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. 8381 _____ Ивлева О.А.

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 программы (PSP) и среды, передаваемой программе.

Основные теоретические положения.

При начальной загрузке программы формируется PSP, который размещается 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.

Формат PSP представлен в табл. 1.

Таблица 1 – Формат PSP

Смещение (16-ричн)	Длина поля (байт)	Содержимое поля
0	2	INT 20h
2	2	Сегментный адрес первого байта недоступной памяти. Программа не должна модифицировать содержимое памяти за этим адресом.
4	6	Зарезервировано
0A	4	Вектор прерывания 22h (IP, CS)
0E	4	Вектор прерывания 23h (IP, CS)
12	4	Вектор прерывания 24h (IP, CS)

Окончание табл. 1.

2C	2	Сегментный адрес среды, передаваемой программе.
5C		Область форматируется как стандартный неоткрытый блок управления файлом (FCB)
6C		Область форматируется как стандартный неоткрытый блок управления файлом (FCB). Перекрывается, если FCB с адреса 5Ch открыт.
80	1	Число символов в хвосте командной строки.
81		Хвост командной строки - последовательность символов после имени вызываемого модуля.

Выполнение работы.

Был написан текст исходного .COM модуля. Программа читает содержимое среды DOS и PSP, передаваемые программе. Код программы указан в приложении А.

Программа была отлажена и запущена в DOSBox. Вывод программы представлен на рис. 1.

Рисунок 1 – Вывод .COM модуля

Контрольные вопросы

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

Область недоступной памяти начинается с адреса 9FFFh и заканчивается адресом FFFFh. Она является служебной – в нее DOS не может загрузить пользовательские программы.

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

Адрес недоступной памяти указывает на последний параграф памяти, отведенной для пользовательских программ.

3. Можно ли в эту область памяти писать?

DOS не контролирует обращение программы к памяти, поэтому можно.

Среда, передаваемая программе

1. Что такое среда?

Среда – область памяти, хранящая переменные окружения в виде символьных строк в формате «параметр=значение», которая передается программе при ее запуске. 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 создается среда? Перед запуском приложения или в другое время?

Корневая среда создается во процессе начальной загрузки DOS. Когда COMMAND.COM запускает программу, ей передается копия корневой среды (копируются строки окружения в новую область памяти, находящуюся перед запускаемой программой).

3. Откуда берется информация, записываемая в среду?

Окружение для командного процессора, создаваемое в процессе начальной загрузки чаще всего содержит переменные COMSPEC, PROMPT, PATH, которые заносятся в окружение из файла AUTOEXEC.BAT. При запуске программы ей

передается копия корневой среды. Пользователь может включить в окружение строки определения дополнительных переменных с помощью команды SET.

Выводы.

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

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. LAB2.ASM

```
CODE        SEGMENT

ASSUME      CS:CODE, DS:CODE, ES:NOTHING, SS:NOTHING

ORG         100H

START:      JMP   BEGIN

STR1        DB "Memory Address:      ", 13,10, "$"
STR2        DB "Env. Address:      ", 13,10, "$"
STR3        DB "Tail:                                          ", 13,10, "$"
STR4        DB "Env. Data: ", 13, 10, "$"
STR5        DB 13, 10, "$"
STR6        DB "Path: ", 13, 10, "$"


PRINT       PROC  NEAR

            PUSH  AX
            MOV         AH, 09H
        INT       21H
            POP   AX
        RET

PRINT       ENDP


TETR_TO_HEX PROC  NEAR

        AND       AL, 0FH
        CMP       AL, 09H
        JBE       NEXT
            ADD         AL, 07H

    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, 4H
          SHR AL, CL
            CALL  TETR_TO_HEX
          POP CX
          RET

BYTE_TO_HEX ENDP


WREAD_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

WREAD_TO_HEX      ENDP


      BEGIN:

            MOV         AX, DS:[02H]
            MOV         DI, OFFSET STR1
            ADD         DI, 19
            CALL  WREAD_TO_HEX
            MOV         DX, OFFSET STR1
            CALL  PRINT

            MOV         AX, DS:[2CH]
            MOV         DI, OFFSET STR2
            ADD         DI, 17
            CALL  WREAD_TO_HEX
            MOV         DX, OFFSET STR2
            CALL  PRINT
```

```
        XOR         CX, CX
        XOR         SI, SI
        MOV         CL, DS:[80H]
        CMP         CL, 0
        JE          NO_TAIL
        MOV         DI, OFFSET STR3
        ADD         DI, 6H

  READ:
        MOV         AL, DS:[81H + SI]
        MOV         [DI], AL
        INC         DI
        INC         SI
        LOOP  READ

  NO_TAIL:
        MOV   DX, OFFSET STR3
        CALL  PRINT

        MOV   DX, OFFSET STR4
        CALL  PRINT
        XOR   DI, DI
        MOV   BX, 2CH
        MOV   DS, [BX]

  BEGIN_STRING:
        CMP   BYTE PTR [DI], 00H
        JE          ENTR
        MOV   DL, [DI]
        MOV   AH, 02H
        INT   21H
        JMP   END_DATA

  ENTR:
        PUSH  DS
        MOV   CX, CS
        MOV   DS, CX
        MOV   DX, OFFSET STR5
        CALL  PRINT
        POP   DS

END_DATA:
        INC   DI
        CMP   WORD PTR [DI], 0001H
        JE          PATH
        JMP   BEGIN_STRING
```

```
        PATH:
              PUSH  DS
              MOV   AX, CS
              MOV   DS, AX
              MOV   DX, OFFSET STR6
              CALL  PRINT
              POP   DS
              ADD   DI, 2

        CIRCLE:
              CMP   BYTE PTR [DI], 00H
              JE          END_PATH
              MOV   DL, [DI]
              MOV   AH, 02H
              INT   21H
              INC   DI
              JMP   CIRCLE

        END_PATH:
              PUSH  DS
              MOV   CX, CS
              MOV   DS, CX
              MOV   DX, OFFSET STR5
              CALL  PRINT
              POP   DS

              XOR         AL, AL
              MOV         AH, 4CH
              INT         21H

CODE          ENDS
END           START
```