset COMMAND_TAIL_EMPTY
PRINT_TAIL:
    CALL    PRINT_STRING
    RET
C_TAIL ENDP
;-------------------------------------------------------------
------------------
ENV_C PROC near
    MOV     DX, offset ENV_CONTENT
    CALL    PRINT_STRING
    XOR     DI, DI
    MOV     DS, DS:[2Ch]
READ_STRING:
    CMP     byte ptr [DI], 00h
    jz      END_CONTENT
    MOV     DL, [DI]
```

```
    MOV     AH, 02h
    INT     21h
    jmp     FIND
END_CONTENT:
    CMP     byte ptr [DI+1], 00h
    jz      FIND
    PUSH    DS
    MOV     CX, CS
    MOV     DS, CX
    MOV     DX, offset END_STRING
    CALL    PRINT_STRING
    POP     DS
FIND:
    INC     DI
    CMP     word ptr [DI], 0001h
    jz      PATH
    jmp     READ_STRING
PATH:
    PUSH    DS
    MOV     AX, CS
    MOV     DS, AX
    MOV     DX, offset MODULE_PATH
    CALL    PRINT_STRING
    POP     DS
    ADD     DI, 2
LOOP_PATH:
    CMP     byte ptr [DI], 00h
    jz      EXIT
    MOV     DL, [DI]
    MOV     AH, 02h
    INT     21h
    INC     DI
    jmp     LOOP_PATH
EXIT:
```

```
        RET
    ENV_C ENDP
    ;-------------------------------------------------------------
-----------------
    ; CODE
    BEGIN:
    ; TASK
        XOR     AX, AX
        CALL    UMA
        CALL    ENV_A
        CALL    C_TAIL
        CALL    ENV_C
    ; EXIT TO MS_DOS
        XOR     AL, AL
        MOV     AH, 4Ch
        INT     21h

    TESTPC    ENDS
        END START   ; конец модуля, START - точка входа
```