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

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Операционные системы»

Тема: Исследование интерфейсов программных модулей

Студент гр. 0382 _____ Ильин Д.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

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 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.

Сохраните результаты, полученные программой, и включите их в отчет.

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

Порядок выполнения работы.

1. За основу берём шаблон из методички для формата .com
2. Пишем строки, которые потом будем выводить для каждой из позадач, дабы лучше воспринималась информация
3. Создадим процедуры для печати символа и строки(print_str и print_symb)
4. Далее, в созданной процедуре для данной лабораторной(LAB2), выполняем каждую из поставленных подзадач из пункта первого вышеупомянутого задания

Тестирование.

Рисунок 1 — работа созданной программы

Вывод: программа работает корректно.

Контрольные вопросы.

Сегментный адрес недоступной памяти

1. На какую область памяти указывает адрес недоступной памяти?

   На сегмент, находящийся за выделенной программе памяти.
2. Где расположен этот адрес по отношению области памяти, отведенной программе?

   После памяти, выделенной программе.
3. Можно ли в эту область памяти писать?

   Можно.

Среда, передаваемая программе

1. Что такое среда?

DOS имеет специальную область памяти, называемую средой или же окружением, в которой он хранит набор строк символов, которые могут использоваться программами. Формат: имя-переменной (строка символов, не содержащая знаков равенства и пробелов) = значение (любая строка символов).

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

   Изначально - при запуске ОС, однако при запуске приложений, создается копия этой среды, в неё добавляются дополнительные параметры для данного приложения, если это требуется.

3. Откуда берется информация, записываемая в среду?

Из файла Autoexec.bat при запуске ОС, командой SET. Исключения:

- CONFIG - определяется в файле Config.sys,

- PROMPT - определяется отдельной командой DOS - оболочки.

Вывод.

В ходе работы были изучено устройство сегмента PSP. Была написана программа, выполняющая поставленные задачи из задания приведённого выше, а точнее выводящая в консоль определённые данные.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД МОДУЛЕЙ

Файл lb2_com.asm

```
; Шаблон текста программы на ассемблере для модуля типа .COM
TESTPC SEGMENT
    ASSUME CS:TESTPC, DS:TESTPC, ES:NOTHING, SS:NOTHING
    ORG 100H
START: JMP BEGIN

; ДАННЫЕ
SEG_ADR_MEM_D db 'Segment address of unavailable memory:
h',0DH,0AH,'$'
SEG_ADR_ENV_D db 'Segment address of the environment:     h',0DH,0AH,'$'
TAIL_COMM_LINE_D db 'The tail of the command line in symbolic form:
',',$'
EMPTY_TAIL_COMM_LINE_D db 'The tail of the command is empty',0DH,0AH,'$'
CONT_ENV_AREA_D db 'The contents of the environment area in symbolic
form: ',0DH,0AH,'$'
PATH_LOAD_MODULE_D db 'The path of the loaded module: ','$'


;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 шестн. 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_str PROC near
    mov AH, 09h
    int 21h
    ret
print_str ENDP

print_symb PROC near
    push ax
    mov ah, 02h
    int 21h
    pop ax
    ret
print_symb ENDP

LAB2 PROC near
```

```
    push AX
     push CX
     push DX
     push DI
     push ES

    mov ax, ds:[2h]
    mov di, offset SEG_ADR_MEM_D + 42
    call WRD_TO_HEX
    mov dx, offset SEG_ADR_MEM_D
    call print_str

    mov ax, ds:[2Ch]
    mov di, offset SEG_ADR_ENV_D + 39
    call WRD_TO_HEX
    mov dx, offset SEG_ADR_ENV_D
    call print_str

    mov cl, ds:[80h]
    cmp cl, 0h
    jne normal_command_tail
    mov dx, offset EMPTY_TAIL_COMM_LINE_D
    call print_str
    jmp cont_env
normal_command_tail:
    mov dx,offset TAIL_COMM_LINE_D
    call print_str
    mov di,81h
loop_command_tail:
    mov dl, ds:[di]
    call print_symb
    inc di
    loop loop_command_tail
    mov dl, 0Dh
    call print_symb
    mov dl, 0Ah
    call print_symb
cont_env:

    mov dx, offset CONT_ENV_AREA_D
    call print_str
    mov es, ds:[2Ch]
    xor di, di
loop_cont_env:
    mov dl, es:[di]
    cmp dl, 0h
    je final_cont_env
    call print_symb
    inc di
    jmp loop_cont_env
final_cont_env:
```

```
    mov dl, 0Dh
    call print_symb
    mov dl, 0Ah
    call print_symb
    inc di
    mov dl, es:[di]
    cmp dl, 0h
    jne loop_cont_env

    mov di, 3
    mov dx, offset PATH_LOAD_MODULE_D
    call print_str
loop_path:
    mov dl, es:[di]
    cmp dl, 0h
    je final_path
    call print_symb
    inc di
    jmp loop_path
final_path:
    mov dl, 0Dh
    call print_symb
    mov dl, 0Ah
    call print_symb

    pop ES
     pop DI
     pop DX
     pop CX
     pop AX

    ret
LAB2 ENDP


; КОД
BEGIN:
    call LAB2
; Выход в DOS
    xor AL,AL
    mov AH,4Ch
    int 21H
TESTPC ENDS
    END START ;конец модуля, START - точка входа
```