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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студентка гр. 7383 Маркова А.В.

Преподаватель Ефремов М.А.

Санкт-Петербург

2019

Постановка задачи.

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.

Описание процедур, используемых в работе:

- TETR_TO_HEX – вспомогательная функция, которая переводит из двоичной в шестнадцатеричную систему счисления, используется для работы функции BYTE_TO_HEX;
- BYTE_TO_HEX – переводит байтовое число из регистра AL в шестнадцатеричную систему счисления;
- WRD_TO_HEX – переводит число из регистра AX в шестнадцатеричную систему;
- BYTE_TO_DEC – переводит байтовое число в десятичную систему счисления;
- LINE_OUTPUT – выводит сообщение на экран;
- SEGMENT_INACCESSIBLE – получение сегментного адреса недоступной памяти, взятого из PSP, в шестнадцатеричном виде;
- SEGMENT_ENVIRONMENT – получение сегментного адреса среды, передаваемой программе, в шестнадцатеричном виде;
- ENVIRONMENT – выведение на экран содержимого области среды в символьном виде;
- PATH – выведение на экран пути загруженного модуля;
- TAIL – выведение на экран хвоста командной строки в символьном виде;

Таблица 1 – Описание структур данных:

Название	Тип	Назначение
seg_address_inaccessible	db	Сегментный адрес недоступной памяти (взят из PSP)
seg_address_environment	db	Сегментный адрес среды, передаваемой программе
contents_environment	db	Содержимое области среды
loadable_module_path	db	Путь загружаемого модуля
command_line_tail	db	Хвост командной строки
command_empty	db	Указывает на то, что хвост командной строки пуст
endl	db	Новая строка

Ход работы.

1. Был написан код исходного .COM модуля, который определяет сегментный адрес недоступной памяти, сегментный адрес среды, хвост командной строки, содержимое области и путь загружаемого модуля.
2. Смонтирован виртуальный диск k с каталогом tasm.
3. Было произведено транслирование программы с помощью строки tasm «имя файла», в последствии чего создался объектный файл, результат вызова команды показан на рис. 1.

Рисунок 1 – Транслирование программы

4. Линковка загрузочного модуля с помощью команды tlink/t.
5. Получен LR2.COM модуль из исходного кода для .COM модуля.
6. Результаты работы программы показаны на рис. 2 – 3.

Рисунок 2 – Результат выполнения команды lr2

Рисунок 3 – Результат выполнения команды lr2 hello!

Выводы.

В ходе данной лабораторной работы были исследованы интерфейс управляющей программы и загрузочного модуля. Был написан текст исходного .COM файла, который выводит на экран сегментный адрес недоступной памяти, сегментный адрес среды, хвост командной строки, содержимое области и путь загружаемого модуля. Код программы представлен в приложении А.

Ответы на контрольные вопросы.

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

   На область, которая является доступной для загрузки программы.

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

Этот адрес расположен сразу после области памяти, которая выделена программе.

3. Можно ли в эту область памяти писать?

   Можно, так как DOS не имеет механизма защиты памяти.

Среда, передаваемая программе

1. Что такое среда?

Среда – область памяти, которая содержит переменные, называемые переменными средами и содержащие некоторые данные об операционной системе, в виде последовательности символьных строк (имя = параметр), где каждая строка завершается байтом нулей.

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

Среда создаётся при начальной загрузке DOS. При запуске программы существующая среда просто копируется.

3. Откуда берётся информация, записываемая в среду?

При работе с операционной системой MS – DOS, информация, которая заносится в среду, берётся из системного файла AUTOEXEC.BAT

ПРИЛОЖЕНИЕ А

LR2.ASM

```
EOFLine EQU '$'
TESTPC SEGMENT
        ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
        ORG 100H
START: JMP BEGIN
;ДАННЫЕ
;-----------------------------------------------------------------------
endl                      db ' ',0DH,0AH,'$'
seg_address_inaccessible db 'Segment address of inaccessible memory:
',0DH,0AH,EOFLine
seg_address_environment  db 'Segment address of the environment:
',0DH,0AH,EOFLine
contents_environment     db 'The contents of the environment area:
',0DH,0AH,EOFLine
loadable_module_path     db 'Loadable module path: ',0DH,0AH,EOFLine
command_line_tail        db 'Command-line tail:',EOFLine
command_empty            db ' Empty',0DH,0AH,EOFLine
;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
     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
;--------------------------------------------------------------------------
LINE_OUTPUT PROC near
    push AX
    mov  AH, 09h
      int  21H
```

```
      pop  AX
      ret
LINE_OUTPUT ENDP

SEGMENT_INACCESSIBLE PROC near
    push AX
      push DI
      mov  AX, DS:[02h]
      mov  DI, OFFSET seg_address_inaccessible
      add  DI, 43
      call WRD_TO_HEX
      pop  DI
      pop  AX
      ret
SEGMENT_INACCESSIBLE ENDP

SEGMENT_ENVIRONMENT PROC near
    push AX
      push DI
      mov  AX, DS:[2Ch]
      mov  DI, OFFSET seg_address_environment
      add  DI, 39
      call WRD_TO_HEX
      pop  DI
      pop  AX
      ret
SEGMENT_ENVIRONMENT ENDP

ENVIRONMENT PROC near
    push AX
      push DX
      push DS
      push ES
      mov  DX, OFFSET contents_environment
      call LINE_OUTPUT
      mov  AH, 02h
      mov  ES, DS:[2Ch]
      xor  SI, SI
content:
    mov  DL, ES:[SI]
      int  21h
      cmp  DL, 00h
      je   content_end
```

```
      inc  SI
      jmp  content
content_end:
    mov  DX, OFFSET endl
      call LINE_OUTPUT
      inc  SI
      mov  DL, ES:[SI]
      cmp  DL, 00h
      jne  content
      mov  DX, OFFSET endl
      call LINE_OUTPUT
      pop  ES
      pop  DS
      pop  DX
      pop  AX
      ret
ENVIRONMENT ENDP

PATH PROC near
    push AX
      push DX
      push DS
      push ES
      mov  DX, OFFSET loadable_module_path
    call LINE_OUTPUT
    add  SI, 04h
    mov  AH, 02h
    mov  ES, DS:[2Ch]
way:
    mov  DL, ES:[SI]
    cmp  DL, 00h
    je   way_end
    int  21h
    inc  SI
    jmp  way
way_end:
    pop  ES
    pop  DS
    pop  DX
    pop  AX
    ret
PATH ENDP
```

```
TAIL PROC near
    push AX
     push CX
     push DX
     push SI
    mov  DX, offset command_line_tail
     call LINE_OUTPUT
     mov  CL, DS:[80h]
     cmp  CL, 00h
     je   empty
     mov  SI, 81h
     mov  AH, 02h
command:
     mov  DL, DS:[SI]
     int  21h
     inc   SI
     loop command
     mov  DX, offset endl
     call LINE_OUTPUT
     jmp  end_tail
empty:
    mov  AL, 00h
     mov  [DI], AL
     mov  DX, OFFSET command_empty
     call LINE_OUTPUT
end_tail:
     pop  SI
     pop  DX
     pop  CX
     pop  AX
     ret
TAIL ENDP
;-------------------------------------------------------------------------
BEGIN:
    call SEGMENT_INACCESSIBLE
    mov  DX, OFFSET seg_address_inaccessible
    call LINE_OUTPUT
    call SEGMENT_ENVIRONMENT
    mov  DX, OFFSET seg_address_environment
    call LINE_OUTPUT
     call TAIL
     mov  DX, OFFSET endl
    call LINE_OUTPUT
```

```
    call ENVIRONMENT
     call PATH
;ВЫХОД ИЗ DOS
;-------------------------------------------------------------------------
     xor  AL,AL
     mov  AH,4CH
     int  21h
TESTPC   ENDS
         END START
```