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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студентка гр. 9383 _____ Чебесова И.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

ПОСТАНОВКА ЗАДАЧИ

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 .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.

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

Функции, используемые в программе.

TETR_TO_HEX – переводит десятичную цифру в год ее символа в таблице ASCII.

BYTE_TO_HEX – байт AL переводится в два символа шестнадцатеричного числа. Ответ в AX.

WRD_TO_HEX – переводит в 16 с/с 16-ти разрядного числа.

BYTE_TO_DEC – перевод в 10 с/с, SI – адрес поля младшей цифры.

PRINT_MESSAGE – функция печати строки на экран.

PRINT_MESSAGE_BYTE – функция печати символа на экран.

PRINT_EOF – функция печати символа переноса строки.

UMA_TASK – функция, выводящая сегментный адрес недоступной памяти.

ENV_ADDRESS_TASK – функция, выводящая сегментный адрес среды, передаваемой программе, в шестнадцатеричном виде.

COMMAND_TAIL_TASK – функция, выводящая хвост командной строки в символьном виде.

ENV_CONTENT_TASK – функция, выводящая содержимое области среды в символьном виде.

MODULE_PATH_TASK – функция, выводящая путь загружаемого модуля.

РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ ПРОБЛЕМ

Шаг 1. Был написан текст и построен .COM модуль, который выводит нужную по заданию лабораторной работы информацию.

Рисунок 1. Демонстрация работы .COM модуля с пустым хвостом

Рисунок 2. Демонстрация работы .COM модуля с хвостом “tail”

Ответы на контрольные вопросы

«Сегментный адрес недоступной памяти»:

1. На какую область памяти указывает адрес недоступной памяти?

   Ответ: адрес недоступной памяти указывает на первый байт после области памяти, которая была отведена на программу.

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

   Ответ: по отношению области памяти, отведенной программе, адрес располагается в сторону увеличения адресов.

3. Можно ли в эту область памяти писать?

   Ответ: можно, потому что в DOS нет специальной защиты от перезаписи памяти.

«Среда, передаваемая программе»:

1. Что такое среда?

   Ответ: среда – это такая область памяти, которая хранит в себе переменные среды, а они в свою очередь хранят в себе определенную информацию о состоянии системы.

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

   Ответ: в начале среда создается при запуске операционной системы. А уже при запуске приложения/программы текущая среда копируется в адресное пространство приложения/программы.

3. Откуда берется информация, записываемая в среду?

   Ответ: для DOS эта информация берется из специального системного файла - autoexec.bat, который располагается в корневом каталоге.

Выводы.

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

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

lab2.asm:

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

;-----------------------------
UNAVAILABLE_MEMORY_ADDRESS db 'Unavailable memory address:      ',
0DH, 0AH, '$'
ENV_ADDRESS db 'Segment environment address:    ', 0DH, 0AH, '$'
COMMAND_TAIL db 'Command tail:$'
COMMAND_TAIL_EMPTY db 'Command tail: empty', 0DH, 0AH, '$'
ENV_CONTENT db 'Segment environment content:', 0DH, 0AH, '$'
MODULE_PATH db 'Module path:$'

;-----------------------------
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
    push CX
    mov AH, AL
    call TETR_TO_HEX
    xchg AL, AH
    mov CL, 4
    shr AL, CL
    call TETR_TO_HEX
    pop CX
    ret
```

```
BYTE_TO_HEX ENDP

WRD_TO_HEX PROC near
    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
    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_MESSAGE PROC near
     push AX
    mov AH, 9
    int 21h
     pop AX
    ret
PRINT_MESSAGE ENDP

PRINT_MESSAGE_BYTE  PROC  near
    push AX
    mov AH, 02h
    int 21h
    pop AX
    ret
PRINT_MESSAGE_BYTE  ENDP

PRINT_EOF PROC near
    push AX
    push DX
    mov DL, 0dh
    call PRINT_MESSAGE_BYTE
    mov DL, 0ah
    call PRINT_MESSAGE_BYTE
    pop DX
    pop AX
    ret
PRINT_EOF ENDP

;-----------------------
UMA_TASK PROC near
    push AX
    push DI
```

```
    mov AX,DS:[02h]
     mov DI, offset UNAVAILABLE_MEMORY_ADDRESS
     add DI, 30
     call WRD_TO_HEX
     mov DX, offset UNAVAILABLE_MEMORY_ADDRESS
     call PRINT_MESSAGE
    pop DI
    pop AX
     ret
UMA_TASK ENDP

ENV_ADDRESS_TASK PROC near
    push AX
    push cx
    push DI
    mov AX,DS:[2ch]
     mov DI, offset ENV_ADDRESS
     add DI, 31
     call WRD_TO_HEX
     mov DX, offset ENV_ADDRESS
     call PRINT_MESSAGE
    pop DI
    pop CX
    pop AX
     ret
ENV_ADDRESS_TASK ENDP

COMMAND_TAIL_TASK PROC near
    push AX
    push CX
    push DX
    push DI
    xor CX, CX
    xor DI, DI
    mov CL, DS:[80h]
    cmp CL, 0
    je empty
    mov DX, offset COMMAND_TAIL
```

```
    call PRINT_MESSAGE
for_loop:
    mov DL, DS:[81h + DI]
    call PRINT_MESSAGE_BYTE
    inc DI
    loop for_loop
    call PRINT_EOF
    jmp restore
empty:
    mov DX, offset COMMAND_TAIL_EMPTY
    call PRINT_MESSAGE
restore:
    pop DI
    pop DX
    pop CX
    pop AX
     ret
COMMAND_TAIL_TASK ENDP

ENV_CONTENT_TASK PROC near
    push AX
    push DX
    push ES
    push DI
    mov DX, offset ENV_CONTENT
    call PRINT_MESSAGE
    xor DI, DI
    mov AX, ds:[2ch]
    mov ES, AX
for_loop_2:
    mov DL, ES:[DI]
    cmp DL, 0
    je end_2
    call PRINT_MESSAGE_BYTE
    inc DI
    jmp for_loop_2
end_2:
    call PRINT_EOF
```

```
    inc DI
    mov DL, ES:[DI]
    cmp DL, 0
    jne for_loop_2
    call MODULE_PATH_TASK
    pop DI
    pop ES
    pop DX
    pop AX
     ret
ENV_CONTENT_TASK ENDP

MODULE_PATH_TASK PROC near
    push AX
    push DX
    push ES
    push DI
    mov DX, offset MODULE_PATH
     call PRINT_MESSAGE
     add DI, 3
for_loop_3:
     mov DL, ES:[DI]
     cmp DL,0
     je restore_2
     call PRINT_MESSAGE_BYTE
     inc DI
     jmp for_loop_3
restore_2:
    call PRINT_EOF
    pop DI
    pop ES
    pop DX
    pop AX
     ret
MODULE_PATH_TASK ENDP

;-----------------------------
BEGIN:
```

```
    call UMA_TASK
    call ENV_ADDRESS_TASK
    call COMMAND_TAIL_TASK
    call ENV_CONTENT_TASK

    xor AL, AL
    mov AH, 4ch
    int 21h

TESTPC  ENDS
        END START
```