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.

Студентка гр. 0382 _____ Морева Е.С.

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 сегмента программы (PSP) и среды, передаваемой программе.

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. Оформить отчет в соответствии с требованиями. В отчет включить скриншот с запуском программы и результатами.

Теоретические сведения.

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. При загрузке модуля типа **.EXE** сегментные регистры DS и ES указывают на PSP. Именно по

этой причине значения этих регистров в модуле **.EXE** следует переопределять.

Формат PSP:

Смещение	**Длина поля(байт)**	**Содержимое поля**
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, PROMPT, 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.

Выполнение работы.

В ходе лабораторной работы были написаны следующие процедуры:

1. Вспомогательные процедуры, для осуществления вывода:

- PRINT
- ADR_MEM — определение адреса недоступной памяти, взятого из PSP, вшестнадцатеричном виде. Выводит значение, хранящееся по адресу DS:[2h].
- ADR_ENV — определение сегментного адреса среды, передаваемой программе, в шестнадцатеричном виде. Выводит значение, хранящееся по адресу DS:[2Ch].
- _TAIL — вывод хвоста командной строки в символьном виде. По информации по адресу DS:[80h] (длина возможного хвоста) проверяем наличие хвоста (сравниваем полученное значение с 0). Если хвост – пуст, выводим пустую строку. Иначе, при помощи команды loop, считываем все символы хвоста начиная с DS:[81h] попутно выводя их на экран один за другим.
- CONT_ENV — Вывод содержимого области среды в символьном виде и пути загружаемого модуля. Строки считываются и посимвольно и выводятся каждая с новой строки.
- _PATH — Получение пути загружаемого модуля. Пропускаем два байта, и считываем и выводим необходимую информацию аналогично предыдущему пункту.

Результаты запуска программы:

Рис. 1. Результат запуска lab2.com

Рис. 2. Результат запуска lab2.com с “хвостом”

Исходный код программы см. в приложении А

Контрольные вопросы.

Сегментный адрес недоступной памяти

1) На какую область памяти указывает адрес недоступной памяти?

- Ответ: Адрес недоступной памяти указывает на сегмент, расположенный следом за тем, что отведён программе.

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

- Ответ: Этот адрес расположен в PSP по смещению 2h — первый байт памяти, идущий за областью, выделенной под программу.

3) Можно ли в эту область памяти писать?

- Ответ: В эту область памяти можно писать, так как в MS DOS нет защиты от перезаписи.

Среда, передаваемая программе

1) Что такое среда?

- Ответ: Среда представляет собой область памяти, которая хранит последовательность символьных строк вида: «имя=параметр», которые содержат информацию о системе.

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

- Ответ: Изначально среда создаётся при запуске ОС. Когда запускается приложение — создается копия данной среды, в которую, если в этом есть необходимость, вносятся требуемые приложением дополнительные параметры.

3) Откуда берется информация, записываемая в среду?

- Ответ: Информация, записываемая в среду, берётся из системного пакетного файла AUTOEXEC.BAT, который расположен в корневом каталоге загрузочного устройства.

Выводы.

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

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab2.asm

```
TESTPC SEGMENT
   ASSUME CS:TESTPC, DS:TESTPC, ES:TESTPC, SS:TESTPC
   ORG 100H
START: JMP BEGIN

; Данные
adrMem db 'Inaccessable memory:      h ', 0DH, 0AH, '$'
adrEnv db 'Enviroment Address:     h', 0DH, 0AH, '$'
tail db 'Tail: ', '$'
contEnv db 'Enviroment contein:', 0DH, 0AH, '$'
path db 'Path:', 0DH, 0AH, '$'
EOL db 0DH, 0AH, '$'

; 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 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
;--------------------------------
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
   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 DX
   pop CX
   ret
BYTE_TO_DEC ENDP
;--------------------------------
PRINT PROC near
   push AX
   mov AH,09h
   int 21h
   pop AX
   ret
PRINT ENDP

ADR_MEM PROC near
    mov ax, ds:[2]
    mov di, offset adrMem
    add di, 24
    call WRD_TO_HEX
    mov dx, offset adrMem
    call PRINT
```

```
    ret
ADR_MEM ENDP

ADR_ENV PROC near
    mov ax, ds:[44]
    mov di, offset adrEnv
    add di, 23
    call WRD_TO_HEX
    mov dx, offset adrEnv
    call PRINT
    ret
ADR_ENV ENDP

_TAIL PROC near
    mov cl, ds:[80h]
    mov dx, offset tail
    call PRINT
    cmp cl, 0
    je out_1

    mov si, 81h

    m1:
    mov dl, ds:[si]
    mov ah, 02h
    int 21h
    inc si
    LOOP M1

    out_1:
    mov dx, offset EOL
    call PRINT
    ret

_TAIL ENDP

CONT_ENV PROC near
    mov es, ds:[44]
    MOV DX, offset contEnv
    call PRINT

    xor di, di

next_str:
    mov dl, es:[di]
    cmp dl, 0h
    je final_1
    mov ah, 02h
    int 21h
    inc di
    jmp next_str

final_1:
```

```
    mov dx, offset EOL
    call PRINT
    inc di
    mov dl, es:[di]
    cmp dl, 0h
    jne next_str
    ret
CONT_ENV ENDP

_PATH PROC near
    mov es, ds:[44]
    MOV DX, offset path
    call PRINT


    xor di, di

next_ind:
    mov dl, es:[di]
    cmp dl, 0h
    je final_2
    inc di
    jmp next_ind

final_2:
    inc di
    mov dl, es:[di]
    cmp dl, 0h
    jne next_ind

    add di, 3
next_simbol:
    mov dl, es:[di]
    cmp dl, 0h
    je final_3
    mov ah, 02h
    int 21h
    inc di
    jmp next_simbol

final_3:
    mov dx, offset EOL
    call PRINT


    ret
_PATH ENDP


; Код
BEGIN:
   call ADR_MEM
   call ADR_ENV
```

```
    call _TAIL
    call CONT_ENV
    call _PATH
    xor AL,AL
    mov AH,4Ch
    int 21H
TESTPC ENDS
END START
```