МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ

УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения

вычислительной техники и автоматизированных

систем

**Лабораторная работа №1**

по дисциплине: Архитектура вычислительных систем

тема: «**Разработка программ на ассемблере.**

**Работа с отладчиком x32dbg, пакетом masm32**»

Выполнил: студент группы ВТ-231

Масленников Д. А.

Проверили:  
Осипов О.В.

Бондаренко Т.В.

Белгород 2025

**Цель работы:** получить навыки создания простейших ассемблерных программ с использованием пакета masm32 и научиться пользоваться отладчиком x32dbg.

**Примечание**: на лекции было сказано о том, что при выполнении лабораторной работы можно использовать операционную систему линукс. Поэтому, так как я привык работать именно в ней, лабораторная будет выполнена с ипользованием возможностей этой системы — в частности компилятор nasm, а так же отладчик gdb

### Задание по варианту №14:

Исходный код на MASM32:

.DATA

String1 DB 13, 10, "\_\_\_\_", 0

ds DW 5

result DD ?

p DQ 17.5

ten DT 183333.5

.CODE

START:

XOR ECX, ECX

MOV CX, ds

ADD CX, 5

MOV EAX, ECX

DIV 2

MOV result, EAX

END START

### Исходный код программы на NASM для Linux:

section .data ; секция инициализированных данных

String1 db 13, 10, "\_\_\_\_", 0 ; строка с символами возврата каретки(13) и перевода строки(10)

data\_size dw 5 ; переменная размером 2 байта со значением 5 (аналог ds из MASM32)

align 4 ; выравнивание следующей метки на границу 4 байта

result dd 0 ; переменная размером 4 байта для хранения результата (инициализирована 0)

p dq 17.5 ; переменная размером 8 байт со значением 17.5

ten dt 183333.5 ; переменная размером 10 байт со значением 183333.5

section .text ; секция кода

global \_start ; объявление точки входа для линкера

\_start: ; метка начала программы

xor ecx, ecx ; обнуление регистра ecx

movzx ecx, word [data\_size] ; загрузка значения из data\_size в регистр ecx с расширением нулём

add ecx, 5 ; увеличение значения в регистре ecx на 5

mov eax, ecx ; копирование значения из ecx в eax (для операции деления)

mov edx, 0 ; обнуление регистра edx (обязательно для операции деления)

mov ebx, 2 ; загрузка значения 2 в регистр ebx (делитель)

div ebx ; деление eax на ebx, результат в eax, остаток в edx

mov [result], eax ; сохранение результата деления в переменную result

mov eax, 4 ; системный вызов write (4)

mov ebx, 1 ; файловый дескриптор stdout (1)

mov ecx, String1 ; указатель на строку

mov edx, 6 ; длина строки (6 байт: CR, LF + 4 символа)

int 0x80 ; вызов прерывания

mov eax, 1 ; номер системного вызова exit

xor ebx, ebx ; код возврата 0 (успешное завершение)

int 0x80 ; вызов прерывания для завершения программы

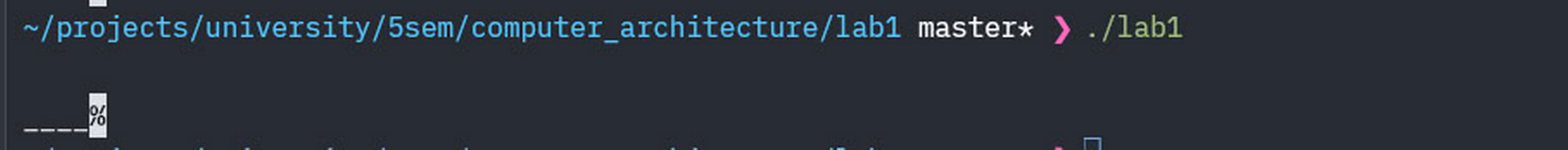
### Компиляция и запуск:

Для компиляции и запуска программы на Linux с использованием NASM и LD используются следующие команды bash:

nasm -f elf32 -g -F dwarf lab1.asm -o lab1.o

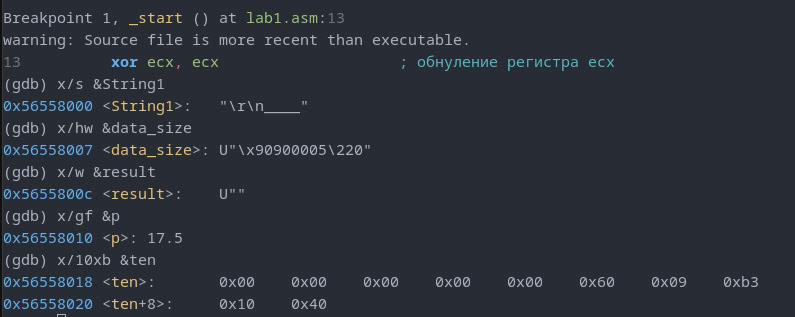
ld -m elf\_i386 lab1.o -o lab1

./lab1

**Результат выполнения**

Код выводит строку согласно одному из подпунктов пункта 4

### Анализ сегмента данных с помощью GDB:

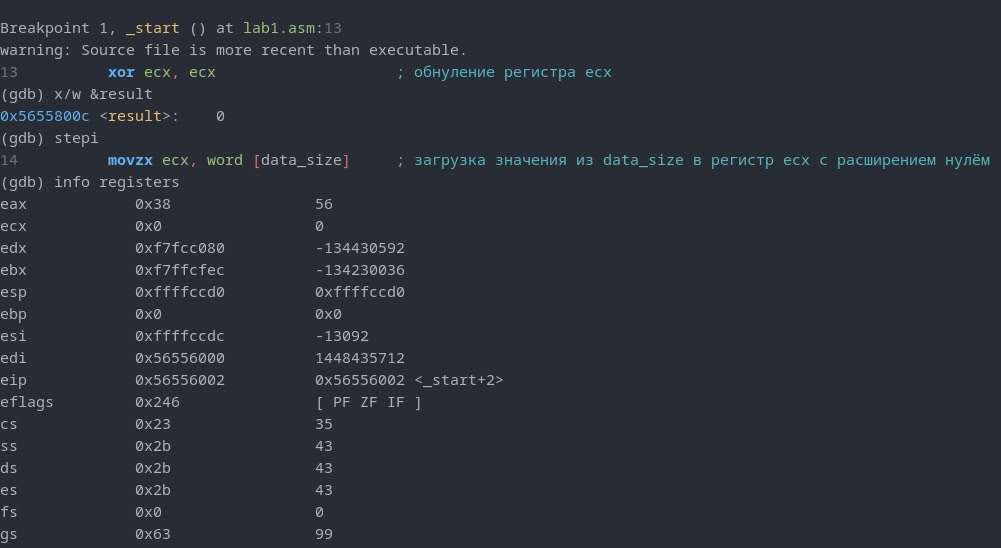
  
Описание:  
Мы переходим на \_start программы, где у нас уже записались все данные. После этого мы просматриваем все переменные, получая информацию о сегменте данных и значении. При получении этой информации стоит не забывать использовать соответствующий заданному типу атрибут

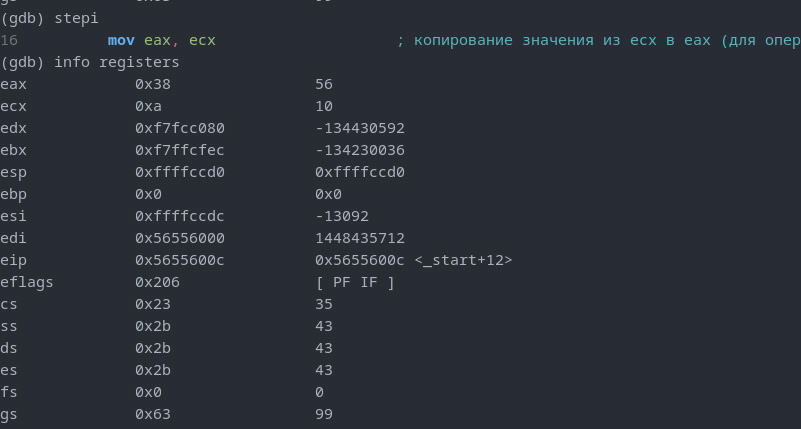
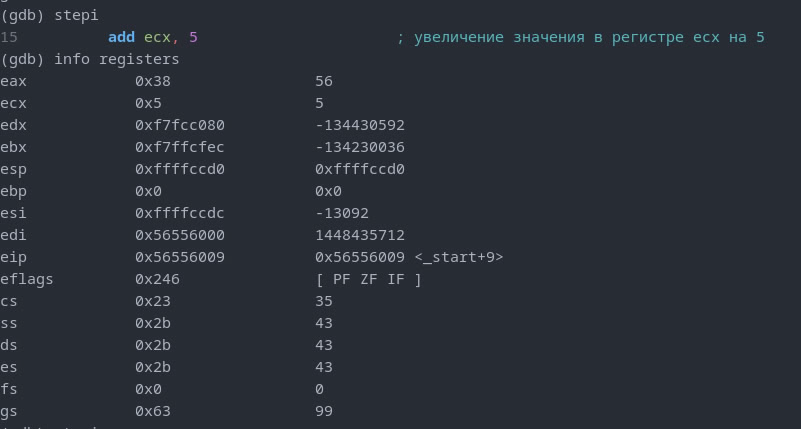
Результаты анализа:

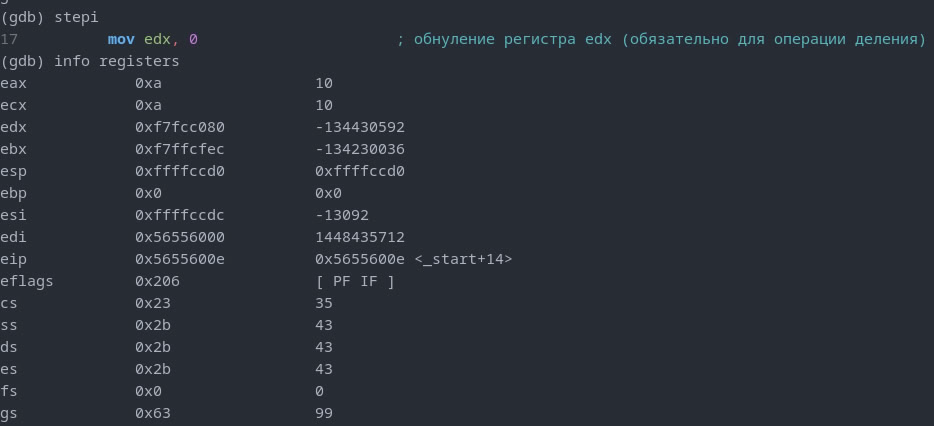
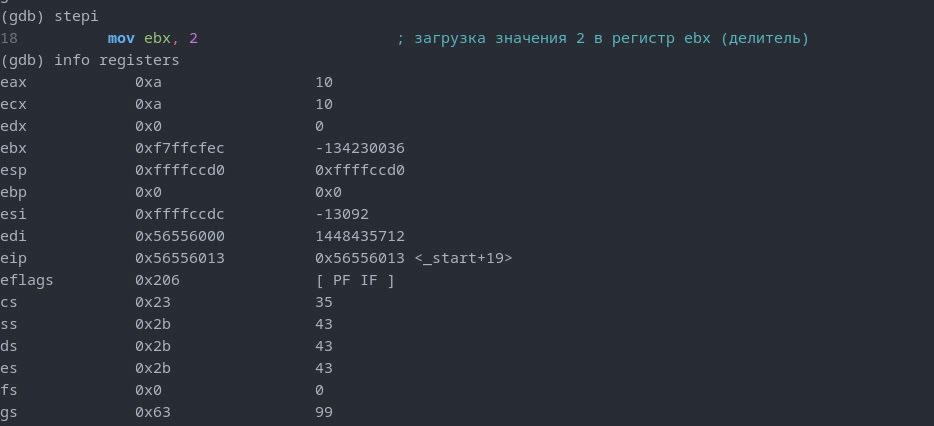
1. **Строка String1**
   * Адрес: 0x56558000
   * Содержимое: 0x0d 0x0a 0x5f 0x5f 0x5f 0x5f 0x00
   * Размер: 7 байт
   * ASCII: "\r\n\_\_\_\_\0"
2. **Переменная data\_size**
   * Адрес: 0x56558007
   * Значение: 5 (0x0005)
   * Размер: 2 байта
3. **Переменная result (до вычислений)**
   * Адрес: 0x5655800c
   * Значение: 0 (0x00000000)
   * Размер: 4 байта
4. **Переменная p**
   * Адрес: 0x56558010
   * Значение: 17.5 (в формате double)
   * Байтовое представление: 0x00 0x00 0x00 0x00 0x00 0x00 0x31 0x40
   * Размер: 8 байт
5. **Переменная ten**
   * Адрес: 0x56558018
   * Значение: 183333.5 (в формате extended precision)
   * Байтовое представление: 0x00 0x00 0x00 0x00 0x00 0x6e 0xb2 0x40 0x00 0x40
   * Размер: 10 байт

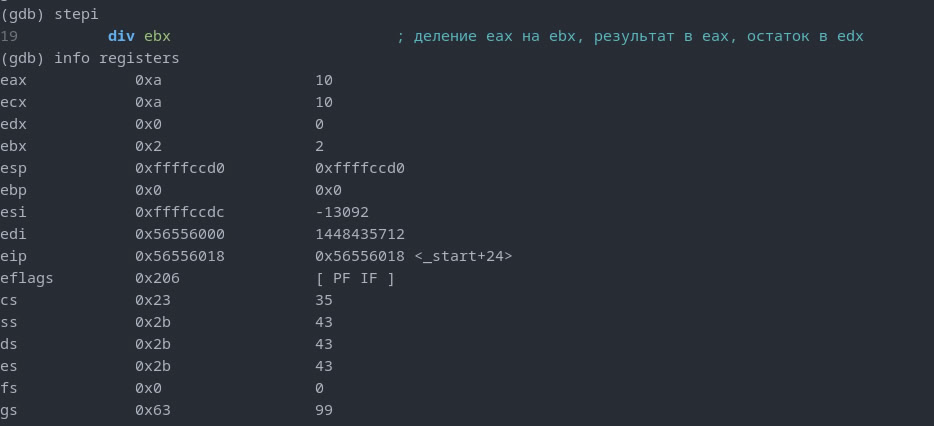
### Пошаговая трассировка программы:

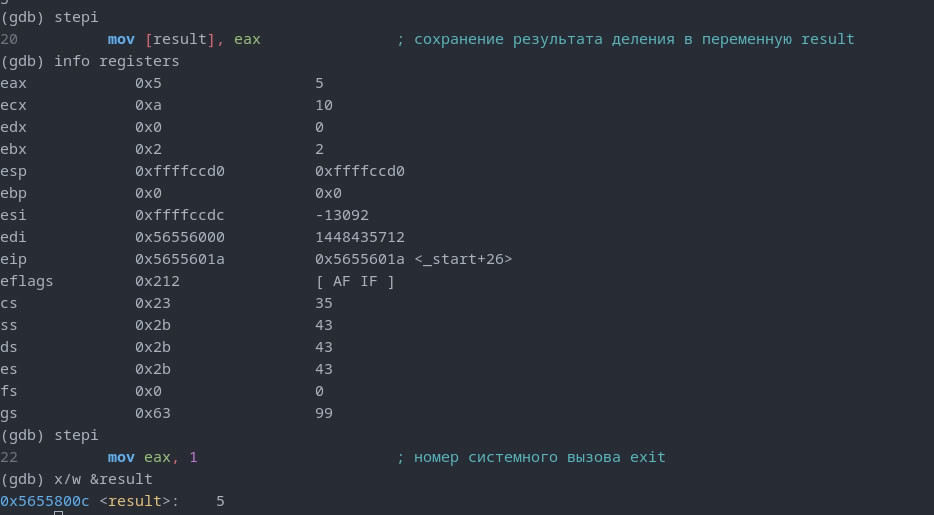
Запускаем программу в GDB и пошагово выполняем инструкции, наблюдая за изменениями регистров и памяти.







  
Программа завершается с кодом 0.

### Выводы:

В ходе лабораторной работы была написана программа на ассемблере NASM для платформы Linux, которая выполняет арифметические операции (сложение и деление) и сохраняет результат в переменную. Программа была успешно скомпилирована и запущена. С помощью отладчика GDB был проведен анализ сегмента данных и пошаговая трассировка программы, в ходе которой было подтверждено корректное выполнение всех инструкций. Получены навыки работы с отладчиком и анализа памяти.