Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра програмування та захисту інформації

Звіт

з виконаної лабораторної роботи № 6

дисципліни “ Технології розробки алгоритмів ”

на тему

“ Дерева. Видалення елементів з бінарного дерева. Невузлове представлення бінарних дерев, збалансовані бінарні дерева, дерево з випадковим пошуком ”

Виконав

студент академічної

групи КІ-15

Аннаев А

Перевірив

Викл.

Гермак В.С.

Кропивницкий- 2017

Лабораторная работа №1

**Тема**: Дескрипторы и таблицы глобальных дескрипторов.

**Цель**: Научиться программно описывать дескрипторы в таблице глобальных дескрипторов и загружать в регистр процессора GDTR, информацию о таблице глобальных дескрипторов.

ЗАДАНИЕ

Разработать и отладить программу, которая:

- описывает таблицу глобальных дескрипторов;

- описывает дескрипторы сегмента данных, сегмента команд, сегмента стека программы;

- инициализирует дескрипторы, загружает в регистр процессора GDTR данные о таблицу глобальных дескрипторов и переводит ЦБ в защищенный режим;

- выполняет 10N итераций цикла, где N - порядковый номер студента в списке журнала, и возвращает процессор в реальный режим.

Номер варианта 8

.386P ; Разрешение трансляции привилегированных команд

                                      ; Структура для описания дескрипторiв сегментовdescr

struc

limit dw 0 ; Предел (биты 0..15)

base\_l dw 0 ; База, биты 0..15

base\_m db 0 ; База, биты 16..23

attr\_1 db 0 ; Байт атрибутов 1

attr\_2 db 0 ; Байт атрибутов 2

base\_h db 0 ; База, биты 24..31

descr ends

; Сегмент данных 16-разрядная программа

data segment use16

; Таблица глобальных дескрипторов GDT

gdt\_null descr <0,0,0,0,0,0> ; Селектор 0 - обязательный; нулевой дескриптор

gdt\_data descr <data\_size-1,0,0,92h, 0,0> ; Селектор 8, сегмент данных

gdt\_code descr <code\_size-1,0,0,98h, 0,0> ; Селектор 16 сегмент команд

gdt\_stack descr <255,0,0,92h, 0,0> ; Селектор 24, сегмент стека

gdt\_size = $ - gdt\_null ; Размер GDT

pdescr dq 0 ; псевдодескриптором для lgdt

real\_sp dw 0 ; переменная для сохранения SP

data\_size = $ - gdt\_null ; Размер сегмента данных

data ends ; Конец сегмента данных

; Сегмент команд

text segment 'code' use16, 16-разрядная программа

         assume CS: text, DS: data

main proc

         mov AX, data ; инициализации DS

         mov DS, AX ; в реальном режиме

; Найдем 32-бiтову линейную адрес сегмента данных

         mov DL, 0

         shld DX, AX, 4

         shl AX, 4

         mov BX, offset gdt\_data

         mov [BX]. base\_l, AX

         mov [BX]. base\_m, DL

; Найдем 32-бiтову линейную адрес сегмента команд

         mov AX, CS

         mov DL, 0

         shld DX, AX, 4

         shl AX, 4

         mov BX, offset gdt\_code

         mov [BX]. base\_l, AX

         mov [BX]. base\_m, DL

; Найдем 32-битную адресную линейнию сегмента стека

         mov AX, SS

         mov DL, 0

         shld DX, AX, 4

         shl AX, 4

         mov BX, offset gdt\_stack

         mov [BX]. base\_l, AX

         mov [BX]. base\_m, DL

; Подготовит дескритор и загрузим регистр GDTR

         mov BX, offset gdt\_data ; Адрес GDT

         mov AX, [BX] .base\_l ; Получим и занесем в pdescr

         mov word ptr pdescr + 2, AX ; базу, биты 0..15

         mov DL, [BX]. base\_m ; Получим i занесем в pdescr

         mov byte ptr pdescr + 4, DL ; базу, биты 16..23

         mov word ptr pdescr, gdt\_size-1 ; Предел GDT

         lgdt pdescr ; Загрузим регистр GDTR

 ; Подготовка к возвращению в реальный режим

         mov AX, 40h ; Настроим ES на область

         mov ES, AX ; данных BIOS

  mov word ptr ES: [67h], offset return ; Смещение точки возврата

         mov ES: [69h], CS ; Сегмент точки возврата

; Подготовка переходу в защищенный режим

         cli ; Запрет аппаратных прерываний

         mov AL, 0Fh ; выборка байта состояния отключения

         out 70h, AL ; Порт КМОП-микросхемы

         mov AL, 0Ah ; Установка режима восстановления

out 71h, AL ; после сброса процессора

; Переход в защищенный режим

         smsw AX ; Получим слово состояния компьютера

         or AX, 1 ; Установим бит PE

         lmsw AX ; Запишем слово состояния

; Процессор работает в защищенном режиме

         mov CX, 800; Количество итераций цикла

m1: mov BX, AX

         loop m1 ; Цикл

; Вернемся в реальный режим

         mov real\_sp, SP ; Сохраним SP

         mov AL, 0FEh ; Команда сброса процессора

         out 64h, AL ; в порт 64h

         hlt ; останавливает процессор до конца сброса

; Процессор работает в реальном режиме

; Возобновит Операционное среду реального режима

return: mov AX, data ; возобновит адресуемисть данных

         mov DS, AX

         mov SP, real\_sp ; возобновит адресуемисть стеке

         mov AX, stk

         mov SS, AX

         sti ; Позволим аппаратные прерывания

; Выполняем проверку функций DOS после возвращения в

; реальный режим

         mov AH, 09h ; Функция вывода на экран строки

         mov DX, offset mes ; Адрес строки

         int 21h

         mov AX, 4C00h; Завершим программу обычным образом

         int 21h

main endp ; Конец главной процедуры

code\_size = $ - main ; Размер сегмента команд

text ends ; Конец сегмента команд

stk segment stack 'stack' ; Начало сегмента стека

         db 256 dup ( '^')

stk ends ; Конец сегмента стека

         end main ; Конец программы

Контрольные вопросы:

1. Какие возможности архитектуры МП реализуются при переходе в защищенный режим?

В защищенном режиме процессор может адресовать до 16 Мб физической и 1 Гб виртуальной памяти, за счет изменения механизма адресации. Это делает возможным хранение в памяти только необходимой на данный момент части программы, а другая часть может храниться во внешней памяти.

2. Что такое дескриптор сегмента?

Служебная структура в памяти, описывает сегмент.

3. Какую информацию хранит дескриптор сегмента?

Длина дескриптора 8 байт.

 База (32 бита) - начало сегмента в линейной памяти.

Лимит (20 бит) - (размер сегенту) -1 (База + Лимит = линейный адрес последнего байта)

Права доступа (12 бит) - флаги, определяющие наличие сегмента в памяти, уровень защиты, тип, разрядность + один пользовательский флаг.

4. Как формируется линейный адрес в защищенном режиме?

Процессор по номеру сегмента дескриптора находит нужный дескриптор, извлекает из него базовый адрес сегмента, увеличивает ее на указанное в конкретной команде смещение (относительный адрес), формирует адрес ячейки памяти.

5. Что такое таблица глобальных дескрипторов?

Таблица с дескрипторами сегментов, используемых программой.

6. Можно ли модифицировать сегменты команд в защищенном режиме во время выполнения программы?

ни

7. Что такое регистр таблицы глобальных дескрипторов?

Специальный 48-битный регистир, что описывает местонахождение и размер таблицы.

8. Какие таблицы следует подготовить для выхода в защищенный режим?

Таблицу глобальных дескрипторов

9. Каким образом можно вернуться с защищенного режима в реальный?

Выполнить программное сброса процессора.

10. Каково назначение теневых регистров? Доступны они программисту?

Теневые регистры недоступны программисту; они автоматически загружаются процессором с таблицы дескрипторов каждый раз, когда процессор загружает соответствующий сегментный регистр. Таким образом, в защищенном режиме программист имеет дело с селекторами, т.е. номерами дескрипторов, а процессор - с самими дескрипторами, хранящихся в теневых регистрах. Именно содержание теневого регистра (в первую очередь, линейный адрес сегмента) определяет область памяти, к которой обращается процессор при выполнении конкретной команды.