

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. 9382

Дерюгин Д.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 в сегментный регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Процедуры программы.

Процедура	Описание
TETR_TO_HEX	Перевод десятичной цифры в код символа, который записывается в AL
BYTE_TO_HEX	Перевод значений байта в число 16-ой СС и его представление в виде двух символов
WRD_TO_HEX	Перевод слова в число 16-ой СС и представление его в виде четырех символов
BYTE_TO_DEC	Перевод значения байта в число 10-ой СС и представляет его в виду символов
print	Вывод строки на экран
UNAVAILABLE_MEMORY_PROC	Печатает на экран сегментный адрес недоступной памяти
ADDRESS_OF_ENVIRONMENT_PROC	Печатает на экран сегментный адрес среды, передаваемой программы

TAIL_PROC	Печатает на экран хвоста командной строки
CONTENT_OF_ENVIRONMENT_PROC	Печатает на экран содержимое области среды
PRINT_CHARACTER_PROC	Вывод символа на экран
PATH_PROC	Печатает на экран путь загружаемого модуля

Задание.

Шаг 1. Для выполнения лабораторной работы 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. Сохраните результаты, полученные программой, и включите их в отчет.

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

Необходимые сведения для составления программы.

При начальной загрузке программы формируется PSP, который размещается в начале первого сегмента программы. PSP занимает 256 байт и располагается с адреса, кратного границе сегмента. При загрузке модулей типа .COM все сегментные регистры указывают на адрес PSP. Именно по этой причине значения этих регистров в модуле .EXE следует переопределять.

Формат PSP:

Смещение	Длина поля(байт)	Содержимое поле
0	2	int 20h
2	2	Сегментный адрес первого байта недоступной памяти. Программа не должна модифицировать содержимое памяти за этим адресом.
4	6	Зарезервировано
0Ah(10)	4	Вектор прерывания 22h (IP, CS)
0Eh(14)	4	Вектор прерывания 23h (IP, CS)
12h(18)	4	Вектор прерывания 24h (IP, CS)
2Ch(44)	2	Сегментный адрес среды, передаваемой программе.
5Ch		Область форматируется как стандартный неоткрытый блок управления файлом (FCB)
6Ch		Область форматируется как стандартный неоткрытый блок управления файлом (FCB). Перекрывается, если FCB с адреса 5Ch открыт.
80h	1	Число символов в хвосте командной строки.
81h		Хвост командной строки – последовательность символов после имени вызываемого модуля.

Область среды содержит последовательность символьных строк вида:

имя=параметр

Каждая строка завершается байтом нулей.

В первой строке указывается имя COMSPEC, которая определяет используемый командный процессор и путь к COMMAND.COM. Следующие строки содержат информацию, задаваемую командами PATH, PROMT, SET. Среда заканчивается также байтом нулей. Таким образом, два нулевых байта являются признаком конца переменных среды. Затем идут два байта, содержащих 00h, 01h, после которых располагается маршрут загруженной программы. Маршрут также заканчивается байтом 00h.

Результат работы программы

```
D:\>LAB2.COM
address of unavailable memory:9FFF
address of environment:0188
tail is empty
content of environment:
PATH=Z:\
COMSPEC=Z:\COMMAND.COM
BLASTER=A220 I7 D1 H5 T6
path: D:\LAB2.COM
```

Рис. 1 Результат работы программы

Ответы на контрольные вопросы

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на 9FFF.

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

После основной программы

3) Можно ли в эту область памяти писать?

Можно, но не рекомендуется делать это.

Среда передаваемая программе

1) Что такое среда?

Область в памяти, в которой хранятся значения переменных среды
«имя=параметр»

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

Во время загрузки модуля.

3) Откуда берется информация, записываемая в среду?

Из файла AUTOEXEC.BAT, который расположен в корневом каталоге загрузочного устройства и содержит информацию о переменных среды

Вывод.

В результате выполнения лабораторной работы был исследован интерфейс управляющей программы и загрузочных модулей.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: lab2.asm

```
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100h
START: JMP BEGIN
; ДАННЫЕ
    UNAVAILABLE_MEMORY      db  'address of unavailable memory:
',0DH,0AH,'$'
    ADDRESS_OF_ENVIRONMENT  db  'address of environment:
',0DH,0AH,'$'
    TAIL                    db  'tail:',0DH,0AH,'$'
    EMPTY_TAIL              db  'tail is empty',0DH,0AH,'$'
    CONTENT_OF_ENVIRONMENT  db  'content of environment:',0DH,0AH,'$'
    PATH                    db  'path: ','$'
    NEW_LINE                db  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 шестн. 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
;-----
BYTE_TO_DEC PROC near
; перевод в 10с/с, SI - адрес поля младшей цифры
    push CX
    push DX
    push SI
    xor AH,AH
    xor DX,DX
    mov CX,10

loop_bd:
    div CX
    or DL,30h
    mov [SI],DL
    dec SI
    xor DX,DX
    cmp AX,10
    jae loop_bd
    cmp AL,00h
    je end_l
    or AL,30h
    mov [SI],AL
end_l:

```



```

        pop SI
    pop DX
    pop CX

    ret
BYTE_TO_DEC ENDP

```

```

print proc near
    mov ah, 09h
    int 21h
    ret
print ENDP

```

```

UNAVAILABLE_MEMORY_PROC proc near
    mov ax, ds:[02h]
    mov di, offset UNAVAILABLE_MEMORY
    add di, 33
    call WRD_TO_HEX
    mov dx, offset UNAVAILABLE_MEMORY
    call print
    ret
UNAVAILABLE_MEMORY_PROC ENDP

```

```

ADDRESS_OF_ENVIRONMENT_PROC proc near
    mov ax, ds:[2Ch]
    mov di, offset ADDRESS_OF_ENVIRONMENT
    add di, 26
    call WRD_TO_HEX
    mov dx, offset ADDRESS_OF_ENVIRONMENT
    call print
    ret
ADDRESS_OF_ENVIRONMENT_PROC ENDP

```

```

TAIL_PROC proc near
    sub cx, cx
    mov cl, ds:[80h]
    cmp cl, 0
    je empty_tail_m
    sub ax, ax
    sub di, di
print_tail:
    mov al, ds:[81h + di]
    inc di

```

```

        call print
        loop print_tail

empty_tail_m:
        mov dx, offset EMPTY_TAIL
        call print

        ret
TAIL_PROC ENDP

CONTENT_OF_ENVIRONMENT_PROC proc near
        sub    si, si
        sub bx, bx
        mov    dx, offset CONTENT_OF_ENVIRONMENT
        call print
        mov    es, ds:[2ch]

cont:
        mov    bl, es:[si]
        cmp    bl, 0
        jne    next_character
        inc    si
        mov    bl, es:[si]
        mov    dx, offset NEW_LINE
        call print
next_character:
        call PRINT_CHARACTER_PROC
        mov    ax, es:[si]
        cmp    ax, 1
        jne    cont
        ret
CONTENT_OF_ENVIRONMENT_PROC ENDP

PATH_PROC proc near
        mov    dx, offset PATH
        call print
        add    si, 2
looping:
        mov    bl, es:[si]
        cmp    bl, 0
        je     exit
        call PRINT_CHARACTER_PROC
        jmp    looping

```

```

exit:
    ret
PATH_PROC ENDP

PRINT_CHARACTER_PROC proc near
    mov     dl, bl
    mov     ah, 02h
    int     21h
    inc     si
    ret
PRINT_CHARACTER_PROC ENDP

BEGIN:
    call    UNAVAILABLE_MEMORY_PROC
    call    ADDRESS_OF_ENVIRONMENT_PROC
    call    TAIL_PROC
    call    CONTENT_OF_ENVIRONMENT_PROC
    call    PATH_PROC

    xor     al, al
    mov     ah, 4Ch
    int     21h
TESTPC ENDS
END START

```