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

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Организация ЭВМ и систем»

Тема: «Построение модуля динамической структуры »

Студент гр. 8381 _____ Переверзев Д.Е.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2019

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.

Выполнение работы.

Программный .EXE модуль, выполняет следующие функции:

1.Подготавливает параметры для запуска загрузочного модуля из того же каталога, в котором он находится сам. Вызываемому модулю передаёт новую среду и новую командную строку.

2.Вызываемый модуль запускается с использованием загрузчика.

После запуска проверяется выполнение загрузчика, а затем результат выполнения вызываемой программы. Проверяет причину завершения и, в зависимости от значения, выводит соответствующее сообщение. Если причина завершения 0, то выводится код завершения.

Была запущена программа, когда оба модуля находятся в текущем каталоге и запуск lab2 с символом «1». Результат работы программы представлен на рис.1

Рисунок 1 – результат работы программы, оба модуля в текущем каталоге. Была запущена программу, когда оба модуля находятся в текущем каталоге и запуск lab2 с комбинацией клавиш «Ctrl+c» . Результат работы программы представлен на рис. 2.

Рисунок 2 – результат работы программы, оба модуля в текущем каталоге, ввод комбинации клавиш “Ctrl+C”.

Была запущена программа, когда оба модуля находятся не в текущем каталоге, и введен символ ‘k’. Результат работы программы представлен на рис.3

Рисунок 3 – результат работы программы, оба модуля не в текущем каталоге, ввод символа ‘y’.

Была запущена программа, когда оба модуля находятся не в текущем

каталоге и введена комбинация клавиш “Ctrl+C”. Результат работы программы представлен на рис. 4.

Рисунок 4 – результат работы программы, оба модуля не в текущем каталоге, ввод комбинации клавиш “Ctrl+C”.

Была запущена программа, когда модули находятся в разных каталогах. Результат работы программы представлен на рис. 5.

```
C:\DOCUME~1\UNIVER~1\OS\LR6\OTHERDIR>LAB6.EXE

File not found

C:\DOCUME~1\UNIVER~1\OS\LR6\OTHERDIR>
```

Рисунок 5 – результат работы программы, модули находятся в разных каталогах.

Ответы на контрольные вопросы.

1. **Как реализовано прерывание Ctrl+C?**

При нажатии комбинации клавиш Ctrl+C вызывается прерывание 23h, которое завершает текущей процесс и передает управление порождаемому процессу.

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

В точке вызова функции 4Ch прерывания int 21h.

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

В точке вызова функции 01h прерывания int 21h.

Выводы.

В процессе выполнения данной лабораторной работы была исследована возможность построения загрузочного модуля динамической структуры.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ. lab6.asm

```
_CODE SEGMENT
        ASSUME CS:_CODE, DS:_DATA, ES:NOTHING, SS:_STACK
PRINT   PROC    near
        push    ax
        mov     ah,09h
        int     21h
        pop     ax
        ret
PRINT   ENDP

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

FreeMemory PROC
        mov     bx,offset DUMMY_SEGMENT
        mov     ax, es
        sub     bx, ax
        mov     cl, 4h
        shr     bx, cl
        mov     ah, 4Ah
        int     21h
        jnc     NO_ERROR
        cmp     ax, 7
        mov     dx, offset Mem_7
        je      YES_ERROR
        cmp     ax, 8
        mov     dx, offset Mem_8
        je      YES_ERROR
        cmp     ax, 9
```

```
        mov     dx, offset Mem_9
    YES_ERROR:
        call    PRINT
        xor     al,al
        mov     ah,4Ch
        int     21H
    NO_ERROR:;create block of parametrs
        mov     ax, es
        mov     ParameterBlock,0
        mov     ParameterBlock+2, ax
        mov     ParameterBlock+4, 80h
        mov     ParameterBlock+6, ax
        mov     ParameterBlock+8, 5Ch
        mov     ParameterBlock+10, ax
        mov     ParameterBlock+12, 6Ch
        ret
FreeMemory ENDP

RunProc PROC    NEAR
        mov     es, es:[2Ch]
        mov     si, 0
    env:
        mov     dl, es:[si]
        cmp     dl, 00h
        je      EOL_
```

```
    inc     si
    jmp     env
EOL_:
    inc     si
    mov     dl, es:[si]
    cmp     dl, 00h
    jne     env
    add     si, 03h
    push    di
    lea     di, PATH
path_:
    mov     dl, es:[si]
    cmp     dl, 00h
    je      EOL2
    mov     [di], dl
    inc     di
    inc     si
    jmp     path_
EOL2:
    sub     di, 8
    mov     [di], byte ptr 'L'
    mov     [di+1], byte ptr 'A'
    mov     [di+2], byte ptr 'B'
    mov     [di+3], byte ptr '2'
    mov     [di+4], byte ptr '.'
```

```
mov     [di+5], byte ptr 'C'
mov     [di+6], byte ptr 'O'
mov     [di+7], byte ptr 'M'
mov     [di+8], byte ptr 0h
pop     di
mov     KEEP_SP, SP
mov     KEEP_SS, SS
push    ds
pop     es
mov     bx,offset ParameterBlock
mov     dx,offset PATH
mov     ax,4B00h
int     21h
jnc     IS_LOADED
push    ax
mov     ax,_DATA
mov     ds,ax
pop     ax
mov     SS,KEEP_SS
mov     SP,KEEP_SP


cmp     ax,1
mov     dx,offset Err_1
je      EXIT1
```

```
    cmp     ax,2
    mov     dx,offset Err_2
    je      EXIT1
    cmp     ax,5
    mov     dx,offset Err_5
    je      EXIT1
    cmp     ax,8
    mov     dx,offset Err_8
    je      EXIT1
    cmp     ax,10
    mov     dx,offset Err_10
    je      EXIT1
    cmp     ax,11
    mov     dx,offset Err_11
EXIT1:
    call    PRINT
    xor     al,al
    mov     ah,4Ch
    int     21h

IS_LOADED:
    mov     ax,4d00h
    int     21h
    cmp     ah,0
    mov     dx,offset End_0
```

```
        je      EXIT2
        cmp     ah,1
        mov     dx,offset End_1
        je      EXIT2
        cmp     ah,2
        mov     dx,offset End_2
        je      EXIT2
        cmp     ah,3
        mov     dx,offset End_3
    EXIT2:
        call    PRINT
        mov     di,offset END_CODE
        call    BYTE_TO_HEX
        add     di,0Ah
        mov     [di],al
        add     di,1h
        xchg    ah,al
        mov     [di],al
        mov     dx, offset END_CODE
        call    PRINT
        ret
RunProc ENDP

MAIN    PROC    NEAR
        mov     ax, _DATA
```

```
        mov     ds, ax
        call    FreeMemory
        call    RunProc
        mov     ax, 4C00h
        int     21h
        ret
    LAST_BYTE:
MAIN    ENDP
_CODE   ENDS
_STACK  SEGMENT STACK
        db  512 dup(0)
_STACK  ENDS
_DATA   SEGMENT
    ParameterBlock dw ?
    dd ?
    dd ?
    dd ?
    Mem_7    DB 0DH, 0AH,'Memory control unit destroyed',0DH,0AH,'$'
        Mem_8        DB 0DH, 0AH,'Not enough memory to perform the function',0DH,0AH,'$'
        Mem_9        DB 0DH, 0AH,'Wrong address of the memory block',0DH,0AH,'$'
    Err_1    DB 0DH, 0AH,'The number of function is wrong',0DH,0AH,'$'
    Err_2    DB 0DH, 0AH,'File not found',0DH,0AH,'$'
    Err_5    DB 0DH, 0AH,'Disk error',0DH,0AH,'$'
    Err_8    DB 0DH, 0AH,'Insufficient value of memory',0DH,0AH,'$'
```

```
    Err_10   DB 0DH, 0AH,'Incorrect environment string',0DH,0AH,'$'
    Err_11   DB 0DH, 0AH,'Wrong format',0DH,0AH,'$'
    End_0    DB 0DH, 0AH,'Normal termination',0DH,0AH,'$'
    End_1    DB 0DH, 0AH,'End by Ctrl-Break',0DH,0AH,'$'
    End_2    DB 0DH, 0AH,'The completion of the device error',0DH,0AH,'$'
    End_3    DB 0DH, 0AH,'Completion by function 31h',0DH,0AH,'$'
    PATH     DB '                                        ',0DH,0AH,'$',0
    KEEP_SS  DW 0
    KEEP_SP  DW 0
    END_CODE DB 'End code:   ',0DH,0AH,'$'
_DATA   ENDS
DUMMY_SEGMENT SEGMENT
DUMMY_SEGMENT ENDS
        END     MAIN
```