

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по практической работе № 2
по дисциплине «Операционные системы»
Тема: Исследование интерфейсов программных модулей

Студентка гр.

Ефимова М.А

Преподаватель

Губкин А. Ф.

Санкт-Петербург

2020

Цель работы.

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

Необходимые сведения для составления программы.

При начальной загрузке программы формируется psp, который размещается в начале первого сегмента программы. Он занимает 256 байт и располагается с адреса, кратного границе сегмента. При загрузке модулей типа .com все сегментные регистры указывают на адрес нашего psp. При загрузке модуля типа .exe сегментные регистры ds, es указывают на psp, поэтому следует их переопределять.

Формат psp:

0 – int 20h

2 – сегментный адрес первого байта недоступной памяти.

4 – зарезервировано

0Ah(10) – вектор прерывания 22h

0Eh(14) – вектор прерывания 23h

12h(18) – вектор прерывания 24h

2Ch(44) – сегментный адрес среды, перед. программе

5Ch – область форматир. Как стандартный неоткрытый блок управления файлом

6Ch – область форматир. Как стандартный неоткрытый блок управления файлом; перекрывается, если 5Ch открыт

80h – число сегментов в хвосте командной строки

81h – хвост командной строки – последовательность символов после имени вызыв. модуля.

Ход работы.

Получен модуль **.COM**, печатающий следующую информацию:

- 1)Сегментный адрес среды
- 2)Хвост командной строки
- 3)Содержимое области среды
- 4)Путь загружаемого модуля
- 5)Сегментный адрес недоступной памяти

Выполнение работы.

В первой строке указывается имя TESTPC, которая определяет командный процессор и путь к COMMAND.COM.

- TESTPC SEGMENT
- ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

Были объявлены строки для вывода информации:

- MEM_ADRESS db 'Locked memory address: h',13,10,'\$'
- ENV_ADRESS db 'Environment address: h',13,10,'\$'
- TAIL db 'Command line tail: ',13,10,'\$'
- NULL_TAIL db 'In Command tail no sybmols',13,10,'\$'
- CONTENT db 'Content:',13,10, '\$'
- END_STRING db 13, 10, '\$'
- PATH db 'Patch: ',13,10,'\$'

Функции для считывания данных из префикса и преобразования чисел (см. табл.1)

Функции.

Таблица 1 – функции в программе

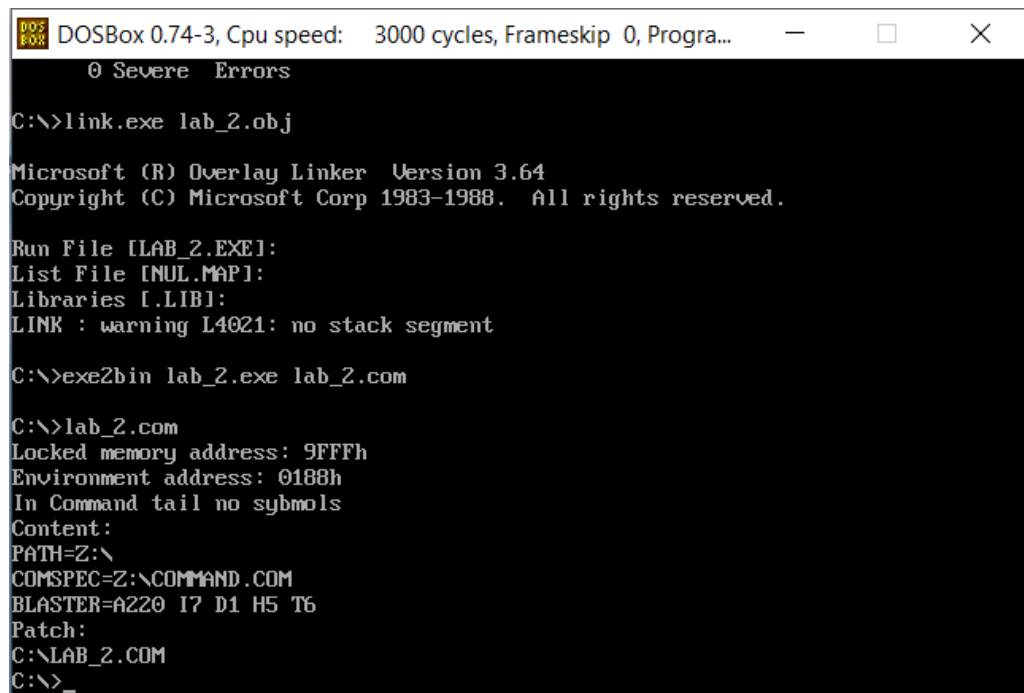
Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в

	код символа
BYTE_TO_HEX	Перевод байта в 16-ной с/с в символьный код
WRD_TO_HEX	Перевод слова в 16-ной с/с в символьный код
BYTE_TO_DEC	Перевод байта в 16-ной с/с в символьный код в 10-ной с/с
WRITESTRING	Вывод строки на экран
PSP_MEMORY	Получение адреса недоступной памяти
PSP_ENVIROMENT	Получение адреса среды
PSP_TAIL	Получение хвоста командной строки
PSP_CONTENT	Получения содержимого области среды и пути загружаемого файла

Среда заканчивается также байтом нулей:

- 1) Два нулевых байта являются признаком конца переменных среды.
- 2) Затем идут два байта, содержащих 00h,01h, после которых располагается маршрут загруженной программы.
- 3) Маршрут также заканчивается байтом 00h.

В результате выполнения были получены следующие значения(рис.1):



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
0 Severe Errors

C:\>link.exe lab_2.obj

Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

Run File [LAB_2.EXE]:
List File [NUL.MAP]:
Libraries [LIB]:
LINK : warning L4021: no stack segment

C:\>exe2bin lab_2.exe lab_2.com

C:\>lab_2.com
Locked memory address: 9FFFh
Environment address: 0188h
In Command tail no sybmols
Content:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
Patch:
C:\LAB_2.COM
C:\>_
```

Рисунок 1 – результат работы программы

Выводы.

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

ПРИЛОЖЕНИЕ А

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

Сегментный адрес недоступной памяти:

1. На какую область памяти указывает адрес недоступной памяти?

Указывает на память, в которой есть транзитная часть командного процессора COMMAND.COM. Транзитная часть командного процессора располагается в старших адресах памяти. Она содержит загрузчик программ в ОП и исполнитель внутренних команд.

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

Он расположен сразу же за областью памяти, которая выделена программе.

3. Можно ли в эту область памяти писать?

Можно, так как в DOS общее адресное пространство. После работы управление переходит резидентной части командного процессора, которая, считав с диска, может восстановить транзитную.

Среда, передаваемая программе:

1. Что такое среда?

Среда - это область в памяти, которую операционная система и другие программы используют как склад для самой разной информации.

Набор строк вида: (имя=значение). Переменная и значение – любые текстовые величины, байт 0 завершает каждую строку.

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

Среда создаётся при загрузке ОС и перед запуском приложения.

3. Откуда берётся информация, записываемая в среду?

Информация берется при загрузке. При загрузке выделяется область памяти для среды, содержимое которой копируется из среды родительского процесса.

Исходный код

Приложение LAB_2.ASM

TESTPC SEGMENT

ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING

ORG 100H

START: JMP BEGIN

; Данные

MEM_ADRESS db 'Locked memory address: h',13,10,'\$'

ENV_ADRESS db 'Environment address: h',13,10,'\$'

TAIL db 'Command line tail: ',13,10,'\$'

NULL_TAIL db 'In Command tail no sybmols',13,10,'\$'

CONTENT db 'Content:',13,10, '\$'

END_STRING db 13, 10, '\$'

PATH db 'Patch: ',13,10,'\$'

; Процедуры

;-----

;Перевод десятичной цифры в код символа

TETR_TO_HEX PROC near

and AL,0Fh ;AL->00001010, устанавли.текущ.режим

cmp AL,09 ;if 00001010<=00001001

jbe next ;короткий переход next

add AL,07 ;AL = 00010001

next:

add AL,30h ;преобразование числа в символ

ret

TETR_TO_HEX ENDP

;-----

;Перевод байта в 16-ной с/с в символьный код

BYTE_TO_HEX PROC near

;байт в AL переводится в два символа шест. числа в AX

```

push CX      ;заносим в стек сл.
mov AH,AL
call TETR_TO_HEX
xchg AL,AH      ;в AH младшая цифра
mov CL,4
shr AL,CL;в AL старшая цифра, побитовый сдвиг Al на 4
call TETR_TO_HEX
pop CX ;выталкиваем сл. из верш.стека в рег. CX
ret
BYTE_TO_HEX ENDP
;-----
;Перевод слова в 16-ной с/с в символьный код
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 ;BH = младшей цифре
call BYTE_TO_HEX
mov [DI],AH      ;младшую цифру
dec DI
mov [DI],AL      ;старшую цифру
dec DI      ;-1
mov AL,BH
call BYTE_TO_HEX
mov [DI],AH
dec DI
mov [DI],AL
pop BX
ret
WRD_TO_HEX ENDP
;-----
;Перевод байта в 16-ной с/с в символьный код в 10-ной с/с
BYTE_TO_DEC PROC near

```



```

; перевод в 10с/с
; SI - адрес поля младшей цифры
    push CX
    push DX
    xor AH,AH ;очистка части частного AH
    xor DX,DX ;очистка части частного DX
    mov CX,10
loop_bd:
    div CX
    or DL,30h ;преобразование остатка к ASCII коду.
    mov [SI],DL ;сохранение цифры суммы в памяти
    dec SI ;смещение следующего байта слагаемых
    xor DX,DX
    cmp AX,10
    jae loop_bd
    cmp AL,00h ;если символ не задан - переходим к записи в новый файл
    je end_l
    or AL,30h ;получение ASCII-кода цифры сумм
    mov [SI],AL ;сохранение цифры суммы в памяти
end_l:
    pop DX
    pop CX
    ret
BYTE_TO_DEC ENDP
;-----
WRITESTRING PROC near
    mov AH,09h ;номер функции 21-го прерывания, которая выводит строку на
экран
    int 21h
    ret
WRITESTRING ENDP
;-----
;Получение адреса недоступной памяти
PSP_MEMORY PROC near

```

```

;память
mov ax,ds:[02h]    ;DS:TESTPC
mov di, offset MEM_ADRESS    ;указываем смещ. на строку
add di, 26
call WRD_TO_HEX
mov dx, offset MEM_ADRESS    ;push,pop dx до этого
call WRITESTRING
ret
PSP_MEMORY ENDP

```

```

;-----
;Получение адреса среды
PSP_ENVIROMENT PROC near
    ;среда
    mov ax,ds:[2Ch]
    mov di, offset ENV_ADRESS
    add di, 24
    call WRD_TO_HEX    ;перевод в 16 с/с
    mov dx, offset ENV_ADRESS
    call WRITESTRING
    ret
PSP_ENVIROMENT ENDP

```

```

;-----
;Получение хвоста командной строки
PSP_TAIL PROC near
    ;оставшееся
    xor cx, cx
    mov cl, ds:[80h]    ;Загружаем в cl длину введенной командной строки.
    mov si, offset TAIL    ;в si смещение для строки TAIL
    add si, 19
    cmp cl, 0h    ;если пустая
    je empty_tail
    xor di, di    ;очистка части частного di

```

```

        xor ax, ax      ;очистка части частного ax
readtail:
        mov al, ds:[81h+di] ;AL = 10000001b = 81h,начиная с 81h идем по всем
СИМВОЛАМ
        inc di          ;чтобы дальше продолжалось
        mov [si], al
        inc si
        loop readtail   ;цикл
        mov dx, offset TAIL
        jmp end_tail
empty_tail:
        mov dx, offset NULL_TAIL ;в dx строку о пустом
end_tail:
        call WRITESTRING
        ret
PSP_TAIL ENDP
;-----
;Получения содержимого области среды и пути загружаемого файла
PSP_CONTENT PROC near
        ;ENVIROMENT CONTENT
        mov dx, offset CONTENT
        call WRITESTRING
        xor di,di
        mov ds, ds:[2Ch] ;psp
        ;читаем строку
read_string:
        cmp byte ptr [di], 00h ;сверяется значение байта в памяти
        jz end_str      ;if =
        mov dl, [di]
        mov ah, 02h     ; 02h выводит символ на экран
        int 21h
        jmp find_end
        ;конец строки
end_str:

```

cmp byte ptr [di+1],00h ; сравниваем переопределение типа данных
расположенных по адресу di+1

jz find_end

push ds

mov cx, cs

mov ds, cx ;ds = cs

mov dx, offset END_STRING ;в dx смещение для строки

call WRITESTRING

pop ds

;находим конец

find_end:

inc di

cmp word ptr [di], 0001h ;атрибут символа, если больше нет символов, то
переход

jz read_path

jmp read_string

read_path:

push ds

mov ax, cs

mov ds, ax ;ds = cs

mov dx, offset PATH ;в dx смещение для строки

call WRITESTRING

pop ds

add di, 2

loop_path:

cmp byte ptr [di], 00h ;не равно ли нулю

jz complete

mov dl, [di]

mov ah, 02h ;вывода на экран

int 21h

inc di

jmp loop_path

complete:

ret

PSP_CONTENT ENDP

; Код

BEGIN:

call PSP_MEMORY

call PSP_ENVIROMENT

call PSP_TAIL

call PSP_CONTENT

xor AL,AL

mov AH,4Ch ;прерывание 21h

int 21H

TESTPC ENDS

END START