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 программных модулей»

Студент гр. 8381 _____ Муковский Д.В.

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. Именно по этой причине значения этих регистров в модуле .EXE следует переопределять.

Таблица 1 – Формат PSP

Смещение	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
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 форматируется как стандартный неоткрытый блок управления файлом (FCB)
6Ch		Область форматируется 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 - последовательность символов после имени вызываемого модуля.

Область среды содержит последовательность символьных строк вида: имя = параметр.

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. Среда заканчивается также байтом нулей. Таким образом, два нулевых 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.

Написан и отлажен 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.

Результат выполнения программы представлен на рис. 1.

```
S:\>lr2.com
1.Unavailable memory segment address: 9FFF
2.Segment address of the environment: 0188
3.No command line tail
4.Program environment content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

5.Path:
S:\LR2.COM
```

Рисунок 1 – Вывод программы

Контрольные вопросы.

Сегментный адрес недоступной памяти.

1. **На какую область памяти указывает адрес недоступной памяти?**

   Область недоступной памяти начинается с 9FFFh и заканчивается адресом FFFFh. Ее адрес указывает на служебную часть памяти, которую DOS не может выделить под программу.

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?**

   Да, так как DOS не контролирует обращение программы к памяти.

Среда, передаваемая программе:

1. **Что такое среда?**

   Среда – это область памяти, в которой в виде последовательности символьных строк вида: “имя=параметр, 0” (переменная среды) записаны значения переменных. Среда служит для передачи программам требуемых параметров. Параметры заносятся в среду с помощью системной команды SET. Системные и прикладные программы могут анализировать текущий состав среды и извлекать из него относящиеся к ним параметры.

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

   Начальная среда, в которой будут работать активизируемые программы, создается при начальной загрузке DOS. При запуске программы происходит лишь копирование среды в новую область памяти.

3. **Откуда берется информация, записываемая в среду?**

   Информация, записываемая в среду, берётся из системного файла *autoexec.bat*. Это системный пакетный файл, который содержит информацию о ключевых переменных среды, таких как: PATH, PROMPT и COMSPEC.

Вывод.

В результате выполнения данной лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей. Была написана программа, которая выводит на экран сегментный адрес недоступной памяти, взятый из PSP, сегментный адрес среды, передаваемой программе, хвост командной строки и путь загружаемого модуля.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ LR2.ASM

```
LAB SEGMENT
     ASSUME CS:LAB, DS:LAB, ES:NOTHING, SS:NOTHING
     ORG 100H
MAIN: JMP BEGIN

;данные
UNAVAILABLE_MEMORY         db        "Unavailable memory segment
address:     ", 13, 10, "$"
ENVIRONMENT            db     "Segment address of the environment:
", 13, 10, "$"
LINE_TAIL            db     "Command line tail:                 ",
13, 10, "$"
NO_TAIL                  db      "No command line tail", 13, 10,
"$"
ENVIRONMENT_CONTENT      db      "Program environment content:",
13, 10, "$"
PATH                     db      "Path:", 13, 10, "$"
CONTENT_EMPTY_LINE       db      13, 10, "$"


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
;байт в al переводится в два символа шест. 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
                POP  CX               ;в ah младшая цифра
                RET
        BYTE_TO_HEX     ENDP

        ;-------------------------------------------------------------
----------------

        WRD_TO_HEX PROC NEAR
        ;перевод в 16 с/с 16 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
                XOR  AH, AH
                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 10 с/с, si - адрес поля младшей цифры
                PUSH CX
                PUSH DX
                PUSH AX
```

```
                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   AX, 00H
                JBE   END_L
                OR AL, 30H
                MOV   [SI], AL
      END_L:
                POP   AX
                POP   DX
                POP   CX
                RET
      BYTE_TO_DEC     ENDP
      ;-------------------------------------------------------------------
-----------------

      PRINT PROC NEAR
                PUSH AX
                MOV AH, 09H
                INT 21H
                POP AX
                RET
      PRINT ENDP

      ;-------------------------------------------------------------------
-----------------



      BEGIN:

                PUSH DX
```

```
PUSH AX
UNAVAILABLE_MEMORY_PRINT:

      MOV DI, OFFSET UNAVAILABLE_MEMORY
      ADD DI, 39
      MOV AH, DS:[02H]
      MOV AL, DS:[03H]
      CALL WRD_TO_HEX
      MOV DX, OFFSET UNAVAILABLE_MEMORY
      CALL PRINT


ENVIRONMENT_PRINT:

      MOV DI, OFFSET ENVIRONMENT
      ADD DI, 39
      MOV AH,DS:[2CH]
      MOV AL,DS:[2DH]
      CALL WRD_TO_HEX
      MOV DX, OFFSET ENVIRONMENT
      CALL PRINT


LINE_TAIL_PRINT:

      MOV AL,DS:[80H]
      MOV DI, OFFSET LINE_TAIL
      ADD DI, 18
      TEST AL,AL
      JE NO_TAIL_PRINT
      MOV SI,81H
      TAIL:
            MOV AL, DS:[SI]
            MOV [DI], AL
            INC SI
            INC DI
            LOOP TAIL
      MOV DX, OFFSET LINE_TAIL
      CALL PRINT
```

```
        JMP ENVIRONMENT_CONTENT_PRINT

NO_TAIL_PRINT:
        MOV DX, OFFSET NO_TAIL
        CALL PRINT

ENVIRONMENT_CONTENT_PRINT:
        MOV DX, OFFSET ENVIRONMENT_CONTENT
        CALL PRINT
        XOR DI,DI
        XOR AX,AX
        MOV BX, 2CH
        MOV ES, [BX]
        MOV DX, OFFSET CONTENT_EMPTY_LINE

        LINE_PRINT:
                MOV AL, ES:[DI]
                CMP AL, 0
                JNE PRINT_SYMB
                MOV DX, OFFSET CONTENT_EMPTY_LINE
                CALL PRINT
                INC DI
                MOV AX, ES:[DI]
                CMP AX,0001H
                JE PATH_PRINT
                JMP LINE_PRINT

        PRINT_SYMB:
                MOV DL,AL
                XOR AL,AL
                MOV AH,02H
                INT 21H
                INC DI
                MOV AX, ES:[DI]
                CMP AX,0001H
                JE PATH_PRINT
                JMP LINE_PRINT

PATH_PRINT:
```

```
        ADD DI,2
        MOV DX, OFFSET PATH
        CALL PRINT

    PATH_SYMB_PRINT:
        MOV AL, ES:[DI]
        TEST AL, AL
        JE EXIT
        MOV DL, AL
        MOV AH, 02H
        INT 21H
        INC DI
        JMP PATH_SYMB_PRINT

    EXIT:
    POP       AX
    POP  DX
    XOR       AL, AL
    MOV  AH, 4CH
    INT       21H
    RET

LAB  ENDS
        END  MAIN
```