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

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Операционные системы»

Тема: Построение модуля динамической структуры

Студент гр. 9381 _____ Гурин С.Н.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Исследование возможности построения загрузочного модуля динамической структуры. В отличии 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.

Порядок выполнения работы.

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.

Был реализован загрузочный модуль динамической структуры, а так же был отредактирован программный модуль из второй лабораторной работы. Была добавлена возможность считать символ с клавиатуры.

Запустим программу из директории с модулями и введем символ A

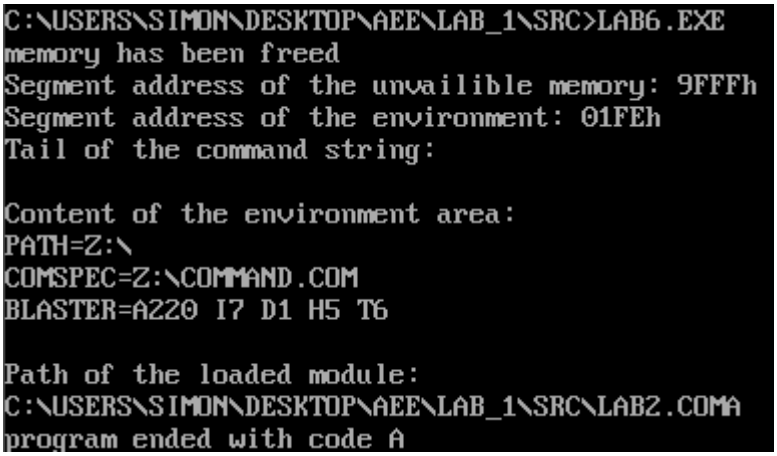

Рис. 1

Теперь запустим программу и завершим ее с помощью Ctrl-C

Рис. 2

Теперь запустим программу, находясь в другой директории

Рис. 3

Теперь запустим программу из другой директории и завершим ее через Ctrl-C.

Рис. 4

Далее запустим программу при условии, что модули находятся в разных директориях.

Рис. 5

Ответы на контрольные вопросы

1) Как реализовано прерывание Ctrl-C?

При нажатии этих клавиш управление передается по адресу 0000:008Ch. Этот адрес копируется в PSP ф-циями 26h и 4Ch и восстанавливается из PSP при выходе из программы.

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

Если код завершения 0, то программа завершается при выполнении ф-ции 4Ch прерывания int 21h.

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

Если клавиши нажаты, то программа завершается в том месте, в котором произошло это нажатие (в месте ожидания нажатия клавиши: 01h вектора прерывания 21h)

Вывод

В ходе выполнения лабораторной работы был написан загрузочный модуль динамической структуры, а так же был изучен принцип работы с памятью.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab6.asm

```
SSTACK                  SEGMENT   STACK
                        DW        128 DUP(?)
SSTACK                  ENDS

DATA                    SEGMENT
PARAMETER_BLOCK         DW        0
                        DD        0
                        DD        0
                        DD        0
PROGRAM                 DB        'LAB2.COM', 0
MEM_FLAG                DB        0
CMD_L                   DB        1H, 0DH
CL_POS                  DB        128 DUP(0)
KEEP_SS                 DW        0
KEEP_SP                 DW        0
KEEP_PSP                DW        0

STR_MCB_CRASH_ERR       DB        'ERR: MCB CRASHED', 0DH, 0AH, '$'
STR_NO_MEM_ERR          DB        'ERR:  THERE  IS  NOT  ENOUGH  MEMORY  TO
EXECUTE THIS FUNCTION', 0DH, 0AH, '$'
STR_ADDR_ERR            DB        'ERR:  INVALID  MEMORY  ADDRESS',  0DH,
0AH, '$'
STR_FREE_MEM            DB        'MEMORY HAS BEEN FREED' , 0DH, 0AH, '$'

STR_FN_ERR              DB        'ERR:  INVALID  FUNCTION  NUMBER',
0DH, 0AH, '$'
STR_FILE_ERROR          DB        'ERR: FILE NOT FOUND', 0DH, 0AH, '$'
STR_DISK_ERR            DB        'ERR: DISK ERROR', 0DH, 0AH, '$'
STR_MEMORY_ERROR        DB        'ERR:  INSUFFICIENT  MEMORY',  0DH,  0AH,
'$'
STR_ENVS_ERR            DB        'ERR:  WRONG  STRING  OF  ENVIRONMENT  ',
0DH, 0AH, '$'
STR_FORMAT_ERR          DB        'ERR: WRONG FORMAT', 0DH, 0AH, '$'
```

```
STR_NORM_FIN        DB      0DH, 0AH, 'PROGRAM ENDED WITH CODE    '
, 0DH, 0AH, '$'
STR_CTRL_END        DB      0DH, 0AH, 'PROGRAM ENDED BY CTRL-BREAK'
, 0DH, 0AH, '$'
STR_DEVICE_ERR      DB      0DH, 0AH, 'PROGRAM ENDED BY DEVICE
ERROR' , 0DH, 0AH, '$'
STR_INT_END         DB      0DH, 0AH, 'PROGRAM ENDED BY INT 31H' ,
0DH, 0AH, '$'

END_DATA            DB      0
DATA                ENDS

CODE SEGMENT

                    ASSUME    CS:CODE, DS:DATA, SS:SSTACK

FREE_MEMORY         PROC
                        PUSH AX
                        PUSH BX
                        PUSH CX
                        PUSH DX

                        MOV  AX, OFFSET END_DATA
                        MOV  BX, OFFSET __END
                        ADD  BX, AX

                        MOV  CL, 4
                        SHR  BX, CL
                        ADD  BX, 2BH
                        MOV  AH, 4AH
                        INT  21H

                        JNC  _ENDF
                        MOV  MEM_FLAG, 1

                MCB_CRASH:
                        CMP  AX, 7
                        JNE  NOT_ENOUGHT_MEMORY
```

```
                MOV  DX, OFFSET STR_MCB_CRASH_ERR
                PUSH AX
                MOV  AH, 09H
                INT  21H
                POP  AX
                JMP  FREE_
        NOT_ENOUGHT_MEMORY:
                CMP  AX, 8
                JNE  ADDR
                MOV  DX, OFFSET STR_NO_MEM_ERR
                PUSH AX
                MOV  AH, 09H
                INT  21H
                POP  AX
                JMP  FREE_
        ADDR:
                CMP  AX, 9
                MOV  DX, OFFSET STR_ADDR_ERR
                PUSH AX
                MOV  AH, 09H
                INT  21H
                POP  AX
                JMP  FREE_
        _ENDF:
                MOV  MEM_FLAG, 1
                MOV  DX, OFFSET STR_FREE_MEM
                PUSH AX
                MOV  AH, 09H
                INT  21H
                POP  AX

        FREE_:
                POP  DX
                POP  CX
                POP  BX
                POP  AX
                RET
FREE_MEMORY         ENDP
```

```
LOAD                PROC
                        PUSH AX
                        PUSH BX
                        PUSH CX
                        PUSH DX
                        PUSH DS
                        PUSH ES
                        MOV  KEEP_SP, SP
                        MOV  KEEP_SS, SS

                        MOV  AX, DATA
                        MOV  ES, AX
                        MOV  BX, OFFSET PARAMETER_BLOCK
                        MOV  DX, OFFSET CMD_L
                        MOV  [BX+2], DX
                        MOV  [BX+4], DS
                        MOV  DX, OFFSET CL_POS

                        MOV  AX, 4B00H
                        INT  21H

                        MOV  SS, KEEP_SS
                        MOV  SP, KEEP_SP
                        POP  ES
                        POP  DS

                        JNC  LOADS
                FN_ERR:
                        CMP  AX, 1
                        JNE  FILE_ERR
                        MOV  DX, OFFSET STR_FN_ERR
                        PUSH AX
                        MOV  AH, 09H
                        INT  21H
                        POP  AX
                        JMP  LOAD_END
                FILE_ERR:
```

```
            CMP  AX, 2
            JNE  DISK_ERR
            MOV  DX, OFFSET STR_FILE_ERROR
            PUSH AX
            MOV  AH, 09H
            INT  21H
            POP  AX
            JMP  LOAD_END
DISK_ERR:
            CMP  AX, 5
            JNE  MEM_ERR
            MOV  DX, OFFSET STR_DISK_ERR
            PUSH AX
            MOV  AH, 09H
            INT  21H
            POP  AX
            JMP  LOAD_END
MEM_ERR:
            CMP  AX, 8
            JNE  ENVS_ERR
            MOV  DX, OFFSET STR_MEMORY_ERROR
            PUSH AX
            MOV  AH, 09H
            INT  21H
            POP  AX
            JMP  LOAD_END
ENVS_ERR:
            CMP  AX, 10
            JNE  FORMAT_ERR
            MOV  DX, OFFSET STR_ENVS_ERR
            PUSH AX
            MOV  AH, 09H
            INT  21H
            POP  AX
            JMP  LOAD_END
FORMAT_ERR:
            CMP  AX, 11
            MOV  DX, OFFSET STR_FORMAT_ERR
```

```
            PUSH AX
            MOV  AH, 09H
            INT  21H
            POP  AX
            JMP  LOAD_END
LOADS:
            MOV  AH, 4DH
            MOV  AL, 00H
            INT  21H
_NEND:
            CMP  AH, 0
            JNE  CTRLC
            PUSH DI
            MOV  DI, OFFSET STR_NORM_FIN
            MOV  [DI+26], AL
            POP  SI
            MOV  DX, OFFSET STR_NORM_FIN
            PUSH AX
            MOV  AH, 09H
            INT  21H
            POP  AX
            JMP  LOAD_END
CTRLC:
            CMP  AH, 1
            JNE  DEVICE
            MOV  DX, OFFSET STR_CTRL_END
            PUSH AX
            MOV  AH, 09H
            INT  21H
            POP  AX
            JMP  LOAD_END
DEVICE:
            CMP  AH, 2
            JNE  INT_31H
            MOV  DX, OFFSET STR_DEVICE_ERR
            PUSH AX
            MOV  AH, 09H
            INT  21H
```

```
                POP  AX
                JMP  LOAD_END
        INT_31H:
                CMP  AH, 3
                MOV  DX, OFFSET STR_INT_END
                PUSH AX
                MOV  AH, 09H
                INT  21H
                POP  AX
        LOAD_END:
                POP  DX
                POP  CX
                POP  BX
                POP  AX
                RET
LOAD            ENDP

PATH            PROC
                PUSH AX
                PUSH BX
                PUSH CX
                PUSH DX
                PUSH DI
                PUSH SI
                PUSH ES

                MOV  AX, KEEP_PSP
                MOV  ES, AX
                MOV  ES, ES:[2CH]
                MOV  BX, 0

        FINDZ:
                INC  BX
                CMP  BYTE PTR ES:[BX-1], 0
                JNE  FINDZ

                CMP  BYTE PTR ES:[BX+1], 0
                JNE  FINDZ
```

```
                ADD  BX, 2
                MOV  DI, 0

            _LOOP:
                MOV  DL, ES:[BX]
                MOV  BYTE PTR [CL_POS+DI], DL
                INC  DI
                INC  BX
                CMP  DL, 0
                JE        _END_LOOP
                CMP  DL, '\'
                JNE  _LOOP
                MOV  CX, DI
                JMP  _LOOP
            _END_LOOP:
                MOV  DI, CX
                MOV  SI, 0
            _FN:
                MOV  DL, BYTE PTR [PROGRAM+SI]
                MOV  BYTE PTR [CL_POS+DI], DL
                INC  DI
                INC  SI
                CMP  DL, 0
                JNE  _FN

                POP  ES
                POP  SI
                POP  DI
                POP  DX
                POP  CX
                POP  BX
                POP  AX
                RET
PATH            ENDP

MAIN            PROC FAR
                PUSH DS
```

```
                        XOR  AX, AX
                        PUSH AX
                        MOV  AX, DATA
                        MOV  DS, AX
                        MOV  KEEP_PSP, ES
                        CALL FREE_MEMORY
                        CMP  MEM_FLAG, 0
                        JE        _END
                        CALL PATH
                        CALL LOAD
                _END:
                        XOR  AL, AL
                        MOV  AH, 4CH
                        INT  21H

MAIN            ENDP

__END:
CODE            ENDS
                     END  MAIN
```