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

ЛАБОРАТОРНАЯ РАБОТА № 6

по дисциплине «Операционные системы»

Тема: Построение модуля динамической структуры

Студент гр. 9381 _____ Матвеев А. Н.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры. В отличие от предыдущих лабораторных работ в этой работе рассматривается приложение, состоящее из нескольких модулей, а не из одного модуля простой структуры. В этом случае разумно предположить, что все модули приложения находятся в одном каталоге и полный путь в этот каталог можно взять из среды, как это делалось в работе 2. Понятно, что такое приложение должно запускаться в соответствии со стандартами ОС. В работе исследуется интерфейс между вызывающим и вызываемым модулями по управлению и по данным. Для запуска вызываемого модуля используется функция **4B00h** прерывания int 21h. Все загрузочные модули находятся в одном каталоге. Необходимо обеспечить возможность запуска модуля динамической структуры из любого каталога.

Задание.

Шаг 1. Для выполнения лабораторной работы необходимо написать и отладить программный модуль типа .EXE, который выполняет функции:

1) Подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором находится он сам. Вызываемому модулю передается новая среда, созданная вызывающим модулем и новая командная строка.

2) Вызываемый модуль запускается с использованием загрузчика.

3) После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы. Необходимо проверять причину завершения и, в зависимости от значения, выводить соответствующее сообщение. Если причина завершения 0, то выводится код завершения. В качестве вызываемой программы необходимо взять программу ЛР 2, которая распечатывает среду и командную строку. Эту программу следует немного модифицировать, вставив перед выходом из нее обращение к функции ввода символа с клавиатуры. Введенное значение записывается в регистр AL и затем происходит обращение к функции выхода 4Ch прерывания int 21h.

Шаг 2. Запустите отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры. Введите произвольный символ из числа A-Z. Посмотрите причину завершения и код. Занесите полученные данные в отчет.

Шаг 3. Запустите отлаженную программу, когда текущим каталогом является каталог с разработанными модулями. Программа вызывает другую программу, которая останавливается, ожидая символ с клавиатуры. Введите комбинацию символов Ctrl-C. Посмотрите причину завершения и код. Занесите полученные данные в отчет.

Шаг 4. Запустите отлаженную программу, когда текущим каталогом является какой- либо другой каталог, отличный от того, в котором содержатся разработанные программные модули. Повторите ввод комбинаций клавиш. Занесите полученные данные в отчет.

Шаг 5. Запустите отлаженную программу, когда модули находятся в разных каталогах. Занесите полученные данные в отчет.

Выполнение работы.

1) Написал и отладил программный модуль типа .EXE, который выполняет функции:

- Подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором находится он сам. Вызываемому модулю передается новая среда, созданная вызывающим модулем и новая командная строка.
- Вызываемый модуль запускается с использованием загрузчика.
- После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы. Необходимо проверять причину завершения и, в зависимости от значения, выводить соответствующее сообщение. Если причина завершения 0, то выводится код завершения.

Модуль lab2.com был модифицирован так, чтобы в конце его выполнения считывался символ с клавиатуры.

2) Запустил отлаженную программу, когда текущим каталогом являлся каталог с разработанными модулями. Программа lab6.exe вызывала другую программу (lab2.com), которая остановилась, ожидая символ с клавиатуры. Ввёл произвольный символ из числа A-Z. (см. рис. 1).

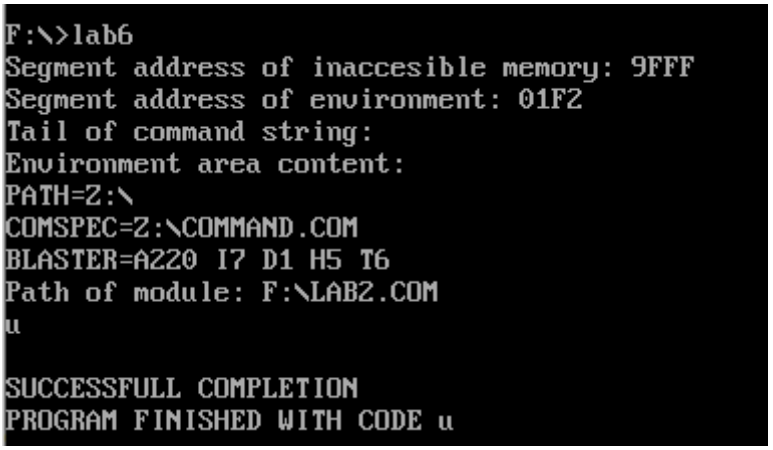

Рис. 1.

Как видно из рисунка 1, после ввода символа с клавиатуры в конце программы отобразилась нажатая клавиша и сообщение о выходе из программы в нормальном режиме.

3) Запустил отлаженную программу, когда текущим каталогом являлся каталог с разработанными модулями. Программа lab6.exe вызывала другую программу (lab2.com), которая остановилась, ожидая символ с клавиатуры. Ввел комбинацию символов Ctrl-C (см. рис. 2).

Рис. 2.

Как видно из рисунка 2, программа снова нормально завершилась. Символ сердечка допустим, т.к. комбинация клавиш Ctrl + C не поддерживается в DOSBox.

4) Запустил отлаженную программу, когда текущим каталогом являлся какой- либо другой каталог, отличный от того, в котором содержатся разработанные программные модули. Повторил ввод комбинаций клавиш (см. рис. 3-4).

Рис. 3-4.

5) Запустил отлаженную программу, когда модули находились в разных каталогах: lab6.exe остался в директории OTHER, lab2.com был перемещён в корневую директорию (см. рис. 5).

Рис. 5.

Как видно из рис. 5, вызываемая программа не была загружена, поскольку файл не был найден. (Код 2).

Вывод.

В результате выполнения лабораторной работы были изучены возможности построения загрузочного модуля динамической структуры.

Ответы на контрольные вопросы.

1. Как реализовано прерывание Ctrl-C?

ОТВЕТ: DOS вызывает INT 23H, когда распознает, что нажата комбинация Ctrl-C. Адрес в этом векторе (0000:008Ch) – адрес, по которому передается управление, когда DOS распознает, что пользователь нажал Ctrl-C. Этот адрес (адрес по вектору INT 23H) копируется в PSP функциями DOS 26H и 4Ch и восстанавливается из PSP при завершении программы.

2. В какой точке заканчивается вызываемая программа, если код причины завершения 0?

ОТВЕТ: Если код причины завершения 0, то вызываемая программа заканчивается в месте вызова функции 4Ch прерывания int 21h.

3. В какой точке заканчивается вызываемая программа по прерыванию Ctrl-C?

ОТВЕТ: При нажатии во время выполнения программы комбинации клавиш Ctrl-C программа завершится непосредственно в том месте, в котором было вызвано прерывание. В нашем случае это произошло в месте ожидания нажатия клавиши: 01h вектора прерывания 21h.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл lab6.asm.

```
 ASSUME CS:CODE, DS:DATA, SS:AStack
AStack SEGMENT STACK
 DW 64 DUP(?)
AStack ENDS

DATA SEGMENT
 ENVIR_ADDR     dw 0 ; сегментный адрес среды. Если он равен 0, то вызываемая программа наследует
 ;среду вызывающей программы
 ; в противном случае вызывающая программа должна сформировать область памяти в качестве среды,
 ;начинающуюся с адреса, кратного 16 и переместить этот адрес в блок параметров
 CMD            dd 0 ; сегмент и смещение командной строки
 SEG_1FCB  dd 0 ; сегмент и смещение первого FCB
 SEG_2FCB  dd 0 ; сегмент и смещение второго FCB
 KEEP_SP        dw 0
 KEEP_SS        dw 0
 PATH      db   '                                                lab2.com
',0dH,0ah,'$',0h
 EOL            db '   ',0dh,0ah,'$'
 ERR_1_7   db 'THE MEMORY CONTROL BLOCK IS DESTROYED (CODE 7)', 0dh, 0ah, '$'
    ERR_1_8          db 'NOT ENOUGHT MEMORY TO RUN THE FUNCTION (CODE 8)', 0dh, 0ah, '$'
    ERR_1_9          db 'WRONG MEMORY BLOCK ADDRESS (CODE 9)', 0dh, 0ah, '$'
DATA ENDS

CODE SEGMENT
;-----------------------------
WRITE PROC NEAR ;Вывод на экран сообщения
     push ax
     mov  ah, 09h
     int  21h
```

```
    pop    ax
    ret
WRITE ENDP
;-----------------------------
COMPLETION_PROC PROC NEAR
 jmp BEGIN_COMPLETION
 END_MES   db   'PROGRAM FINISHED WITH CODE    ', 0dh, 0ah, '$'
 END_MES_0 db   'SUCCESSFULL COMPLETION', 0dh, 0ah, '$'
 END_MES_1 db   'COMPLETION BY CTRL-BREAK', 0dh, 0ah, '$'
 END_MES_2 db   'COMPLETION BY DEVICE ERROR', 0dh, 0ah, '$'
 END_MES_3 db    'COMPLETION BY 31H FUNCTION', 0dh, 0ah, '$' ;функция
31h оставляет программу резидентной
BEGIN_COMPLETION:
 push ds
 push ax
 push dx
 push bx
 ; подготовка выполнена, вызывается загрузчик OS:
 mov ax, 4D00h
 int 21h
 ; если вызываемая программа не была загружена, то устанавливается
 ; флаг переноса CF = 1 и в AX заносится код ошибки
 push ax
 mov ax, SEG END_MES
 mov ds, ax
 pop ax
 ; анализ кодов ошибки
 lea dx, END_MES_0
 cmp ah, 0h ; нормальное завершение
 je WRITE_ER1

 lea dx, END_MES_1
 cmp ah, 1h ; завершение по CTRL-BREAK
 je WRITE_ER1

 lea dx, END_MES_2 ; завершение по ошибке устройства
 cmp ah, 2h
 je WRITE_ER1
```

```
 lea dx, END_MES_3 ; завершение по функции 31h, оставляющей программу
резидентной
 cmp ah, 3h

WRITE_ER1:
 call WRITE
 lea dx, END_MES
 mov bx, dx
 add bx, 1Bh
 mov byte ptr [bx], al
 call WRITE

 pop bx
 pop dx
 pop ax
 pop ds
 ret
COMPLETION_PROC ENDP
;-----------------------------
ERROR_PROC PROC NEAR
; если вызываемая программа не была загружена, то устанавливается флаг
переноса CF=1
; и в AX заносится код ошибки
 jmp BEGIN_ERR
 ERR_2      DB 'PROGRAM WAS NOT DOWNLOADED', 0DH, 0AH, '$'
 ERR_2_1    DB 'WRONG NUMBER OF THE FUNCTION (CODE 1)', 0DH, 0AH, '$'
 ERR_2_2  DB 'FILE NOT FOUND (CODE 2)', 0DH, 0AH, '$'
 ERR_2_5  DB 'DISK ERROR (CODE 5)', 0DH, 0AH, '$'
 ERR_2_8  DB 'NOT ENOUGHT MEMORY (CODE 8)', 0DH, 0AH, '$'
 ERR_2_10 DB 'WRONG ENVIROMENT STRING (CODE 10)', 0DH, 0AH, '$'
 ERR_2_11 DB 'WRONG FORMAT (CODE 11)', 0DH, 0AH, '$'
BEGIN_ERR:
 push ds
 push ax
 push dx

 push ax
```

```
 mov ax, SEG ERR_2
 mov ds, ax
 pop ax

 lea dx, ERR_2
 call WRITE

 lea dx, ERR_2_1
 cmp ax, 1h ; номер функции неверен
 je WRITE_ERR_2

 lea dx, err_2_2
 cmp ax, 2h ; файл не найден
 je WRITE_ERR_2

 lea dx, err_2_5
 cmp ax, 5h ; ошибка диска
 je WRITE_ERR_2

 lea dx, err_2_8
 cmp ax, 8h ; недостаточный объём памяти
 je WRITE_ERR_2

 lea dx, err_2_10
 cmp ax, 10h ; неправильная строка среды
 je WRITE_ERR_2

 lea dx, ERR_2_11
 cmp ax, 11h ; неверный формат

WRITE_ERR_2:
 call WRITE
 pop dx
 pop ax
 pop ds
 ret
ERROR_PROC ENDP
;-----------------------------
```

```
Main PROC FAR
 mov ax, DATA
 mov ds, ax

 ;Освобождение памяти
 lea bx, ENDPROG
 ; если сегментный адрес среды != 0, то вызываемая программа должна
 ; сформировать область памяти в качестве среды, начинающуюся с адреса
 ; кратного 16 и поместить этот адрес в блок параметров
 mov cl,4h
 shr bx,cl
 add bx,30h ;размер памяти, необходимый для лабораторной 6

 ;код завершения формируется вызываемой программой в регистре AL
 ;перед выходом в OS с помощью функции 4Ch прерывания int 21h
 mov ah,4AH
 int 21h
 ; переход если  CF=0 --> программа загружена и надо обработать её
завершение
 jnc PROCESS_COMPLETION

 lea dx, ERR_1_7
 cmp ax, 7h
 je WRITE_ERR
 lea dx, ERR_1_8
 cmp ax, 8h
 je WRITE_ERR
 lea dx, ERR_1_9
 WRITE_ERR:
     call WRITE
     jmp FINAL

PROCESS_COMPLETION:
 ;Заполнение блока параметров
 mov ENVIR_ADDR, 00h

 mov ax, es
 mov word ptr CMD, ax
```

```
 mov word ptr CMD+2, 0080h

 mov word ptr SEG_1FCB, ax
 mov word ptr SEG_2FCB+2, 005CH

 mov word ptr SEG_2FCB,ax
 mov word ptr SEG_1FCB, 006CH

 ;Подготовка среды, содержащей имя и путь вызываемой программмы
 push es
 push dx
 push bx

 mov es, es:[2Ch]; сегментный адрес среды, передаваемый программе
 mov si, 0
ENVIR:
 mov dl, es:[si]
 cmp dl, 00h          ; конец строки?
 je EOL_
 inc si
 jmp ENVIR
EOL_:
 inc si
 mov dl, es:[si]
 cmp dl, 00h          ;конец среды?
 jne ENVIR

 add si, 03h    ; si указывает на начало маршрута

 push di
 lea di, PATH
PATH_:
 mov dl, es:[si]
 cmp dl, 00h          ;конец маршрута?
 je EOL2
 mov [di], dl
 inc di
 inc si
```

```
 jmp PATH_
EOL2:
 sub di, 05h
 mov [di], byte ptr '2'
 mov [di+2], byte ptr 'c'
 mov [di+3], byte ptr 'o'
 mov [di+4], byte ptr 'm'
 mov [di+5], byte ptr 0h

 pop di
 pop bx
 pop ds
 pop es

 ;Сохраняем содержимое регистров SS и SP переменных

 push ds
 mov KEEP_SP, sp
 mov KEEP_SS, ss

 mov ax, DATA
 mov es, ax     ;es:bx должен указывать на блок параметров
 lea bx, ENVIR_ADDR
 mov ds, ax     ; ds:dx должен указывать на подготовленную строку
 lea dx, PATH

 mov ax, 4B00h  ; Вызываем загрузчик OS
 int 21h

 ;Восстанавливаем параметры
 pop ds
 mov ss, KEEP_SS
 mov sp, KEEP_SP

 push ax
 push dx
 push ds
 mov ax, DATA
```

```
 mov ds, ax
 lea dx, EOL
 call WRITE
 call WRITE
 pop ds
 pop dx
 pop ax

 jnc NULL_ ; CF=0
 call ERROR_PROC
 jmp FINAL
NULL_:
 call COMPLETION_PROC

FINAL:
 xor al, al
 mov ah, 4ch
 int 21h
Main ENDP

ENDPROG:
CODE ENDS
END Main
```