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

ОТЧЕТ

по практической работе № 2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студент гр. 8383 _____ Степанов В.Д.

Преподаватель _____ Губкин А.Ф.

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.

Выполнение работы.

Был написан и отложен 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 работы программы приведен на рисунке 1. Код программы приведен приложении А.

Сегментный адрес недоступной памяти.

1) На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на конец основной оперативной памяти, которая располагается с сегментного адреса 0000h до 9FFFh.

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

Исходя из рисунка 2 можно сделать вывод, что адрес располагается в конце программной памяти.

3) Можно ли в это область памяти писать?

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

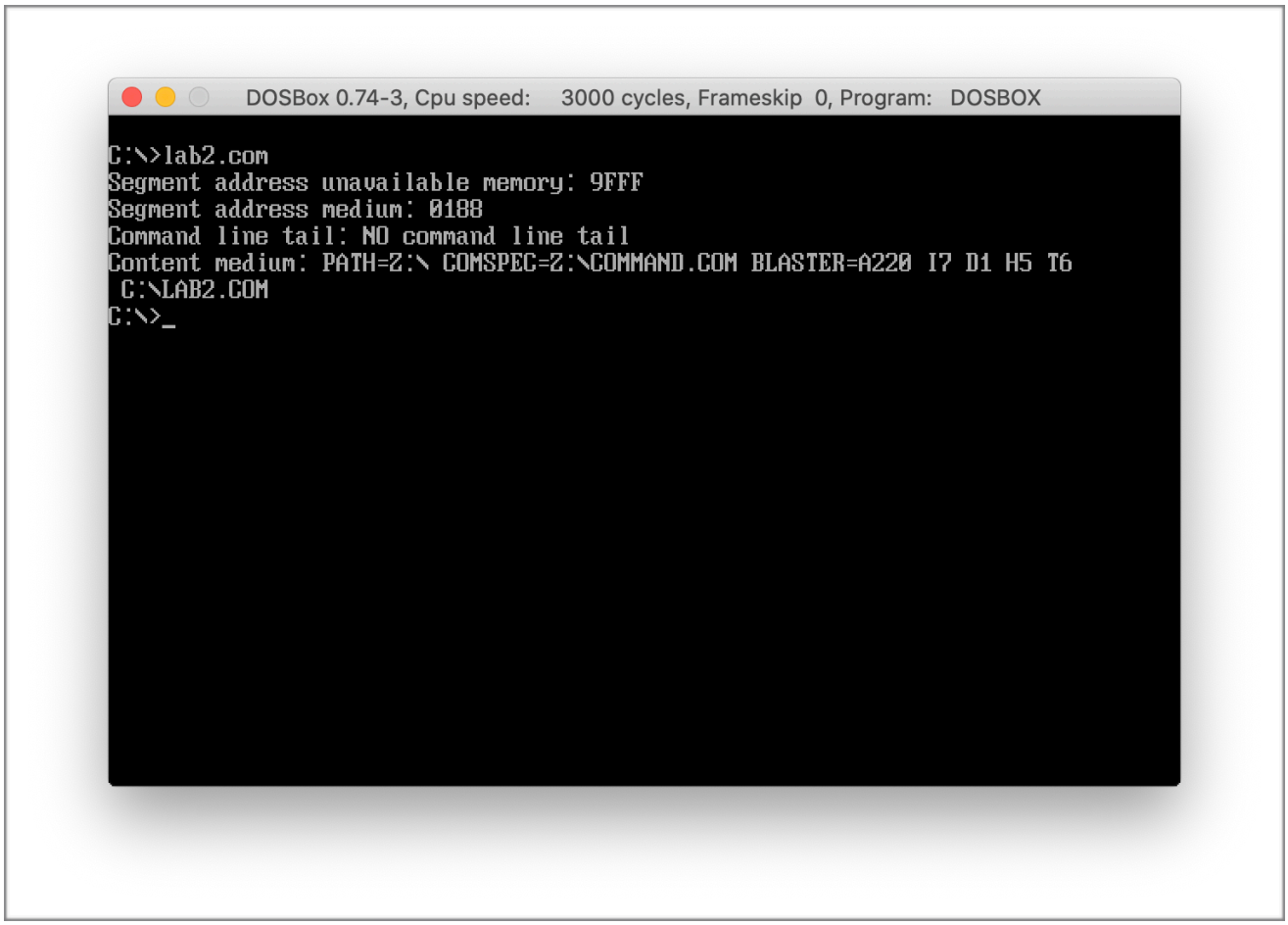

Рисунок 1 – Вывод работы программы

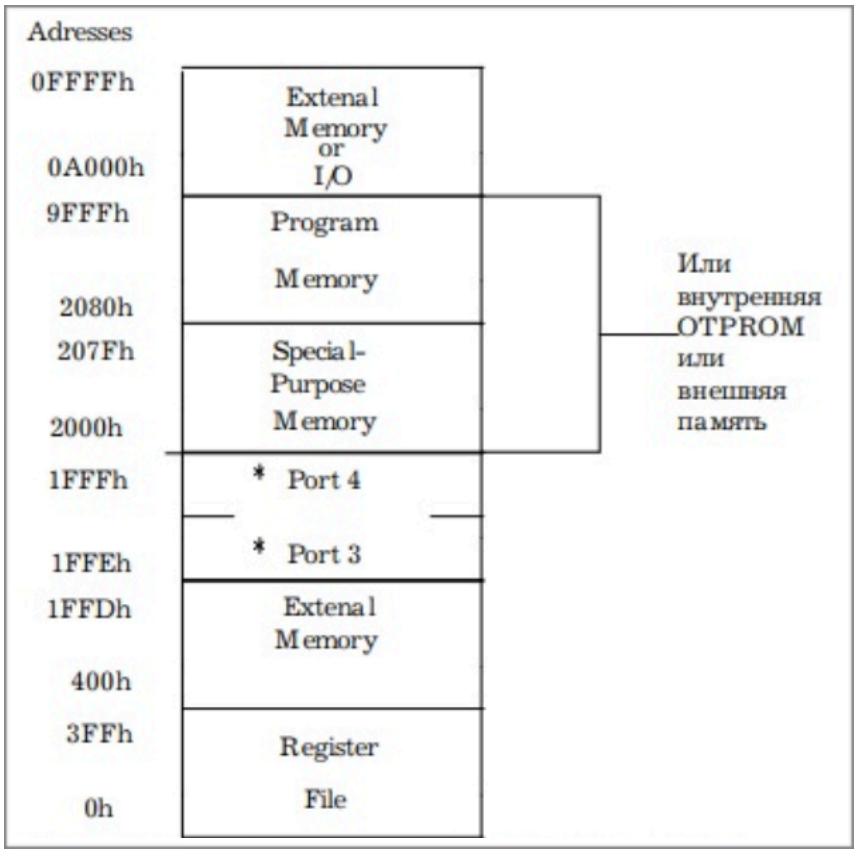

Рисунок 2 – Карта адресов памяти

Среда передаваемая программе.

1) Что такое среда?

Среда представляет собой текстовый массив, состоящий из строк вида:
“переменная=значение”,0

Переменная и значение – любые текстовые величины, байт 0 завершает каждую строку. Общая длина строк не превышает 32 Кбайта. Среда, передаваемая задаче от COMMAND, содержит параметр COMPSEG=“имя фала, содержащего используемый COMAND.COM”. Она так же содержит значения, установленные командами PATH, PROMPT, SET.

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

Когда одна программа открывает другую, создаётся порожденный процесс, которы получает собственный экземпляр блока среды (по умолчанию создается копия среды родителя). При запуске DOS создается среда COMMAND.COM.

3) Откуда берется информация, записываемая в среду?

Из родительской среды.

Выводы.

В ходе лабораторной работы были исследованы интерфейс управляющей программы и загрузочных модулей, префикса сегмента программы (PSP) и среды, передаваемой программе.

ПРИЛОЖЕНИЕ А
КОД ДЛЯ .COM МОДУЛЯ

```
LAB1  SEGMENT
      ASSUME  CS:LAB1, DS:LAB1, ES:NOTHING, SS:NOTHING
      ORG 100H
 START: JMP BEGIN

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

SEG_ADR_MEM db "Segment address unavailable memory: $"
SEG_ADR_MED db 13, 10, "Segment address medium: $"
TAIL_COM db 13, 10, "Command line tail: $"
NO_TAIL db "NO command line tail$"
CONTENT_MED DB 13, 10, "Content medium: $"
EMPTY_OUT DB 13, 10, "$"
OUT_1 db  "  $"

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

PRINT PROC near

 push ax
 sub ax, ax
 mov ah, 9h
 int 21h
 pop ax

 ret
PRINT 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

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

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

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

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

PRINT_NUMBER_DEC PROC
 push AX
 push BX

 sub BX, BX
 mov Bl, 10
 mov AH, 0
 div Bl
 mov DX, AX

 add DL, '0'
 sub AX, AX
 mov AH, 02h
 int 21h

 mov DL, DH

 add DL, '0'
 sub AX, AX
 mov AH, 02h
 int 21h

 pop BX
 pop AX
 ret
PRINT_NUMBER_DEC ENDP
;-------------------------------------------
```

```
WRITE_TAIL PROC

push ax
push bx
push dx
push si

 mov dx, offset TAIL_COM
 call PRINT

 mov bl, es:[0080h]
 cmp bl, 0
 jne PRINT_TAIL
 mov dx, offset NO_TAIL
 call PRINT
 jmp END_PROC
PRINT_TAIL:

 xor SI, SI
 xor AX, AX
 CYCLE:
 mov dl, es:[81h + SI]
 mov ah, 02h
 int 21h
 inc si
 loop CYCLE

END_PROC:

pop si
pop dx
pop bx
pop ax

ret
WRITE_TAIL ENDP

;-----------------------------------------
WRITE_CONTENT_MED prOC
push ax
push bx
push dx
push si
```

```
 mov dx, offset CONTENT_MED
 call PRINT
 xor si, si
 mov bx, 2Ch
 mov es, [bx]


READ:
 cmp word ptr es:[si], 0h
 je EMPTY

 mov dl, es:[si]
 mov ah, 02h
 int 21h

 jmp CHECK

EMPTY:
 mov dx, offset EMPTY_OUT
 CALL PRINT

CHECK:
 inc si
 cmp word ptr es:[si], 0001h
 je  NEXT_1
 jmp READ


NEXT_1:

 add si, 2
LOOP_1:
 cmp byte ptr es:[si], 00h
 je END_P

 mov dl, es:[si]
 mov ah, 02h
 int 21h
 inc si
 jmp LOOP_1

END_P:
```

```
pop si
pop dx
pop bx
pop ax

 ret
WRITE_CONTENT_MED ENDP


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

 BEGIN:



 mov dx, offset  SEG_ADR_MEM
 call PRINT

 mov bx, es:[0002h]
 mov al, bh
 mov di, offset OUT_1
 call BYTE_TO_HEX
 mov [di], ax
 mov dx, offset OUT_1
 CALL PRINT

 mov al, bl
 mov di, offset OUT_1
 call BYTE_TO_HEX
 mov [di], ax
 mov dx, offset OUT_1
 CALL PRINT

 mov dx, offset  SEG_ADR_MED
 call PRINT

 mov bx, es:[002Ch]
 mov al, bh
 mov di, offset OUT_1
 call BYTE_TO_HEX
 mov [di], ax
```

```
mov dx, offset OUT_1
CALL PRINT

mov al, bl
mov di, offset OUT_1
call BYTE_TO_HEX
mov [di], ax
mov dx, offset OUT_1
CALL PRINT

xor di, di
xor dx, dx

CALL WRITE_TAIL

call WRITE_CONTENT_MED



xor AX, AX
mov  AH, 4Ch
int 21h

LAB1 ENDS
END START
```