

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. 8382

Нечепуренко Н.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

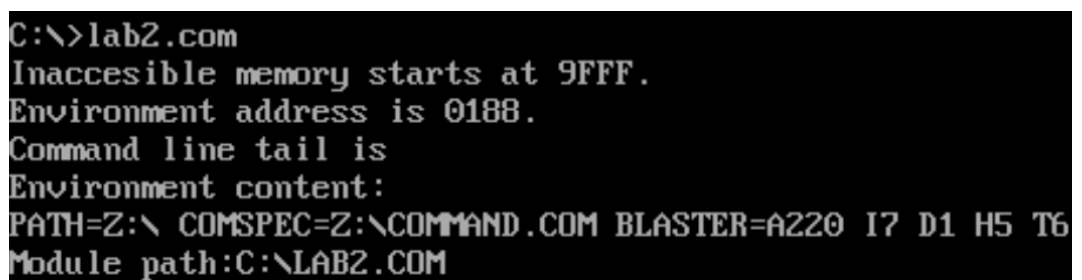
Исследование интерфейса управляющей программы и загрузочных модулей. Этот интерфейс состоит в передаче запускаемой программе управляющего блока, содержащего адреса и системные данные. Так загрузчик строит префикс сегмента программы (PSP) и помещает его адрес в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Выполнение работы.

Был написан программный модуль типа .com, который выводит на экран 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 программного модуля приведен на рисунке 1.



```
C:\>lab2.com
Inaccessible memory starts at 9FFF.
Environment address is 0188.
Command line tail is
Environment content:
PATH=Z:\ COMSPEC=Z:\COMMAND.COM BLASTER=A220 I7 D1 H5 T6
Module path:C:\LAB2.COM
```

Рисунок 1 – Результат работы программы

Исходный код модуля приведен в Приложении А.

Контрольные вопросы.

Сегментный адрес недоступной памяти.

1. На какую область памяти указывает адрес недоступной памяти?

Данный адрес указывает на начало области памяти, в которую нельзя загружать пользовательские программы. На рисунке 2 представлена часть схемы распределения адресного пространства.

Объем адресного пространства		Физические адреса	Сегментные адреса	
1 Кбайт	Векторы прерываний	00000h	0000h	}
256 байт	Область данных BIOS	00400h	0040h	
512 байт	Область данных DOS	00500h	0050h	
		00700h	0070h	
{ Более 70 Кбайт	Операционная система MS-DOS			} Обычная память (640 Кбайт)
	Загружаемые драйверы DOS			
	COMMAND.COM (резидентная часть)			
	Свободная память для загружаемых прикладных и системных программ (менее 570 Кбайт)			
64 Кбайт	Графическая видеопамять	A0000h	A000h	
32 Кбайт	Свободные адреса	B0000h	B000h	
		B8000h	B800h	

Рисунок 2 – Распределение адресного пространства

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

Он следует за блоком памяти выделенной программе (см. рис. 2).

3. Можно ли в эту область памяти писать?

Можно. Например, можно записать данные в видеобuffer и увидеть информацию на экране, если данные корректны.

Среда, передаваемая программе.

1. Что такое среда?

Среда окружения – совокупность значений системных переменных, путей, открытых файловых дескрипторов и других ресурсов операционной системы, передаваемые процессу (программе) при его запуске.

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

В процессе запуска DOS формируется корневая среда. Запускаемые программы получают копию среды родительского процесса во время загрузки программы в оперативную память.

3. Откуда берётся информация, записываемая в среду?

Корневая среда создается из файла `autoexec.bat` при запуске операционной системы.

Выводы.

В результате выполнения лабораторной работы был исследован префикс сегмента программы, переданный исполняемой программе операционной системой. Были получены адрес недоступной памяти и адрес среды, хвост командной строки, путь до файла и содержимое среды в символьном виде.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ.

```
codeseg segment
    assume cs:codeseg, ds:codeseg, es:nothing, ss:nothing
    org 100h
    start: jmp begin

endl ine db 13, 10, "$"
inaccessible_memory db "Inaccessible memory starts at      .", 13, 10, "$"
envir_address db "Environment address is      .", 13, 10, "$"
cmd_tail db "Command line tail is ", "$"
envir_content db "Environment content: ", 13, 10, '$'
module_path db "Module path:", "$"

begin:
inaccessible_memory_label:
    mov ax, cs:[2h]
    mov di, offset inaccessible_memory
    push di
    add di, 32
    call WRD_TO_HEX
    pop di
    call print

envir_address_label:
    mov ax, cs:[2ch]
    mov di, offset envir_address
    push di
    add di, 26
    call WRD_TO_HEX
    pop di
    call print

cmd_tail_label:
    mov di, offset cmd_tail
    call print

    xor cx, cx
    mov cl, cs:[80h]
    cmp cx, 0
    je cmd_tail_end
```

```

        mov si, 81h
        mov ah, 02h

cmd_tail_loop:
        mov dl, cs:[si]
        int 21h
        inc si
        loop cmd_tail_loop

cmd_tail_end:
        mov di, offset endlne
        call print

envir_content_label:
        mov di, offset envir_content
        call print
        mov si, 2Ch
        mov es, [si]
        mov si, 0
        mov ah, 02h

envir_content_outer_loop:
        mov dl, 0
        cmp dl, es:[si]
        je envir_content_end
envir_content_inner_loop:
        mov dl, es:[si]
        int 21h
        inc si
        cmp dl, 0
        jne envir_content_inner_loop
        jmp envir_content_outer_loop

envir_content_end:
        mov di, offset endlne
        call print

module_path_label:
        mov di, offset module_path
        call print

        add si, 3

```

```

module_path_loop:
    mov dl, es:[si]
    int 21h
    inc si
    cmp dl, 0
    jne module_path_loop

module_path_end:
    mov di, offset endlne
    call print

final:
    mov ax, 4c00h
    int 21h

print proc near
    ; prints di content
    push dx
    push ax
    mov ah, 9h
    mov dx, di
    int 21h
    pop ax
    pop dx
    ret
print 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
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

codeseg ends
end start

```