

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

Студент гр.0382

Литягин С.М.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

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 среды программы (PSP) и среды, передаваемой программе.

Задание.

1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .COM, который выбирает и распечатывает следующую 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.

Сохраните результаты, полученные программой, и включите их в отчет.

2. Оформление отчета в соответствии с требованиями. В отчет включите скриншот с запуском программы и результатами.

Выполнение работы:

1. За основу был взят шаблон .COM модуля из методического пособия, в котором реализованы процедуры преобразования двоичных кодов в символы шестнадцатеричных и десятичных чисел.

Для выполнения задания лабораторной работы были написаны следующие процедуры: PSAUM, PSAE, PCEA, PCLT.

Искомую информацию можно получить по адресу PSP:[смещение]. Поскольку сегментный регистр данных указывает на начало PSP, то в программе получаем информацию по адресу DS:[смещение].

Процедура PSAUM нужна для вывода сегмента адреса недоступной памяти. Для этого записываем значение по адресу DS:[2h] в регистр AX. Затем вызываем процедуру WRD_TO_HEX, что переводит шестнадцатеричные два байта в строковый формат, и выводим сообщение на консоль.

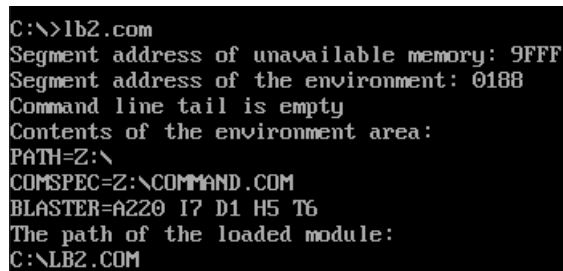
Процедура PSAE нужна для вывода сегментного адреса среды, передаваемого программе. Нужная нам информация лежит по адресу DS:[2Ch]. Записываем нужное значение в регистр AX. Вызываем процедуру WRD_TO_HEX, что переводит шестнадцатеричные два байта в строковый формат, и выводим сообщение на консоль.

Процедура PCLT нужна для вывода хвоста командной строки в символьном виде. Сначала проверяем содержимое адреса DS:[80h]. Если значение равно нулю, то в консоль выводится сообщения, что хвост командной строки пуст. Иначе, начиная с DS:[81h] выводим последовательность символов после имени вызываемого модуля.

Процедура PCEA нужна для вывода содержимого области среды в символьном виде и пути загружаемого модуля. Заносим в сегментный регистр ES значение, что находится по адресу DS:[2Ch] (как было ранее сказано, это сегментный адрес среды, передаваемый программе). Обнуляем регистр DI. Будем заносить значения по адресу ES:[DI] в регистр DL, инкрементируя DI. Поскольку область среды содержит последовательность символьных строк,

каждая из которых завершается байтом нулей, то будем выводить эти строки посимвольно, пока в регистре DL не будет значения 0h. Если после инкрементирования в DL будет значение 0h, то мы вывели все содержимое области среды. Увеличив DI на 3, получаем начало маршрута загруженной программы. Выводим его также посимвольно, пока в DL не окажется значение 0h.

Результаты работы программы представлены на рисунках 1 и 2.



```
C:\>lb2.com
Segment address of unavailable memory: 9FFF
Segment address of the environment: 0188
Command line tail is empty
Contents of the environment area:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
The path of the loaded module:
C:\LB2.COM
```

Рисунок 1 – запуск программы с пустым хвостом командной строки



```
C:\>lb2.com fffffff
Segment address of unavailable memory: 9FFF
Segment address of the environment: 0188
Command line tail: fffffff
Contents of the environment area:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
The path of the loaded module:
C:\LB2.COM
```

Рисунок 2 – запуск программы с непустым хвостом командной строки

Исходный программный код смотрите в приложении А.

Ответы на контрольные вопросы.

1. Сегментный адрес недоступной памяти:

- На какую область памяти указывает адрес недоступной памяти?

Он указывает на сегмент, расположенный сразу после выделенной программе памяти.

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

Первый байт памяти, расположенный после выделенной программе памяти. Найти этот адрес можно в PSP по смещению 2h.

- Можно ли в эту область памяти писать?

Можно, т.к. в DOS нет защиты памяти от перезаписи.

2. Среда, передаваемая программе:

- Что такое среда?

Среда – это область памяти, содержащая последовательность символьных строк вида “имя = параметр”, в которых записана информация переменных окружений. Есть несколько переменных окружений, например, PATH, в которое записан путь к каталогу, где система ищет исполняемый файл.

- Когда создается среда? Перед запуском приложения или в другое время?

Изначально среда создается при запуске ОС. Когда же запускается приложение, создается копия этой среды, в которую добавляются дополнительные параметры для данного приложения, если это требуется.

- Откуда берется информация, записываемая в среду?

При запуске DOS после старта командного интерпретатора COMMAND.COM выполняется системный пакетный файл AUTOEXEC.BAT, расположенный в корневом каталоге. Он и устанавливает ключевые переменные среды.

Выводы.

В ходе работы был исследован интерфейс управляющей программы и загрузочных модулей, а также исследован префикс среды программы (PSP) и среды, передаваемой программе. Была написана программа, выводящая в консоль информацию согласно заданию.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb2.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START:    jmp BEGIN

; ДАННЫЕ
SAUM db 'Segment address of unavailable memory:      ', 0DH, 0AH,
'$'
SAE db 'Segment address of the environment:      ', 0DH, 0AH, '$'
CLT db 'Command line tail:      ', '$'
ECLT db 'Command line tail is empty', 0DH, 0AH, '$'
CEA db 'Contents of the environment area:      ', 0DH, 0AH, '$'
PLM db 'The path of the loaded module:      ', 0DH, 0AH, '$'

; ПРОЦЕДУРЫ
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 16-го 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, в 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
;-----
PRINT PROC near
    push AX
    mov AH, 09h
    int 21h
    pop AX
    ret
PRINT ENDP
;-----
PRINT_SYM PROC near
    push AX
    mov AH, 02h
    int 21h
    pop AX
    ret
PRINT_SYM ENDP
;-----
PSAUM PROC near
    mov AX, DS:[2h]
    mov DI, offset SAUM + 42
    call WRD_TO_HEX
    mov DX, offset SAUM
    call PRINT
    ret
PSAUM ENDP
;-----
PSAE PROC near
    mov AX, DS:[2Ch]
    mov DI, offset SAE + 39
    call WRD_TO_HEX
    mov DX, offset SAE
    call PRINT
    ret
PSAE ENDP
;-----
PCEA PROC near
    mov DX, offset CEA
    call PRINT
    mov ES, DS:[2Ch]
    xor DI, DI
line:
    mov DL, ES:[DI]
    cmp DL, 0h
    je end_line
    call PRINT_SYM
    inc DI
    jmp line
end_line:
    mov DL, 0Dh
    call PRINT_SYM

```

```

        mov DL, 0Ah
        call PRINT_SYM
        inc DI
        mov DL, ES:[DI]
        cmp DL, 0h
        jne line

        mov DX, offset PLM
        call PRINT
        add DI, 3
path_line:
        mov DL, ES:[DI]
        cmp DL, 0h
        je end_path
        call PRINT_SYM
        inc DI
        jmp path_line
end_path:
        ret
PCEA ENDP
;-----
PCLT PROC near
        xor CX, CX
        mov CL, DS:[80h]
        cmp CL, 0h
        je empty
        mov DX, offset CLT
        call PRINT
        mov SI, 81h
loop_clt:
        mov DL, DS:[SI]
        call PRINT_SYM
        inc SI
        loop loop_clt
        mov DL, 0Dh
        call PRINT_SYM
        mov DL, 0Ah
        call PRINT_SYM
        ret
empty:
        mov DX, offset ECLT
        call PRINT
        ret
PCLT ENDP
;-----
; КОД
BEGIN:
        call PSAUM
        call PSAE
        call PCLT
        call PCEA
        xor AL, AL
        mov AH, 4Ch
        int 21h
TESTPC ENDS
END START

```