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

Студентка гр. 8382 _____ Рочева А.К.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Ход выполнения.

Чтобы вывести на экран сегментный адрес недоступной памяти в шестнадцатеричном виде, используем функцию *WRD_TO_HEX*. Предварительно в регистр *ax* поместим число по адресу *cs:[02h]* (т.к при загрузке *.com* модуля все сегментные регистры, включая *cs*, указывают на PSP), потому что именно оно и является нужным сегментным адресом (в PSP он хранится по смещению *2h*).

Точно так же выведем сегментный адрес среды, передаваемой программе (он находится по смещению *2Ch*).

Вывод хвоста командной строки происходит в метке *writeTail*. Сначала узнается кол-во символов в хвосте (это число находится в PSP по смещению *80h*) и затем выводятся символы по адресу *cs:[81h + bx]*, где *bx* – текущий счетчик символов. Вывод останавливается, когда *bx* становится равным кол-ву символов в хвосте.

Содержимое области среды – это последовательность строк, начинается с адреса *cs:[2Ch]*. Каждая строка завершается байтом нулей. Вывод строк происходит, пока не наступит конец среды (среда так же, как и строки, оканчивается байтом нулей, так что после окончания каждой строки идет проверка следующего за ней байта на равенство с нулем).

Путь загружаемого модуля идет после окончания среды и двух байтов, содержащих 00h и 01h, его вывод происходит так же, как и вывод содержимого области среды, пока не встретится байт 00h.

Результат выполнения lab2.com представлен на рисунке 1.

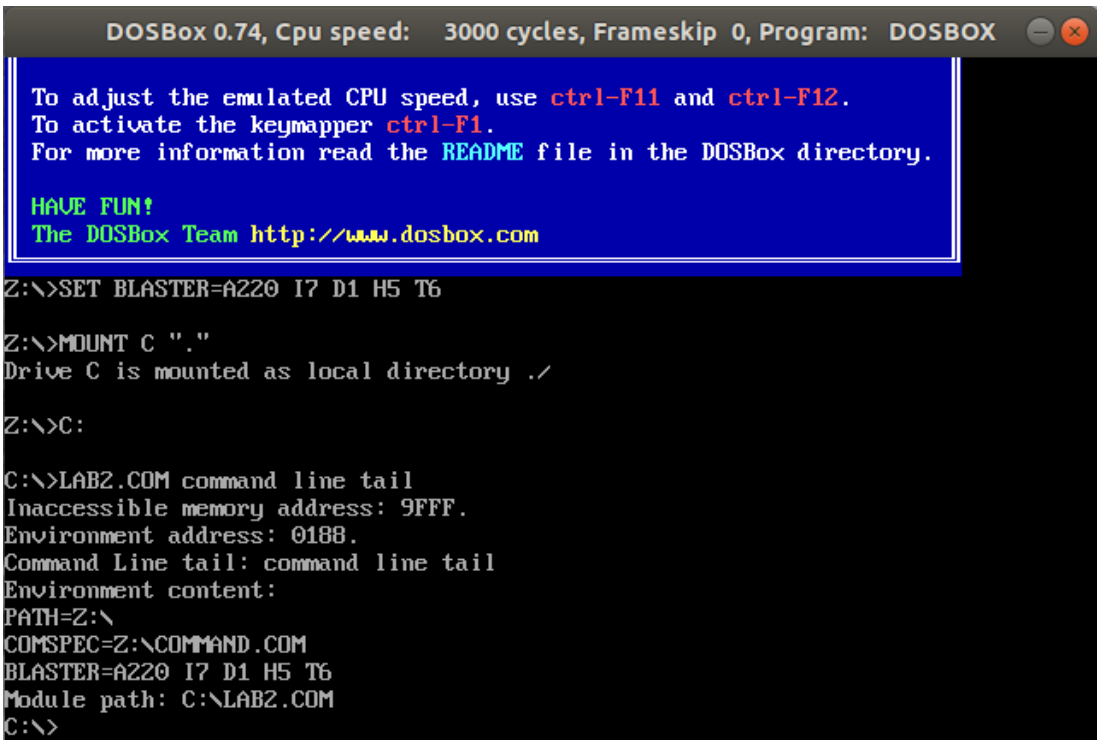

рис. 1 – результат выполнения lab2.com

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

   - На нижнюю границу доступной памяти в системе в параграфах, т.е уже начиная с адреса 9FFFh до адреса FFFFh память становится недоступной для загрузки пользовательских программ в нее.

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

   - Сразу за выделенной для программ памятью.

3. Можно ли в эту область памяти писать?

   - Можно, т.к. DOS не контролирует обращение программ к памяти.

Среда передаваемая программе

1. Что такое среда?

   - это массив строк, которым следует DOS при выполнении программы. Они содержат полезную информацию (к примеру, переменная PATH определяет путь к каталогу, в котором система ищет исполняемый

файл). Элементы строки среды такие же, как и те что можно обнаружить в диско- вом файле CONFIG.SYS.

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

   - Когда COMMAND.COM (дисковый файл, главная функция которого заключается в обработке команд, вводимых пользователем) запускает программу пользователя или одна программа запускает другую, создается порожденный процесс, который получает собственный экземпляр блока среды, при этом по умолчанию создается точная копия среды родителя, однако можно создать совершенно иную среду.

   Корневая среда создается в процессе начальной загрузки DOS.

3. Откуда берется информация, записываемая в среду?
4. При запуске системы автоматически создается файл AUTOEXEC.BAT, который записывается в корневой каталог загрузочного диска. В нем содержатся команды для настройки конфигурации системы. По информации, находящейся в этом файле, создается корневая среда.

Выводы

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

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД LAB2.ASM

```
LAB2            SEGMENT
                  ASSUME CS:LAB2, DS:LAB2, ES:NOTHING, SS:NOTHING
                  ORG 100H ; резервирование места для PSP
START:          JMP BEGIN

inaccessibleMemory db 'Inaccessible memory address:     .', 0DH,0AH, '$'
environmentAddress db 'Environment address:     .', 0DH,0AH, '$'
tail db 'Command Line tail:$'
endLine db 0DH,0AH, '$'
environmentContent db 'Environment content: ', 0DH,0AH, '$'
modulePath db 'Module path: $'


BEGIN:
                  ; сегментный адрес недоступной памяти
                  mov ax, cs:[02h]
                  mov di, offset inaccessibleMemory
                  add di, 32
                  call WRD_TO_HEX
                  mov [di], ax
                  mov dx, offset inaccessibleMemory
                  call WRITE

                  ; сегментный адрес среды
                  mov ax, cs:[2Ch]
                  mov di, offset environmentAddress
                  add di, 24
                  call WRD_TO_HEX
                  mov [di], ax
                  mov dx, offset environmentAddress
                  call WRITE

                  ; хвост командной строки
                  mov dx, offset tail
                  call WRITE
                  mov cl, cs:[80h]
                  mov bx, 0
writeTail:
                  cmp cl, 0
                  je endWrite
                  mov al, cs:[81h+bx]
                  push dx
                  push ax
```

```
                mov dx, ax
                mov ah, 02h
                int 21h
                pop ax
                pop dx
                inc bx
                dec cl
                jmp writeTail
endWrite:
                mov dx, offset endLine
                call WRITE

                ; содержимое области среды
                mov dx, offset environmentContent
                call WRITE
                mov es, cs:[2Ch]
                mov bx, 0
writeEnv:
                mov al, es:[bx]
                cmp al, 0h
                je checkEnd
                push dx
                push ax
                mov dx, ax
                mov ah, 02h
                int 21h
                pop ax
                pop dx
                inc bx
                jmp writeEnv
checkEnd:
                mov al, es:[bx+2]
                cmp al, 0h
                je endWriteEnv
                inc bx
                mov dx, offset endLine
                call WRITE
                jmp writeEnv
endWriteEnv:
                ; путь загружаемого файла

                mov dx, offset modulePath
                call WRITE
                add bx, 3
writePath:
                mov al, es:[bx]
                cmp al, 0h
                je endWritePath
```

```
                push dx
                push ax
                mov dx, ax
                mov ah, 02h
                int 21h
                pop ax
                pop dx
                inc bx
                jmp writePath
endWritePath:

                xor al, al
                mov ah, 4Ch
                int 21h


WRITE PROC near
                mov ah, 09
                int 21h
WRITE 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 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
   ret
BYTE_TO_HEX ENDP
;-------------------------------
WRD_TO_HEX PROC near
;перевод в 16 с/с 16-ти 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
   call BYTE_TO_HEX
   mov [DI],AH
   dec DI
   mov [DI],AL
   pop BX
   ret
WRD_TO_HEX ENDP

LAB2            ENDS
                  END START
```