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

Студент гр. 0382 _____ Злобин А. С.

Преподаватель _____ Губкин А. Ф.

Санкт-Петербург

2022

Цель работы.

Исследование интерфейса управляющей программы и загрузочных модулей. Исследование префикса сегмента программы (PSP) и среды, передаваемой программе.

Задание.

Требуется 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.

При работе были использованы/созданы следующие процедуры:

- Процедура GET_UNVALIABLE_MEMORY выводит сегментный адрес первого байта недоступной памяти с помощью чтения соответствующих байтов в PSP.
- Процедура GET_ENVIRONMENT_ADRESS выводит сегментный адрес среды с помощью чтения соответствующих байтов в PSP.
- Процедура GET_CMD_TAIL печатает хвост командной строки. Сначала считывается количество символов в хвосте, а затем происходит их печать с помощью прерывания 21h с кодом 2
- Процедура GET_CONTENT печатает содержимое области среды и путь содержимого модуля, используя процедуру выше

В результате выполнения были получены следующие значения(рис.1):

Рисунок 1 – результат работы программы

Вопросы.

Сегментный адрес недоступной памяти:

1) На какую область памяти указывает адрес недоступной памяти?

На сегментный адрес основной оперативной памяти, расположенной расположенный за пределами выделенной программе памяти.

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

Первый байт после программы

3) Можно ли в эту область памяти писать?

Да, можно, так как в DOS общее адресное пространство.

Среда, передаваемая программе:

1) Что такое среда?

Среда представляет собой область памяти, в которой в виде символьных строк записаны значения переменных, называемых переменными среды (имя=значение). Здесь переменная и значение – любые текстовые величины, байт 0 завершает каждую строку.

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

Изначально среда создаётся при запуске ОС. Когда одна программа запускает другую программу, то запущенная программа получает свой собственный экземпляр блока среды, который является точной копией среды родителя, но можно создать и другую среду. Следовательно, перед запуском приложения, она может быть изменена в соответствии с требованиями этого приложения. То есть копируется содержимое, которое было создано при загрузке ОС и также, если это необходимо, добавляются дополнительные параметры для данной программы, поэтому прикладная программа имеет доступ к системным переменным и к переменным, включённым в данное окружение.

3) Откуда берётся информация, записываемая в среду?

Из системного пакетного файла AUTOEXEC.BAT, который расположен в корневом каталоге загрузочного устройства. В нём содержатся системные переменные. Остальные переменные могут добавляться перед запуском программ в процессе работы родительских программ.

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
ИСХОДНЫЕ КОДЫ ПРОГРАММ

Название файла: lab2.asm

```
     TESTPC SEGMENT
   ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
   ORG 100H
START: JMP BEGIN
; Данные
UNVALIABLE_MEMORY db 'Locked memory address:     h',13,10,'$'
ENVIRONMENT_ADRESS db 'Environment address:     h',13,10,'$'
CMD_TAIL db 'Command line tail:        ',13,10,'$'
NULL_TAIL db 'In Command tail no sybmols',13,10,'$'
CONTENT db 'Content:',13,10, '$'
END_STRING db 13, 10, '$'
PATH db 'Patch:  ',13,10,'$'

; 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
   mov AH,09h
   int 21h
   ret
PRINT ENDP
;------------------------------

GET_UNVALIABLE_MEMORY PROC near
```

```
   mov ax,ds:[02h]
   mov di, offset UNVALIABLE_MEMORY
   add di, 26
   call WRD_TO_HEX
   mov dx, offset UNVALIABLE_MEMORY
   call PRINT
   ret
GET_UNVALIABLE_MEMORY ENDP

GET_ENVIRONMENT_ADRESS  PROC near
   mov ax,ds:[2Ch]
   mov di, offset PSP_MEMORY
   add di, 24
   call WRD_TO_HEX
   mov dx, offset ENVIRONMENT_ADRESS
   call PRINT
   ret
GET_ENVIRONMENT_ADRESS ENDP

GET_CMD_TAIL PROC near
   xor cx, cx
   mov cl, ds:[80h]
   mov si, offset CMD_TAIL
   add si, 19
   cmp cl, 0h
   je empty_tail
   xor di, di
   xor ax, ax
readtail:
   mov al, ds:[81h+di]
   inc di
```

```
   mov [si], al
   inc si
   loop readtail
   mov dx, offset CMD_TAIL
   jmp end_tail
empty_tail:
      mov dx, offset NULL_TAIL
end_tail:
   call PRINT
   ret
GET_CMD_TAIL ENDP

GET_CONTENT PROC near
   mov dx, offset CONTENT
   call PRINT
   xor di,di
   mov ds, ds:[2Ch]
read_string:
   cmp byte ptr [di], 00h
   jz end_str
   mov dl, [di]
   mov ah, 02h
   int 21h
   jmp find_end
end_str:
   cmp byte ptr [di+1],00h
   jz find_end
   push ds
   mov cx, cs
   mov ds, cx
   mov dx, offset END_STRING
```

```
   call PRINT
   pop ds
find_end:
   inc di
   cmp word ptr [di], 0001h
   jz read_path
   jmp read_string
read_path:
   push ds
   mov ax, cs
   mov ds, ax
   mov dx, offset PATH
   call PRINT
   pop ds
   add di, 2
loop_path:
   cmp byte ptr [di], 00h
   jz complete
   mov dl, [di]
   mov ah, 02h
   int 21h
   inc di
   jmp loop_path
complete:
   ret
GET_CONTENT ENDP

; Код
BEGIN:
   call GET_UNVALIABLE_MEMORY
   call GET_ENVIRONMENT_ADRESS
```

```
    call GET_CMD_TAIL
    call GET_CONTENT

    xor AL,AL
    mov AH,4Ch
    int 21H
TESTPC ENDS
END START
```