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

ЛАБОРАТОРНАЯ РАБОТА № 2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей.

Студент гр. 9381 Матвеев А. Н.

Преподаватель Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Исследование 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.

Сведения о функциях и структурах данных.

Программа использует процедуры, приведённые в таблице ниже.

Названия процедур	Назначение
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
WRITE	Вывод строки. (Функция 09h прерывания 21h)
GET_SEG_ADR	Печатает сегментный адрес недоступной памяти, взятый из PSP, в 16-ном виде

GET_ENVIR	Печатает сегментный адрес среды, передаваемой программе, в 16-ном виде
GET_TAIL_COM	Печатает хвост командной строки в символьном виде
GET_CONTENT_AND_PATH	Печатает содержимое области среды в символьном виде и путь загружаемого модуля

Последовательность действий программы.

1) Вызывается GET_SEG_ADR, которая печатает сегментный адрес недоступной памяти, взятый из PSP, в 16-ном виде.
2) Вызывается GET_ENVIR, которая печатает сегментный адрес среды, передаваемой программе, в 16-ном виде.
3) Вызывается GET_TAIL_COM, которая печатает хвост командной строки в символьном виде.
4) Вызывается GET_CONTENT_AND_PATH, которая печатает содержимое области среды в символьном виде и путь загружаемого модуля.
5) Программа завершается.

Выполнение работы.

Был написан исходный .com модуль lab2.asm. Затем при помощи компилятора masm.exe, компоновщика link.exe и утилиты exe2bin.exe был создан загрузочный модуль lab2.com. Процесс его получения: lab2.asm -> lab2.obj -> lab2.exe -> lab2.com. Далее программа была отлажена и протестирована. Тесты представлены ниже.

Тест 1. Без аргументов командной строки

Тест 2. С аргументами

Результаты исследования проблем.

Сегментный адрес недоступной памяти.

1. На какую область памяти указывает адрес недоступной памяти?

Он указывает на первый байт после памяти, отведённой программе.

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

Он занимает второй и третий байты, считая от начала PSP, причём второй байт - старший байт адреса, а третий - младший.

3. Можно ли в эту область памяти писать?

Можно, так как контроля доступа к памяти в DOS нет.

Среда, передаваемая программе

1. Что такое среда?

Среда - это массив символов, состоящий из последовательности символьных строк вида:

<имя> = <параметр>, 00h

(00h - нулевой байт, обозначающий конец строки). Конец среды - также байт нулей. Среда хранит такую служебную информацию, как данные об используемом командном процессоре, путь к модулю COMMAND.COM и. др.

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

Запуск программ в DOS осуществляется программой COMMAND.COM, (это командный интерпретатор), который имеет свою среду, запуск COMMAND.COM происходит при загрузке DOS, команды, запускаемые интерпретатором получают копию блока его среды. Отсюда вытекает, что изначально среда создается при загрузке DOS, но перед запуском программы она копируется (возможно, с некоторыми изменениями) со среды COMMAND.COM.

3. Откуда берется информация, записываемая в среду?

Она берётся из системного пакетного файла AUTOEXEC.BAT, который находится в корневой папке загрузочного устройства. В нём находятся ключевые переменные среды (наиболее известны PATH, PROMPT, COMSPEC), которые участвуют в создании среды в простейшем случае.

Заключение.

Исследована управляющая программа и загрузочные модули. Исследованы префикс сегмента программы (PSP) и среда, передаваемая программе.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

Файл lab2.asm:

```
MAIN SEGMENT
 ASSUME CS:MAIN, DS:MAIN, ES:NOTHING, SS:NOTHING
      ORG 100H
START: JMP BEGIN

;ДАННЫЕ
INAC_MEM db 'Segment address of inaccesible memory:     ', 0DH, 0AH,
'$'
ENVIR db 'Segment address of environment:     ', 0DH, 0AH, '$'
TAIL_COM db 'Tail of command string: ', '$'
ENVIR_AREA db 'Environment area content: ', 0DH, 0AH, '$'
MOD_PATH db 'Path of module: ', '$'
buffer db 256 dup('$')
other_info db 256 dup ('$')
path db 256 dup ('$')

;процедуры
TETR_TO_HEX PROC near ;представляет 4 младших бита al в виде цифры
16-ой с.сч. и представляет её в символьном виде
      and AL,0Fh
      cmp AL,09
      jbe NEXT
      add AL,07
NEXT:
      add AL,30h ; результат в al
      ret
TETR_TO_HEX ENDP
;--------------------------------

BYTE_TO_HEX PROC near
; байт в al переводится в 2 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 ; предполагает, что в al - младший байт числа, в
ah - старший
; 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 ; сначала обработка: число в al -> al - старшая
цифра, ah - младшая
      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

WRITE PROC NEAR
     push ax
     mov   AH, 9
    int   21h  ; Вызов функции DOS по прерыванию
     pop ax
    ret
WRITE ENDP


;-------------------------------
```

```
GET_SEG_ADR PROC NEAR ;сегментный адрес недоступной памяти
     mov ax, ds:[0002h]
     mov dx, offset INAC_MEM
     mov di, dx
     add di, 42
     call WRD_TO_HEX
     call WRITE
     ret
GET_SEG_ADR ENDP

GET_ENVIR PROC NEAR ;сегментный адрес среды
     mov ax, ds:[002Ch]
     mov dx, offset ENVIR
     mov di, dx
     add di, 35
     call WRD_TO_HEX
     call WRITE
     ret
     GET_ENVIR ENDP

GET_TAIL_COM PROC NEAR ;хвост командной строки
     ;TAIL_COM
     mov dx, offset TAIL_COM
     call WRITE

     xor cx, cx
     mov cl, ds:[0080h] ; храним в cx размер аргумента

     mov bx, offset buffer
     xor di, di ; индекс приёмника изначально нулевой

     cmp cl, 0 ; проверяем на пустоту аргументов командной строки
     je end_of_cycle

     mov si, 0081h   ; по умолчанию смещение указывается относительно
начала ds, поэтому si (источник) будет 81h
     tail:
          mov  al, [si]
```

```
            mov  [bx + di], al
            inc di
            inc si
            cmp di, cx ; если индекс равен максимальному количеству -
выход из цикла
            jb tail

      end_of_cycle:;установка конечных управляющих символов
      mov     byte ptr [bx + di], 0Dh
    mov     byte ptr [bx + di + 1], 0Ah

      mov dx, offset buffer
      call WRITE
      ret
GET_TAIL_COM ENDP

GET_CONTENT_AND_PATH PROC near
      mov dx, offset ENVIR_AREA
      call WRITE
      push ds
      mov ax, ds:[002Ch]; содержимое среды
      mov ds, ax
      xor si, si
      mov di, offset other_info
      content: ; цикл записи содержимого среды из psp в строку
            lodsb
            cmp al, 00h; встретили 0 - проверяем следующий элемент
            je is_end
            stosb
            jmp content
      is_end:
            lodsb
            cmp al, 00h ; если и второй 0, то конец среды
            je end_envir
            mov byte ptr es:[di], 0Ah
            inc di
            dec si
            jmp content
```

```
      end_envir:
            mov     byte ptr es:[di], 0Dh
            mov     byte ptr es:[di+1], 0Ah
            pop ds
            mov dx, offset other_info
            call WRITE
      get_path: ; получение маршрута
            mov dx, offset MOD_PATH
            call WRITE
            push ds
            mov ax, ds:[002Ch]
            mov ds, ax
            add si, 2 ; теперь si установлен на начало маршрута
            mov di, offset es:path
            path_loop:
                  lodsb
                  cmp al, 00h; встретили 0 -  достигли конца пути
                  je the_end
                  stosb
                  jmp path_loop
            the_end:
                  mov     byte ptr es:[di], 0Dh
                  mov     byte ptr es:[di+1], 0Ah
                  pop ds
                  mov dx, offset path
                  call WRITE
      ret
GET_CONTENT_AND_PATH ENDP

;код
BEGIN:
      call GET_SEG_ADR
      call GET_ENVIR
      call GET_TAIL_COM
      call GET_CONTENT_AND_PATH
;выход в ДОС
       xor AL,AL
       mov AH,4Ch
```

```
       int 21H
MAIN ENDS
         END START

; КОНЕЦ МОДУЛЯ, START - ТОЧКА ВХОДА
```