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

ОТЧЕТ

по практической работе № 2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студентка гр. 9383 _____ Лапина А.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 состоит префикс сегмента программы (PSP) и помещает его адрес в сегментные регистр. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Постановка задачи.

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 стоки в символьном виде.
4. Содержимое области среды в символьном виде.
5. Путь загружаемого модуля.

Выполнение работы.

Для выполнения данной работы были реализованы следующие функции:

F1 — получает адрес недоступной памяти

F2 - получает адрес среды

F3 - получает хвост командной строки

F4 - получает содержимое области среды и пути загружаемого файла

Были объявлены строки для вывода информации:

- MEMORY db 'Memory segment: ',0DH,0AH,'$' - сегментный адрес недоступной памяти

- MEDIA db 'Segment media address: ',0DH,0AH,'$' - сегментный адрес среды
- TAIL db 'Tail of command line: ',0DH,0AH,'$' - хвост коммандной строки
- EMPTY db 'Tail of command line: [EMPTY]',0DH,0AH,'$' - пустой хвост
- CONTENT db 'Environment scope content:',0DH,0AH, '$'
- END_STRING db 0DH,0AH, '$'
- PATH db 'Path: ',0DH,0AH, '$'

Для вывода сегментного адреса недоступной памяти в регистр AX заносится машинное слово, располагаемое в префиксе программного сегмента PSP со смещением 02h, затем число записывается в строку в шестнадцатеричном виде, после чего строка выводится на экран (функция F1). Аналогично выводится сегментный адрес среды, передаваемой программе, располагаемый со смещением 2Ch (функция F2).

Для вывода хвоста командной строки в регистр CL помещается число символов в хвосте (смещение 80h), если оно равно 0, то выводится сообщение [EMPTY], иначе хвост командной строки (функция F3). Для вывода содержимого области среды и пути загружаемого модуля была реализована функция F4.

Результаты работ программы представлены на рисунке 1

Рисунок 1 – результат работы программы

Ответы на следующие контрольные вопросы.

Сегментный адрес недоступной памяти:

1. На какую область памяти указывает адрес недоступной памяти?

Адрес недоступной памяти указывает на основную оперативную память. (На первый байт после памяти, выделенного программе)

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

Сегментный адрес недоступной памяти лежит в сегменте PSP со смещением 2Ch.

3. Можно ли в эту область памяти писать?

Программа может модифицировать содержимое этой области памяти, однако не стоит этого делать, так как эта область не отведена программе.

Среда, передаваемая программе:

1. Что такое среда?

Среда – это область памяти, содержащая последовательность символьных строк, представляющих собой набор пар «имя» и «параметр» и хранящих в себе заданную информацию, например, данные о настройках системы, путь к COMMAND.COM и т.п.

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

Среда создаётся при загрузке операционной системы. В MS-DOS переменные среды используются совместно запускаемыми процессами.

3. Откуда берется информация, записываемая в среду?

Из системного пакетного файла AUTOEXEC.BAT. (расположен в корневом каталоге загрузочного устройства).

Выводы.

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

ПРИЛОЖЕНИЕ А

Название файла: com.asm

```
TESTPC SEGMENT
   ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
   ORG 100H    ;так как адресация начинается со смещением 100 в .com
START: JMP BEGIN      ;точка входа (метка)


; Данные
MEMORY db  'Memory segment:     ',0DH,0AH,'$'       ;сегментный адрес
недоступной памяти
MEDIA db 'Segment media address:    ',0DH,0AH,'$'   ;сегментный адрес среды
TAIL db 'Tail of command line:    ',0DH,0AH,'$'     ;хвост коммандной строки
EMPTY db 'Tail of command line: [EMPTY]',0DH,0AH,'$';пустой хвост
CONTENT db 'Environment scope content:',0DH,0AH, '$'
END_STRING db 0DH,0AH, '$'
PATH db 'Path:  ',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
;--------------------------------

F1 PROC near
   mov ax, ds:[02h]

   mov di, offset MEMORY
   add di, 19
   call WRD_TO_HEX
   mov dx, offset MEMORY
   mov AH,09h
   int 21h
   ret
F1 ENDP

F2 PROC near
   mov ax, ds:[2Ch]

   mov di, offset MEDIA
   add di, 25
   call WRD_TO_HEX
   mov dx, offset MEDIA
   mov AH,09h
   int 21h
   ret
```

```
F2 ENDP

F3 PROC near
    mov cx, 0
    mov cl, ds:[80h]
    mov si, offset TAIL
    add si, 22
    cmp cl, 0          ;если пусто
    je empty_tail
    mov di, 0
    mov ax, 0
read_tail:
    mov al, ds:[81h+di]
    inc di
    mov [si], al
    inc si
    loop read_tail     ;цикл считывания

    mov dx, offset TAIL
    jmp write_tail
empty_tail:
    mov dx, offset EMPTY
write_tail:
    mov AH,09h
    int 21h
    ret
F3 ENDP

F4 PROC near
    mov dx, offset CONTENT
    mov AH,09h
    int 21h
    mov di, 0
    mov ds, ds:[2Ch]
read_str:
    cmp byte ptr [di], 0
    je end_str
    mov dl, [di]
    mov ah, 02h
    int 21h
    jmp find_end
end_str:
```

```
    cmp byte ptr [di+1],00h
    je find_end
    push ds
    mov cx, cs
    mov ds, cx
    mov dx, offset END_STRING
    mov AH,09h
    int 21h
    pop ds
find_end:
    inc di
    cmp word ptr [di], 0001h
    je read_path
    jmp read_str
read_path:
    push ds
    mov ax, cs
    mov ds, ax
    mov dx, offset PATH
    mov AH,09h
    int 21h
    pop ds
    add di, 2
loop2:
    cmp byte ptr [di], 0
    je break
    mov dl, [di]
    mov ah, 02h
    int 21h
    inc di
    jmp loop2
break:
    ret
F4 ENDP

; Код
BEGIN:
    call F1    ;определение адреса недоступной памяти
    call F2    ;определение сегментного адреса среды
    call F3    ;определение хвоста
    call F4    ;получает содержимое области среды и путь
```

```
    xor AL,AL
    mov AH,4Ch
    int 21H
TESTPC ENDS
END START; конец модуля, START - точка выхода
```