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. 7383 _____ Медведев И.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2019

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.

Ход работы

В ходе работы были написаны некоторые функции и структуры данных, которые описаны в табл. 1-2.

Таблица 1 – Используемые функции

Название функции	Выполняемая задача
TETR_TO_HEX	Перевод числа из 2-ой в 16-ричную систему счисления.
BYTE_TO_HEX	Осуществляет перевод байта, помещенного в AL, в два символа в шестнадцатеричной системе счисления, помещая результат в AX.
WRD_TO_HEX	Переводит числа размером 2 байта в 16-ричную систему счисления. В AX – число, в DI – адрес последнего символа.
BYTE_TO_DEC	Переводит в 10-ную систему счисления, SI – адрес поля младшей цифры.
WRITE	Вывод сообщения на экран.

Окончание таблицы 1

GET_MEM	Получение и вывод строки с адресом недоступной памяти
GET_ENV	Получение и вывод строки с сегментным адресом среды, передаваемого программе
GET_TAIL	Получение и вывод строки с хвостом командной строки
GET_ENV_CONTENT	Получение и вывод строки с содержимым области среды
GET_PATH	Получение и вывод пути загружаемого модуля

Таблица 2 – Структуры данных

Название	Тип	Назначение
MEM	db	Строка «Memory address: h$»
ENV_ADRESS		Строка «Environment address: h$»
TAIL		Строка «Tail: $»
ENV		Строка «Environment contains: »
ENDL		Строка «$»
PATH		Строка «Path: $»

В ходе работы была написана программа, которая выводит на экран сегментный адрес первого байта недоступной памяти, сегментный адрес среды, передаваемой программе, хвост командной строки, содержимое

области среды в символьном виде и путь загружаемого модуля. Результаты работы представлены на рис. 1-2.

Рисунок 1 – Запуск .COM файла без хвоста.

Рисунок 2 – Запуск .COM файла с хвостом.

Вывод

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

Ответы на контрольные вопросы

Сегментный адрес недоступной памяти.

1) На какую область памяти указывает адрес недоступной памяти?

На границу оперативной памяти и области, доступной для загрузки программ.

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

Адрес указывает на байт, следующий за последним байтом выделенной памяти. Т.е. сразу после памяти, выделенной программе.

3) Можно ли в эту область памяти писать?

Да, т.к. в DOS нет защиты памяти.

Среда, передаваемая программе

1) Что такое среда?

Среда представляет собой последовательность строк имя – параметр, 00h. Имя и параметр являются последовательностью символов. Нулевой байт (00h) является концом данной строки.

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

Среда создается при загрузке DOS.

3) Откуда берется информация, записываемая в среду?

При работе с MS DOS, информация берется из файла autoexec.bat.

ПРИЛОЖЕНИЕ А. КОД ПРОГРАММЫ

lab1_c.asm

```
TESTPC SEGMENT
            ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
            ORG 100H
START:      JMP   BEGIN
;Данные
VersPC      db    'Version PC: $'
VersMSDOS   db    'Version MS-DOS:  .  ',0DH,0AH,'$'
NumOEM      db    'Number OEM:   ',0DH,0AH,'$'
SerNum      db    'Serial Number:      ',0DH,0AH,'$'

PC          db 'PC',0DH,0AH,'$'
PCXT        db 'PC/XT',0DH,0AH,'$'
AT                db 'AT',0DH,0AH,'$'
PS2_30            db 'PS2 model 30',0DH,0AH,'$'
PS2_50_60   db 'PS2 model 50/60',0DH,0AH,'$'
PS2_80            db 'PS2 model 80',0DH,0AH,'$'
PCjr        db 'PCjr',0DH,0AH,'$'
PC_Convert  db 'PC Convertible',0DH,0AH,'$'



Output            PROC near
            mov   ah,09h
            int   21h
            ret
Output            ENDP



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 шестн. числа в AX
            push  cx
            mov   al,ah
            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
            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
            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



PC_VERSION  PROC near

            mov   bx,0F000h
            mov   es,bx
            mov   al,es:[0FFFEh]
            mov   dx,offset VersPC
            call Output
            cmp al,0FFh
            je PC_lab

            cmp al,0FEh
            je PCXT_lab

            cmp al,0FBh
```

```
je PCXT_lab

cmp al,0FCh
je AT_lab

cmp al,0FAh
je PS2_30_lab

cmp al,0F8h
je PS2_50_60_lab

cmp al,0FCh
je PS2_80_lab

cmp al,0FDh
je PCjr_lab

cmp al,0F9h
je PCConvert_lab

PC_lab:
      mov dx, offset PC
      jmp end_lab
PCXT_lab:
      mov dx, offset PCXT
      call Output
      jmp end_lab
AT_lab:
      mov dx, offset AT
      jmp end_lab
PS2_30_lab:
      mov dx, offset PS2_30
      jmp end_lab
PS2_50_60_lab:
      mov dx, offset PS2_50_60
      jmp end_lab

PS2_80_lab:
      mov dx, offset PS2_80
      jmp end_lab
PCjr_lab:
      mov dx, offset PCjr
      jmp end_lab
PCConvert_lab:
      mov dx, offset PC_Convert
      jmp end_lab

end_lab:
      call Output
      mov   ah,30h
      int   21h
      ret
```

```
PC_VERSION  ENDP


VERSION_MS_DOS          PROC near

            mov   si,offset VersMSDOS
            add   si,16
            call  BYTE_TO_DEC
            add   si,3
            mov   al,ah
            call        BYTE_TO_DEC

            ret
VERSION_MS_DOS    ENDP


NUMBER_OEM              PROC near

            mov   al,bh
            lea   si,NumOEM
            add   si,13
            call  BYTE_TO_DEC

            ret
NUMBER_OEM              ENDP

SERIAL_NUMBER           PROC near

            mov   al,bl
            call  BYTE_TO_HEX
            lea   di,SerNum
            add   di,14
            mov   [di],ax
            mov   ax,cx
            lea   di,SerNum
            add   di,19
            call  WRD_TO_HEX

            ret
SERIAL_NUMBER           ENDP


BEGIN:
            call  PC_VERSION
            call  VERSION_MS_DOS
            call  NUMBER_OEM
            call  SERIAL_NUMBER

            lea   dx,VersMSDOS
            call  Output
            lea   dx,NumOEM
            call  Output
            lea   dx,SerNum
            call  Output
```

```
            xor   al,al
            mov   ah,4ch
            int   21h

TESTPC      ENDS
            END   START
```

lab1_e.asm

```
EOL EQU '$'
AStack SEGMENT STACK
            DW 512 DUP(?)
AStack ENDS

DATA  SEGMENT
      VersPC      db    'Version PC: $'
      VersMSDOS   db    'Version MS-DOS:  .  ',0DH,0AH,'$'
      NumOEM      db    'Number OEM:   ',0DH,0AH,'$'
      SerNum      db    'Serial Number:       ',0DH,0AH,'$'

      PC                db 'PC',0DH,0AH,'$'
      PCXT        db 'PC/XT',0DH,0AH,'$'
      AT                db 'AT',0DH,0AH,'$'
      PS2_30            db 'PS2 model 30',0DH,0AH,'$'
      PS2_50_60   db 'PS2 model 50/60',0DH,0AH,'$'
      PS2_80            db 'PS2 model 80',0DH,0AH,'$'
      PCjr        db 'PCjr',0DH,0AH,'$'
      PC_Convert  db 'PC Convertible',0DH,0AH,'$'
DATA  ENDS


CODE  SEGMENT
      ASSUME CS:CODE,   DS:DATA, SS:AStack
Output            PROC FAR
            mov   ah,09h
            int   21h
            ret
Output            ENDP

TETR_TO_HEX       PROC  FAR
            and   al,0fh
            cmp   al,09
            jbe   NEXT
            add   al,07
NEXT:       add   al,30h
            ret
TETR_TO_HEX       ENDP

BYTE_TO_HEX       PROC FAR
            push  cx
            mov   al,ah
            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  FAR
            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  FAR
            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

PC_VERSION        PROC FAR

            mov   bx,0F000h
            mov   es,bx
            mov   al,es:[0FFFEh]
            mov   dx,offset VersPC
            call Output
```

```
            cmp al,0FFh
            je PC_lab
            cmp al,0FEh
            je PCXT_lab
            cmp al,0FBh
            je PCXT_lab
            cmp al,0FCh
            je AT_lab
            cmp al,0FAh
            je PS2_30_lab
            cmp al,0F8h
            je PS2_80_lab
            cmp al,0FDh
            je PCjr_lab
            cmp al,0F9h
            je PCConvert_lab
            PC_lab:
                  mov dx, offset PC
                  jmp end_lab
            PCXT_lab:
                  mov dx, offset PCXT
                  jmp end_lab
            AT_lab:
                  mov dx, offset AT
                  jmp end_lab
            PS2_30_lab:
                  mov dx, offset PS2_30
                  jmp end_lab
            PS2_80_lab:
                  mov dx, offset PS2_80
                  jmp end_lab
            PCjr_lab:
                  mov dx, offset PCjr
                  jmp end_lab
            PCConvert_lab:
                  mov dx, offset PC_Convert
                  jmp end_lab

            end_lab:
                  call Output
                  mov   ah,30h
                  int   21h
                  ret


PC_VERSION  ENDP


VERSION_MS_DOS    PROC NEAR

            lea   si,VersMSDOS
            add   si,16
            call  BYTE_TO_DEC
            add   si,3
            mov   al,ah
```

```
            call        BYTE_TO_DEC

            ret
VERSION_MS_DOS    ENDP


NUMBER_OEM              PROC FAR

            mov   al,bh
            lea   si,NumOEM
            add   si,13
            call  BYTE_TO_DEC

            ret
NUMBER_OEM  ENDP


SERIAL_NUMBER     PROC FAR
            mov   al,bl
            call  BYTE_TO_HEX
            lea   di,SerNum
            add   di,14
            mov   [di],ax
            mov   ax,cx
            lea   di,SerNum
            add   di,19
            call  WRD_TO_HEX

            ret
SERIAL_NUMBER           ENDP



Main              PROC FAR

            mov   ax, DATA
            mov   ds,ax
            call  PC_VERSION
            call  VERSION_MS_DOS
            call  NUMBER_OEM
            call  SERIAL_NUMBER

            lea   dx,VersMSDOS
            call  Output
            lea   dx,NumOEM
            call  Output
            lea   dx,SerNum
            call  Output

            mov   ah,4ch
            int   21h
            ret
Main              ENDP
CODE              ENDS
                  END Main
```