

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

ЛАБОРАТОРНАЯ РАБОТА № 2
по дисциплине «Операционные системы»
Тема: Исследование структур загрузочных модулей.

Студентка гр.9381

Андрух И.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

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.

Задание:

Необходимо написать и отладить 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.

Ход работы.

1. Написали и отладили программный модуль типа .COM, который распечатывает и выводит информацию:

- Сегментный адрес недоступной памяти, взятый из PSP, в шестнадцатеричном виде.
- Сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.
- Хвост командной строки в символьном виде.

- Содержимое области среды в символьном виде.
- Путь загружаемого модуля.



DOSBox 0.74-3, Cpu speed: 3000 cycles, Fram..

```

C:\> exe2bin lb2.exe lb2.com

C:\>lb2.com
Inaccessible memory adress: 9FFF
Environment adress: 0188
Tail of command line:
Environment area contents:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Loadable path: C:\LB2.COM

C:\>lb2.com command
Inaccessible memory adress: 9FFF
Environment adress: 0188
Tail of command line:  command
Environment area contents:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6

Loadable path: C:\LB2.COM

C:\>

```

Сведения о функциях и структурах данных управляющей программы:

Переменные:

Название	Тип	Содержит текст
in_memory	db – определяет байт	Inaccessible memory adress \$
adress		Environment adress \$
tail		Tail of command line \$
content		Environment area contents: \$
path		Loadable path: \$
endl		\$

'\$'-осуществляет переход на другую строку.

Функции:

Название	Описание
TETR_TO_HEX	осуществляет перевод половины байта в символ.
BYTE_TO_HEX	осуществляет перевод байта, помещенного в al, в два символа в шестнадцатеричной системе счисления, помещая результат в ah.
WRD_TO_HEX	осуществляет перевод числового значения, помещенного в регистр AX, в символьную строку в шестнадцатеричной системе счисления, помещая результат в регистр di.

Ответы на контрольные вопросы:

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на значение сегментного адреса памяти, следующей за памятью, выделенной программе(первый байт после блока памяти, отведенного программе).

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

Адрес, указывает на байт, следующий за последним байтом памяти, выделенной, т.е. сразу после памяти, выделенной программе. Адрес 02h в блоке PSP, расположен в сторону увеличения адресов.

3. Можно ли в эту область памяти писать?

Можно, так как контроля доступа к памяти нет.

Среда, передаваемая программе

1. Что такое среда?

Фактически, среда представляет собой текстовый массив, состоящий из строк вида:

$$"<переменная>=<значение>", 0$$

Где *<переменная>* и *<значение>* - текстовые величины и нулевой байт, обозначающий конец строки. Таким образом, среда – совокупность системных переменных среды, данные в которых могут быть необходимы программе во время ее работы.

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

Когда одна программа запускает другую программу, то запущенная программа получает свой собственный экземпляр блока среды, который является точной копией среды родителя (однако, возможно создать совершенно другую среду). Запуск программ осуществляется командным интерпретатором (COMMAND.COM), который имеет свою среду, называемую корневой, а запуск программного интерпретатора происходит при загрузке ОС. Таким образом, среда создается перед запуском приложения, копирование всех переменных среды осуществляется для каждой запускаемой программы.

3. Откуда берется информация, записываемая в среду?

Из системного пакетного файла AUTOEXEC.BAT.

Вывод.

Был изучен интерфейс управляющей программы и загрузочных модулей. Был изучен PSP и среда, передаваемая программе.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

TESTPC SEGMENT

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

ORG 100H

START: JMP BEGIN

```
in_memory db 'Inaccessible memory adress:  $'
adress      db 'Environment adress:      $'
tail        db 'Tail of command line:  $'
content     db 'Environment area contents: $'
path         db 'Loadable path:  $'
endl        db 13, 10, '$'
```

;ПРОЦЕДУРЫ

;перевод десятичной цифры в код символа

;-----

```
TETR_TO_HEX PROC near
    and     al, 0fh ;логическое умножение всех пар битов
    cmp     al, 09
    jbe     NEXT ;Переход если ниже или равно
    add     al, 07
NEXT: add   al, 30h
    ret
TETR_TO_HEX ENDP
```

;перевод байта 16 с.с в символьный код

;байт в al переводится в 2 символа шестнадцетиричного числа в ah

;-----

BYTE_TO_HEX PROC near

push cx

mov ah, al

call TETR_TO_HEX

xchg al, ah ;обмен местами регистра/памяти и регистра

mov cl, 4

shr al, cl ;логический сдвиг вправо

call TETR_TO_HEX

pop cx

ret

BYTE_TO_HEX ENDP

;-----

;Перевод в 16 сс 16-ти разрядного числа

;ah - число, di - адрес последнего символа

WRD_TO_HEX PROC near

push bx

mov bh, ah

call BYTE_TO_HEX

mov [di], ah

dec di ;вычитает 1 из операнда

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

```

```

;-----
-----

```

BEGIN:

```

; сегментные адреса недоступной памяти

```

```

mov  ax, es:[0002h]
mov  di, offset in_memory+31
call WRD_TO_HEX
mov  dx, offset in_memory
mov  ah, 09h
int      21h
mov      dx, offset endl
mov      ah, 09h
int  21h

```

```

; сегментный адрес среды

```

```

mov  ax, es:[002Ch]
mov  di, offset adress+23
call WRD_TO_HEX
mov  dx, offset adress
mov  ah, 09h
int      21h
mov      dx, offset endl
mov      ah, 09h

```

```
int 21h
```

```
;хвост командной строки в символьном виде
```

```
mov dx, offset tail
```

```
mov ah, 09h
```

```
int 21h
```

```
xor cx, cx
```

```
xor bx, bx
```

```
mov cl, byte PTR es:[80h]
```

```
mov bx, 81h
```

```
cycle1:
```

```
cmp      cx, 0h
```

```
je       continue1
```

```
mov dl, byte PTR es:[bx]
```

```
mov      ah, 02h; вывод символа на экран
```

```
int      21h
```

```
inc      bx
```

```
dec      cx
```

```
jmp      cycle1
```

```
continue1:
```

```
mov      dx, offset endl
```

```
mov ah, 09h
```

```
int 21h
```

```
;содержимое области среды в символьном виде
```

```
push es
```

```
mov      dx, offset content
```

```
mov ah, 09h
```

```
int 21h
```

```
mov      dx, offset endl
```

```

mov  ah, 09h
int  21h
mov      bx, es:[002Ch]
mov      es, bx
xor  bx, bx

```

continue2:

```

mov  dl, byte PTR es:[bx]
cmp  dl, 0h
je   cycle2
mov  ah, 02h
int  21h
inc  bx
jmp  continue2

```

cycle2:

```

mov  dx, offset endl
mov  ah, 09h
int  21h
inc  bx
mov  dl, byte PTR es:[bx]
cmp  dl, 0h
je   quit2
jmp  continue2

```

quit2:

;Путь загружаемого модуля

```

add  bx, 3

```

```

        mov     dx, offset endl
        mov     ah, 09h
        int     21h

        mov     dx, offset path
        mov     ah, 09h
        int     21h


cycle3:
        mov     dl, byte PTR es:[bx]
        cmp     dl, 0h
        je      quit3
        mov     ah, 02h
        int     21h
        inc     bx
        jmp     cycle3

quit3:


        mov     dx, offset endl
        mov     ah, 09h
        int     21h


        ;выход в dos
        xor     al, al
        mov     ah, 4ch
        int     21h
        ret

TESTPC  ENDS

END     START

```