

# **Лабораторная работа 9**

**Понятие подпрограммы. Отладчик**

Елисейкина Надежда Михайловна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>6</b>
<b>2</b>	<b>Задание</b>	<b>7</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>9</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>14</b>
<b>5</b>	<b>Выводы</b>	<b>46</b>
	<b>Список литературы</b>	<b>47</b>

# Список иллюстраций

4.1	Создание каталога для выполнения лабораторной работы . . . . .	14
4.2	Листинг 1 . . . . .	15
4.3	Результаты работы программы из листинга 1 . . . . .	16
4.4	Листинг1 с изменениями . . . . .	17
4.5	Результаты работы программы из листинга 1 с изменениями . . .	18
4.6	Листинг 2 . . . . .	19
4.7	Отладка программы из Листинга 2 . . . . .	20
4.8	Проверка работы программы с помощью команды run . . . . .	21
4.9	Брейкпоинт на метку _start . . . . .	22
4.10	Дисассимилированный код программы . . . . .	23
4.11	Отображение команд с Intel'овским синтаксисом . . . . .	24
4.12	Режим псевдографики . . . . .	25
4.13	Установка точки останова по адресу инструкции . . . . .	26
4.14	Просмотр регистров . . . . .	27
4.15	Измененные регистры . . . . .	27
4.16	Значение переменной msg1 . . . . .	28
4.17	Значение переменной msg2 . . . . .	29
4.18	Измененное значение переменной msg1 . . . . .	29
4.19	Измененное значение переменной msg2 . . . . .	30
4.20	Значение регистра edx в различных форматах . . . . .	30
4.21	Значение регистра ebx . . . . .	31
4.22	Завершение работы с файлов . . . . .	32
4.23	Создание исполняемого файла lab09-3.asm . . . . .	32
4.24	Запуск файла в отладчике . . . . .	33
4.25	Точка останова перед первой инструкцией в программе . . . . .	34
4.26	Адрес вершины стека . . . . .	35
4.27	Все позиции стека . . . . .	36
4.28	Листинг 1 самостоятельного задания №1 . . . . .	37
4.29	Результат работы программы . . . . .	38
4.30	Листинг из самостоятельного задания №2 . . . . .	39
4.31	Проверка результата работы программы из Листинга самостоятель- ного задания №2 . . . . .	40
4.32	Запуск программы в отладчике . . . . .	41
4.33	Анализ регистров . . . . .	42
4.34	Повторный запуск программы . . . . .	43
4.35	Листинг программы . . . . .	44

4.36 Проверка результата работы программы . . . . .	45
---	----

## **Список таблиц**

# 1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

## 2 Задание

1. Создать каталог для выполнения лабораторной работы № 9, перейти в него и создать файл lab09-1.asm.
2. Ввести в файл lab09-1.asm текст программы из листинга 1 методического указания. Создать исполняемый файл и проверить его работу.
3. Изменить текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения  $f(g(x))$ , где  $x$  вводится с клавиатуры,  $f(x) = 2x + 7$ ,  $g(x) = 3x + 1$ . Т.е.  $x$  передается в подпрограмму `_calcul` из нее в подпрограмму `_subcalcul`, где вычисляется выражение  $g(x)$ , результат возвращается в `_calcul` и вычисляется выражение  $f(g(x))$ . Результат возвращается в основную программу для вывода результата на экран.
4. Создать файл lab09-2.asm с текстом программы из Листинга 2 методического указания.
5. Получить исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию. Загрузить исполняемый файл в отладчик.
6. Проверить работу программы, запустив ее в оболочке GDB.
7. Установить брейкпоинт на метку `_start` и запустить её.
8. Посмотреть дисассимилированный код программы.
9. Переключить на отображение команд с Intel'овским синтаксисом.

10. Включить режим псевдографики.
11. Установить несколько точек останова.
12. Выполнить 5 инструкций с помощью команды `si`.
13. Посмотреть значение переменных `msg1` и `msg2`.
14. Изменить значение переменных `msg1` и `msg2`.
15. Вывести в различных форматах значение регистра `edx`.
16. Изменить значение регистра `ebx`.
17. Скопировать файл `lab8-2.asm`, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки в файл с именем `lab09-3.asm`. Создать исполняемый файл.
18. Загрузить исполняемый файл в отладчик, указав аргументы.
19. Исследовать расположение аргументов командной строки в стеке после запуска программы.
20. Проверить адрес вершины стека и посмотреть все позиции стека.

#### Задание для самостоятельной работы

1. Преобразовать программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции  $f(x)$  как подпрограмму.
2. В листинге из самостоятельного задания приведена программа вычисления выражения  $(3 + 2) * 4 + 5$ . При запуске данная программа дает неверный результат. Проверить это. С помощью отладчика GDB, анализируя изменения значений регистров, определить ошибку и исправить ее.



## 3 Теоретическое введение

Понятие об отладке

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

- обнаружение ошибки;
- поиск её местонахождения;
- определение причины ошибки;
- исправление ошибки.

Можно выделить следующие типы ошибок:

синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка;

семантические ошибки — являются логическими и приводят к тому, что программа запускается, отработывает, но не даёт желаемого результата;

ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы.

Последний этап — исправление ошибки. После этого при повторном запуске

программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

Наиболее часто применяют следующие методы отладки:

создание точек контроля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообщения);

использование специальных программ-отладчиков.

Отладчики позволяют управлять ходом выполнения программы, контролировать и изменять данные. Это помогает быстрее найти место ошибки в программе и ускорить её исправление. Наиболее популярные способы работы с отладчиком — это использование точек останова и выполнение программы по шагам.

Пошаговое выполнение — это выполнение программы с остановкой после каждой строки, чтобы программист мог проверить значения переменных и выполнить другие действия.

Точки останова — это специально отмеченные места в программе, в которых программа-отладчик приостанавливает выполнение программы и ждёт команд. Наиболее популярные виды точек останова:

Breakpoint — точка останова (остановка происходит, когда выполнение доходит до определённой строки, адреса или процедуры, отмеченной программистом);

Watchpoint — точка просмотра (выполнение программы приостанавливается, если программа обратилась к определённой переменной: либо считала её значение, либо изменила его).

Точки останова устанавливаются в отладчике на время сеанса работы с кодом программы, т.е. они сохраняются до выхода из программы-отладчика или до смены отлаживаемой программы.

Основные возможности отладчика GDB

GDB (GNU Debugger — отладчик проекта GNU) работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования. GDB предлагает обширные средства для слежения и контроля за

выполнением компьютерных программ. Отладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторонних графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его в качестве базовой подсистемы отладки. Отладчик GDB (как и любой другой отладчик) позволяет увидеть, что происходит «внутри» программы в момент её выполнения или что делает программа в момент сбоя. GDB может выполнять следующие действия:

- начать выполнение программы, задав всё, что может повлиять на её поведение;
- остановить программу при указанных условиях;
- исследовать, что случилось, когда программа остановилась;
- изменить программу так, чтобы можно было поэкспериментировать с устранением эффектов одной ошибки и продолжить выявление других.

#### Дизассемблирование программы

Если есть файл с исходным текстом программы, а в исполняемый файл включена информация о номерах строк исходного кода, то программу можно отлаживать, работая в отладчике непосредственно с её исходным текстом. Чтобы программе можно было отлаживать на уровне строк исходного кода, она должна быть откомпилирована с ключом `-g`.

Существует два режима отображения синтаксиса машинных команд: режим Intel, используемый в том числе в NASM, и режим AT&T (значительно отличающийся внешне). По умолчанию в дизассемблере GDB принят режим AT&T. Переключиться на отображение команд с привычным Intel'овским синтаксисом можно, введя команду `set disassembly-flavor intel`.

#### Точки останова

Установить точку останова можно командой `break` (кратко `b`). Типичный аргумент этой команды — место установки. Его можно задать как имя метки или как адрес. Чтобы не было путаницы с номерами, перед адресом ставится «звёздочка».

Информацию о всех установленных точках останова можно вывести командой

info (кратко i).

Для того чтобы сделать неактивной какую-нибудь ненужную точку останова, можно воспользоваться командой `disable`.

Обратно точка останова активируется командой `enable`.

Если же точка останова в дальнейшем больше не нужна, она может быть удалена с помощью команды `delete`.

Пошаговая отладка

Для продолжения остановленной программы используется команда `continue`. Выполнение программы будет происходить до следующей точки останова. В качестве аргумента может использоваться целое число `N`, которое указывает отладчику проигнорировать `N - 1` точку останова (выполнение остановится на `N`-й точке).

Команда `stepi` (кратко `sl`) позволяет выполнять программу по шагам, т.е. данная команда выполняет ровно одну инструкцию.

При указании в качестве аргумента целого числа `N` отладчик выполнит команду `step N` раз при условии, что не будет точек останова или выполнение программы не прервётся по другим причинам.

Команда `nexti` (или `ni`) аналогична `stepi`, но вызов процедуры (функции) трактуется отладчиком как одна инструкция.

Работа с данными программы в GDB

Как уже упоминалось, отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Посмотреть содержимое регистров можно с помощью команды `info registers` (или `i r`).

Изменить значение для регистра или ячейки памяти можно с помощью команды `set`, задав ей в качестве аргумента имя регистра или адрес. При этом перед именем регистра ставится префикс `$`, а перед адресом нужно указать в фигурных скобках тип данных (размер сохраняемого значения; в качестве типа данных можно использовать типы языка Си).

### Понятие подпрограммы

Подпрограмма — это, как правило, функционально законченный участок кода, который можно многократно вызывать из разных мест программы. В отличие от простых переходов из подпрограмм существует возврат на команду, следующую за вызовом. Если в программе встречается одинаковый участок кода, его можно оформить в виде подпрограммы, а во всех нужных местах поставить её вызов. При этом подпрограмма будет содержаться в коде в одном экземпляре, что позволит уменьшить размер кода всей программы.

### Инструкция call и инструкция get

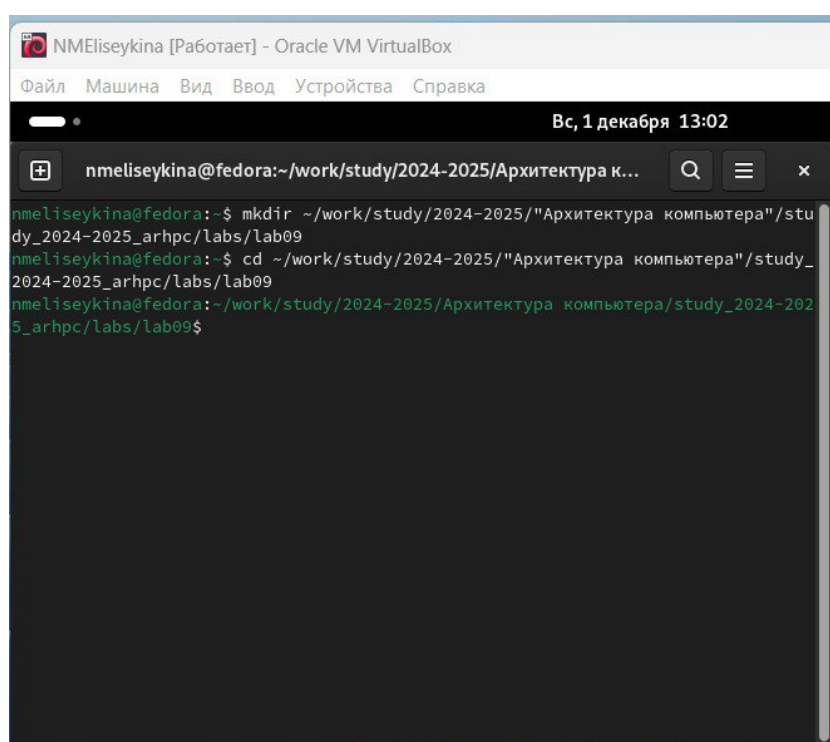
Для вызова подпрограммы из основной программы используется инструкция call, которая заносит адрес следующей инструкции в стек и загружает в регистр eip адрес соответствующей подпрограммы, осуществляя таким образом переход. Затем начинается выполнение подпрограммы, которая, в свою очередь, также может содержать подпрограммы.

Подпрограмма завершается инструкцией get, которая извлекает из стека адрес, занесённый туда соответствующей инструкцией call, и заносит его в eip. После этого выполнение основной программы возобновится с инструкции, следующей за инструкцией call.

Подпрограмма может вызываться как из внешнего файла, так и быть частью основной программы.

## 4 Выполнение лабораторной работы

1. Создали каталог для выполнения лабораторной работы № 9, создали файл lab09-1.asm (рис. 4.1 Создание каталога для выполнения лабораторной работы).



```
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
nmeliseykina@fedora:~$ mkdir ~/work/study/2024-2025/"Архитектура компьютера"/study_2024-2025_arhpc/labs/lab09
nmeliseykina@fedora:~$ cd ~/work/study/2024-2025/"Архитектура компьютера"/study_2024-2025_arhpc/labs/lab09
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$
```

Рис. 4.1: Создание каталога для выполнения лабораторной работы

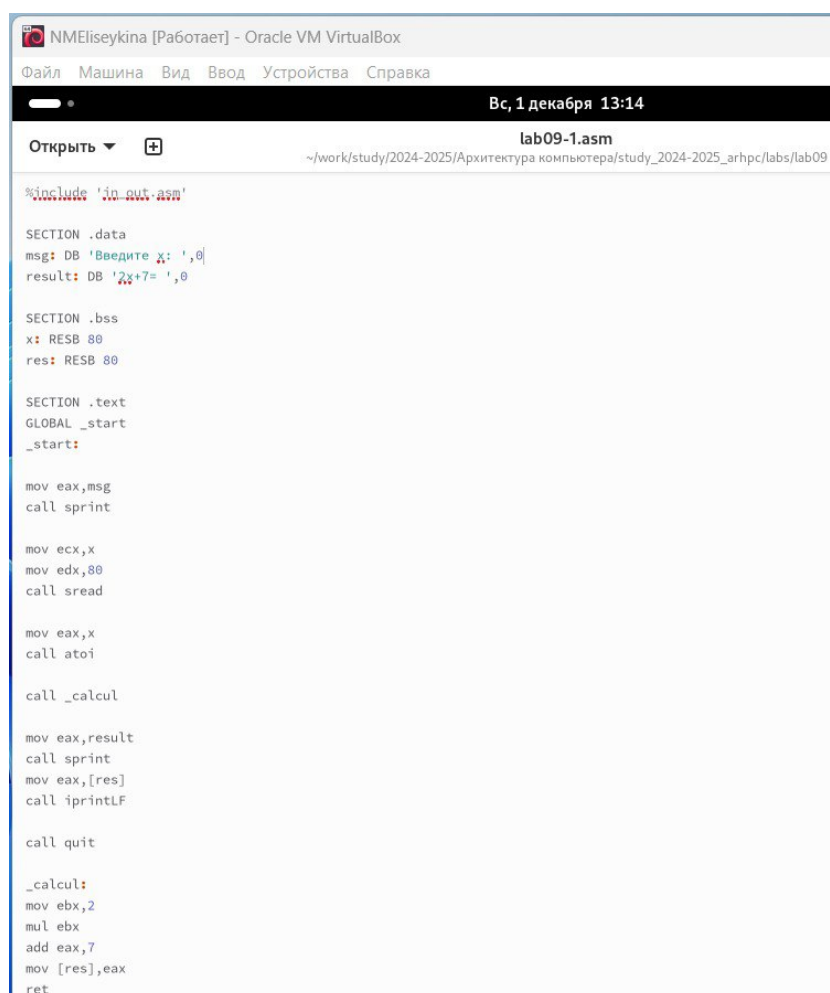
2. В качестве примера рассмотрели программу вычисления арифметического выражения  $f(x) = 2x + 7$  с помощью подпрограммы `_calcul`. (рис 4.2 Листинг1).  
Первые строки программы отвечают за вывод сообщения на экран (call `sprint`),

чтение данных введенных с клавиатуры (call sread) и преобразования введенных данных из символьного вида в численный (call atoi).

Инструкция get является последней в подпрограмме и ее исполнение приводит к возвращению в основную программу к инструкции, следующей за инструкцией call, которая вызвала данную подпрограмму.

Последние строки программы реализуют вывод сообщения (call sprint), результата вычисления (call iprintLF) и завершение программы (call quit).

Ввели в файл lab09-1.asm текст программы из листинга 1 методического указания. Создали исполняемый файл и проверили его работу (рис. 4.3 Результаты работы программы из листинга 1).



```
%include 'in-out.asm'

SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7= ',0

SECTION .bss
x: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,msg
call sprint

mov ecx,x
mov edx,80
call sread

mov eax,x
call atoi

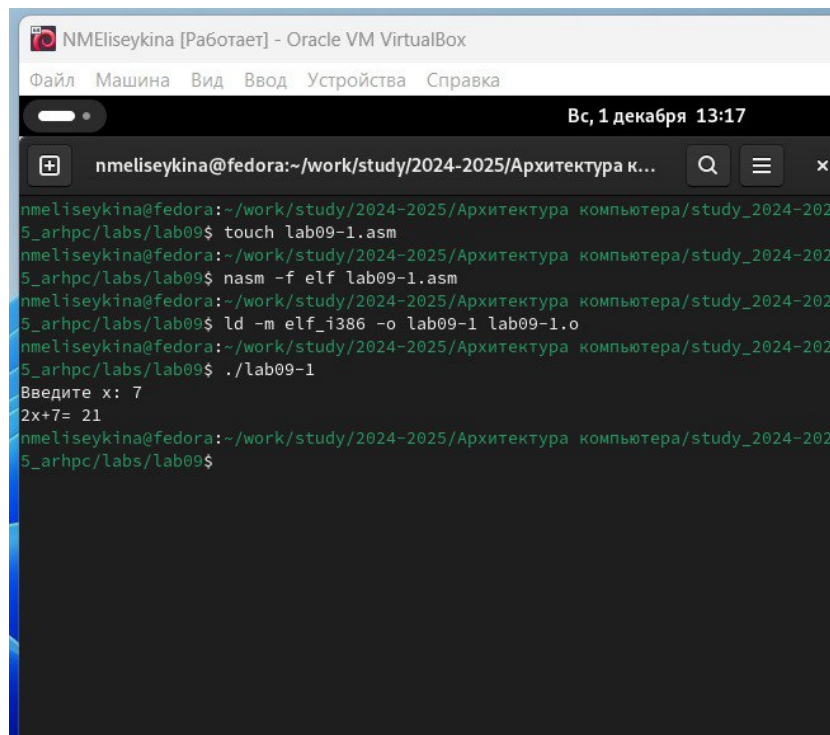
call _calcul

mov eax,result
call sprint
mov eax,[res]
call iprintLF

call quit

_calcul:
mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret
```

Рис. 4.2: Листинг 1



```
nmeliseykina [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Вс, 1 декабря 13:17
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ touch lab09-1.asm
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ nasm -f elf lab09-1.asm
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ./lab09-1
Введите x: 7
2x+7= 21
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$
```

Рис. 4.3: Результаты работы программы из листинга 1

Изменили текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения  $f(g(x))$ , где  $x$  вводится с клавиатуры,  $f(x) = 2x + 7$ ,  $g(x) = 3x + 1$ . Т.е.  $x$  передается в подпрограмму `_calcul` из нее в подпрограмму `_subcalcul`, где вычисляется выражение  $g(x)$ , результат возвращается в `_calcul` и вычисляется выражение  $f(g(x))$ . Результат возвращается в основную программу для вывода результата на экран.



Рис. 4.4: Листинг1 с изменениями

```
nmeliseykina [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Вс, 1 декабря 13:20
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ touch lab09-1.asm
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ nasm -f elf lab09-1.asm
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ./lab09-1
Введите x: 7
2x+7= 21
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ nasm -f elf lab09-1.asm
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ./lab09-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 5
f(g(x))= 35
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$
```

Рис. 4.5: Результаты работы программы из листинга 1 с изменениями

3. Создали файл lab09-2.asm с текстом программы из Листинга 2 (рис. 4.6 Листинг 2).

```
SECTION .data
msg1: db "Hello, ",0x0
msg1len: equ $ - msg1
msg2: db "world!",0x0
msg2len: equ $ - msg2

SECTION .text
global _start
_start:

mov eax, 4
mov ebx, 1
mov ecx, msg1
mov edx, msg1len

int 0x80

mov eax, 4
mov ebx, 1
mov ecx, msg2
mov edx, msg2len

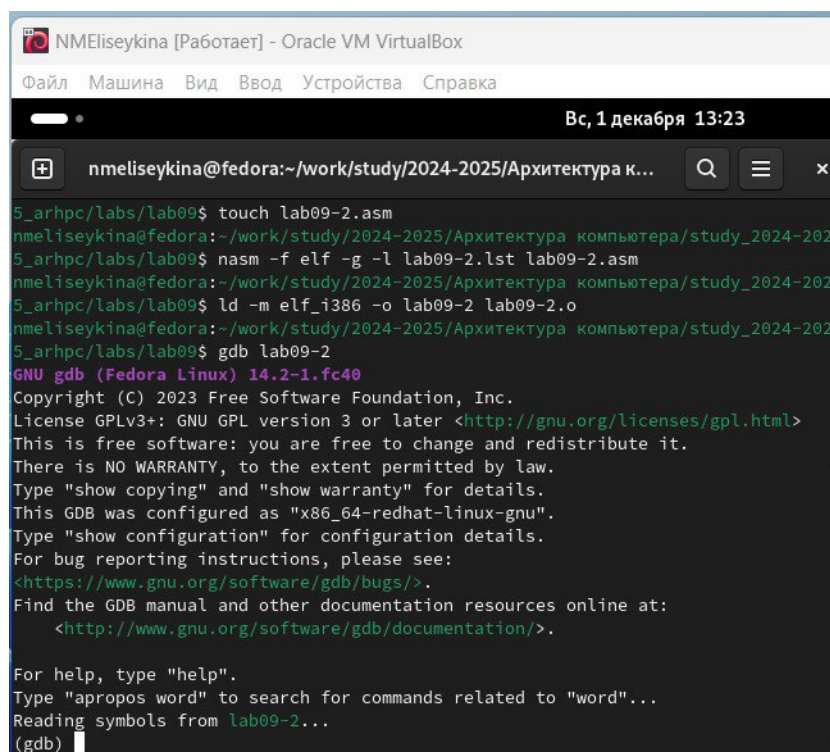
int 0x80

mov eax, 1
mov ebx, 0

int 0x80
```

Рис. 4.6: Листинг 2

Получили исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’ (рис. 4.7 Отладка программы из Листинга 2).



```
nmeliseykina [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка

Вс, 1 декабря 13:23

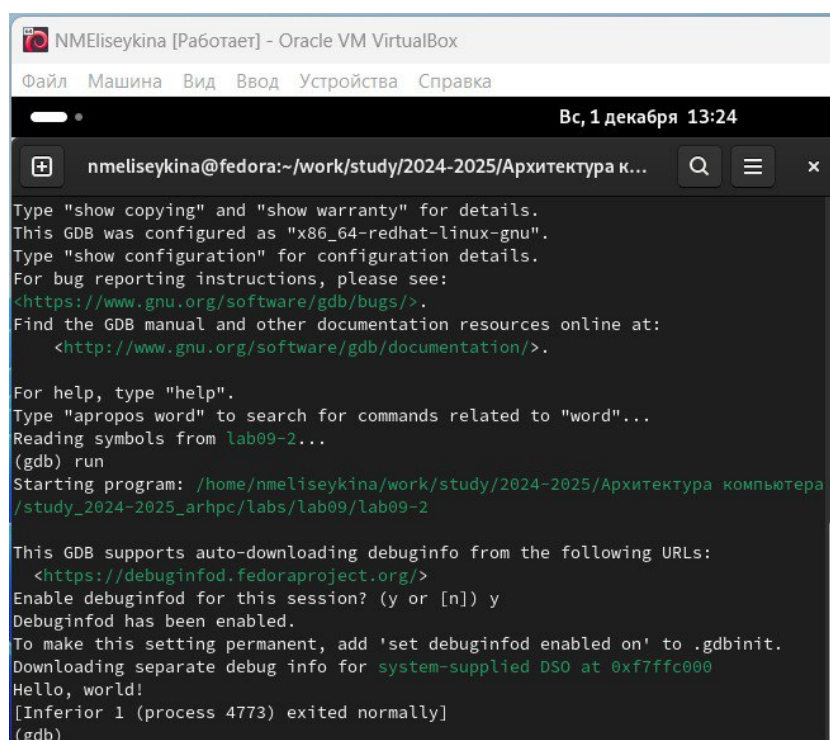
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...

5_arhpc/labs/lab09$ touch lab09-2.asm
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025$ ld -m elf_i386 -o lab09-2 lab09-2.o
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025$ gdb lab09-2
GNU gdb (Fedora Linux) 14.2-1.fc40
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) 
```

Рис. 4.7: Отладка программы из Листинга 2

Проверили работу программы, запустив ее в оболочке GDB с помощью команды `run` (рис. 4.8 Проверка работы программы с помощью команды `run`).



The screenshot shows a terminal window titled "nmeliseykina [Работает] - Oracle VM VirtualBox". The terminal displays the GDB prompt and the execution of a program. The output includes GDB version information, configuration details, and the program's output "Hello, world!".

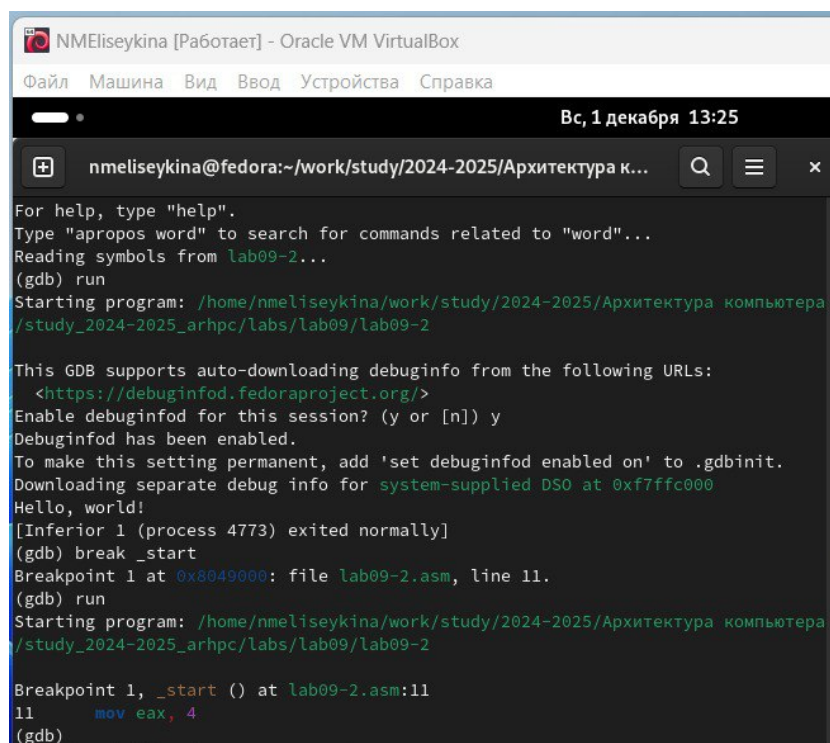
```
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) run
Starting program: /home/nmeliseykina/work/study/2024-2025/Архитектура компьютера
/study_2024-2025_arhpc/labs/lab09/lab09-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 4773) exited normally]
(gdb)
```

Рис. 4.8: Проверка работы программы с помощью команды run

Для более подробного анализа программы установили брейкпоинт на метку `_start` и запустили её (рис. 4.9 Брейкпоинт на метку `_start`).



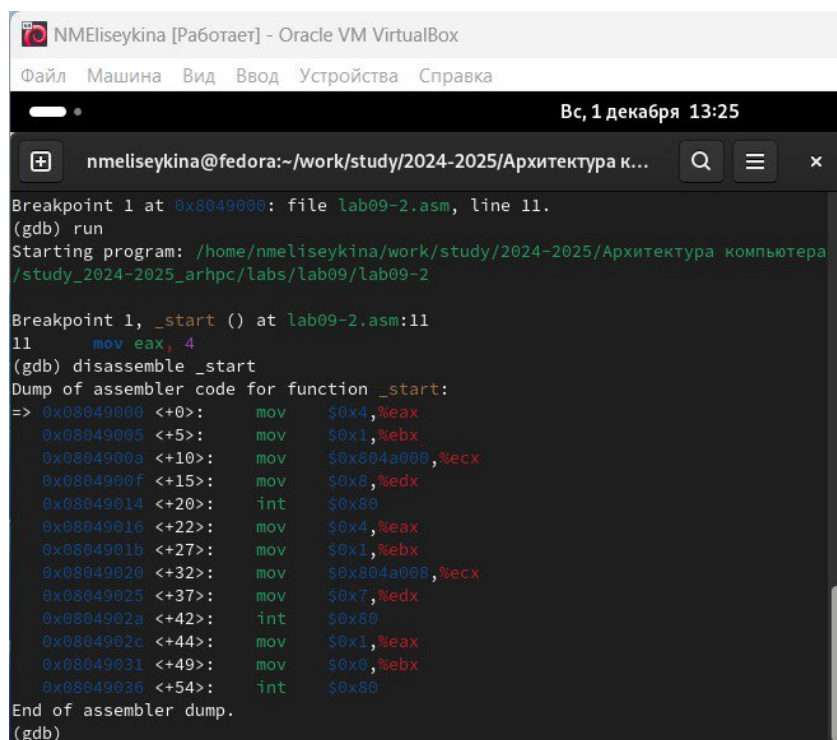
```
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) run
Starting program: /home/nmeliseykina/work/study/2024-2025/Архитектура компьютера
/study_2024-2025_arhpc/labs/lab09/lab09-2

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 4773) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 11.
(gdb) run
Starting program: /home/nmeliseykina/work/study/2024-2025/Архитектура компьютера
/study_2024-2025_arhpc/labs/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:11
11      mov eax, 4
(gdb)
```

Рис. 4.9: Брейкпоинт на метку \_start

Посмотрели дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 4.10 Дисассимилированный код программы).

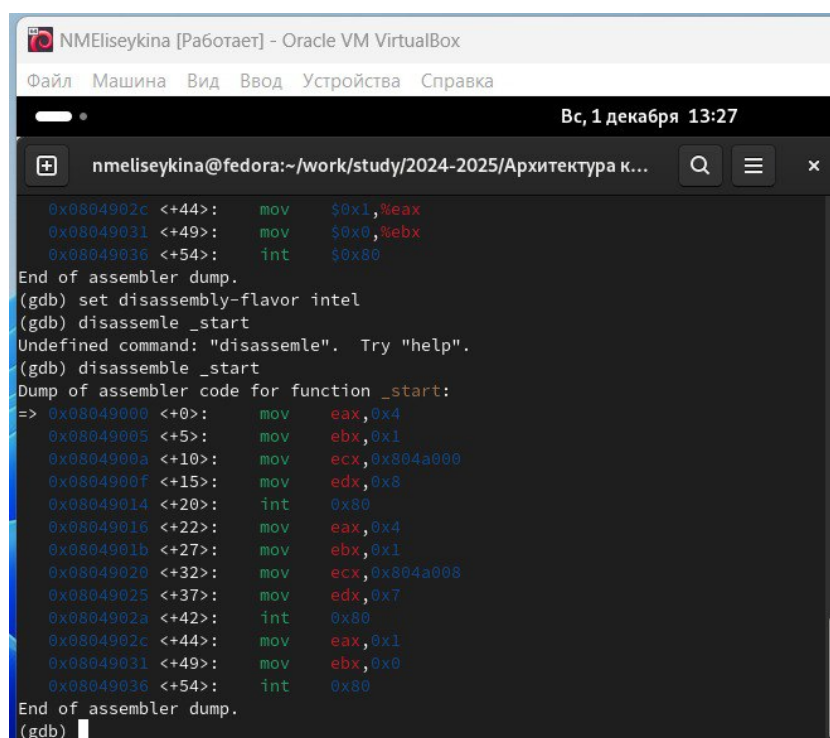


```
Breakpoint 1 at 0x08049000: file lab09-2.asm, line 11.
(gdb) run
Starting program: /home/nmeliseykina/work/study/2024-2025/Архитектура компьютера
/study_2024-2025_arhpc/labs/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:11
11      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     $0x4,%eax
      0x08049005 <+5>:      mov     $0x1,%ebx
      0x0804900a <+10>:     mov     $0x804a000,%ecx
      0x0804900f <+15>:     mov     $0x8,%edx
      0x08049014 <+20>:     int     $0x80
      0x08049016 <+22>:     mov     $0x4,%eax
      0x0804901b <+27>:     mov     $0x1,%ebx
      0x08049020 <+32>:     mov     $0x804a008,%ecx
      0x08049025 <+37>:     mov     $0x7,%edx
      0x0804902a <+42>:     int     $0x80
      0x0804902c <+44>:     mov     $0x1,%eax
      0x08049031 <+49>:     mov     $0x0,%ebx
      0x08049036 <+54>:     int     $0x80
End of assembler dump.
(gdb)
```

Рис. 4.10: Дисассимилированный код программы

Переключились на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. 4.11 Отображение команд с Intel'овским синтаксисом).



```
NMElseykina [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Вс, 1 декабря 13:27
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
0x0804902c <+44>:  mov    $0x1,%eax
0x08049031 <+49>:  mov    $0x0,%ebx
0x08049036 <+54>:  int    $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Undefined command: "disassemble".  Try "help".
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:  mov    eax,0x4
0x08049005 <+5>:  mov    ebx,0x1
0x0804900a <+10>:  mov    ecx,0x804a000
0x0804900f <+15>:  mov    edx,0x8
0x08049014 <+20>:  int    0x80
0x08049016 <+22>:  mov    eax,0x4
0x0804901b <+27>:  mov    ebx,0x1
0x08049020 <+32>:  mov    ecx,0x804a008
0x08049025 <+37>:  mov    edx,0x7
0x0804902a <+42>:  int    0x80
0x0804902c <+44>:  mov    eax,0x1
0x08049031 <+49>:  mov    ebx,0x0
0x08049036 <+54>:  int    0x80
End of assembler dump.
(gdb)
```

Рис. 4.11: Отображение команд с Intel'овским синтаксисом

Отличие заключается в командах, в диссасимилированном отображении в командах используют % и \$, а в Intel отображение эти символы не используются. На такое отображение удобнее смотреть.

Включили режим псевдографики для более удобного анализа программы (рис. 4.12 Режим псевдографики). В этом режиме есть три окна: в верхней части видны названия регистров и их текущие значения; в средней части виден результат диссасимилирования программы; нижняя часть доступна для ввода команд



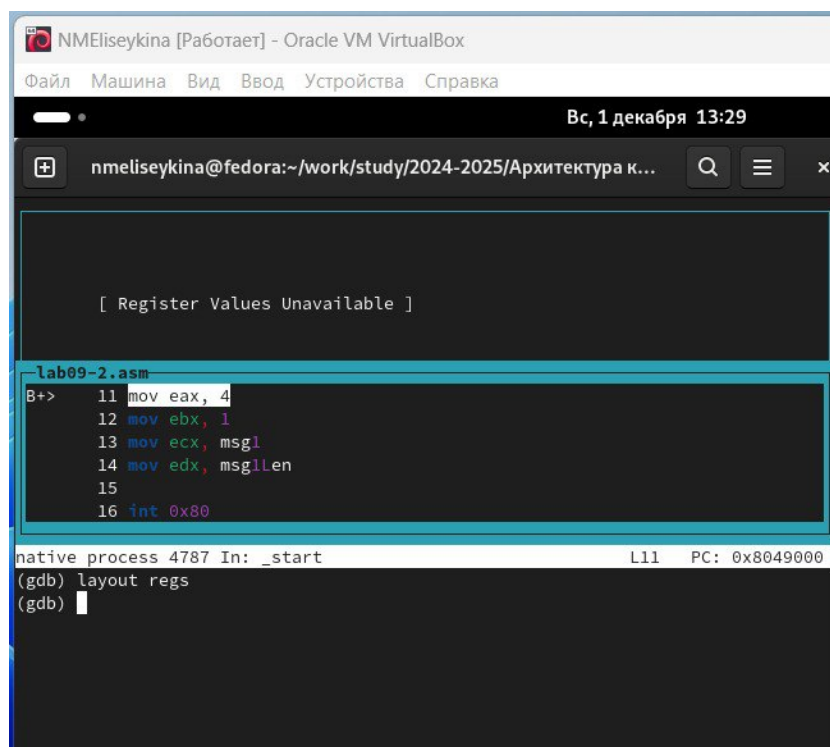


Рис. 4.12: Режим псевдографики

4. Установить точку останова можно командой `break`. Типичный аргумент этой команды — место установки. Его можно задать или как номер строки программы, или как имя метки, или как адрес. Чтобы не было путаницы с номерами, перед адресом ставится «звёздочка»: На предыдущих шагах была установлена точка останова по имени метки (`_start`).

Проверили это с помощью команды `info breakpoints`. Установили еще одну точку останова по адресу инструкции. Адрес инструкции увидели в средней части экрана в левом столбце соответствующей инструкции. Определили адрес предпоследней инструкции (`mov ebx,0x0`) и установили точку останова (рис. 4.13 Установка точки останова по адресу инструкции).

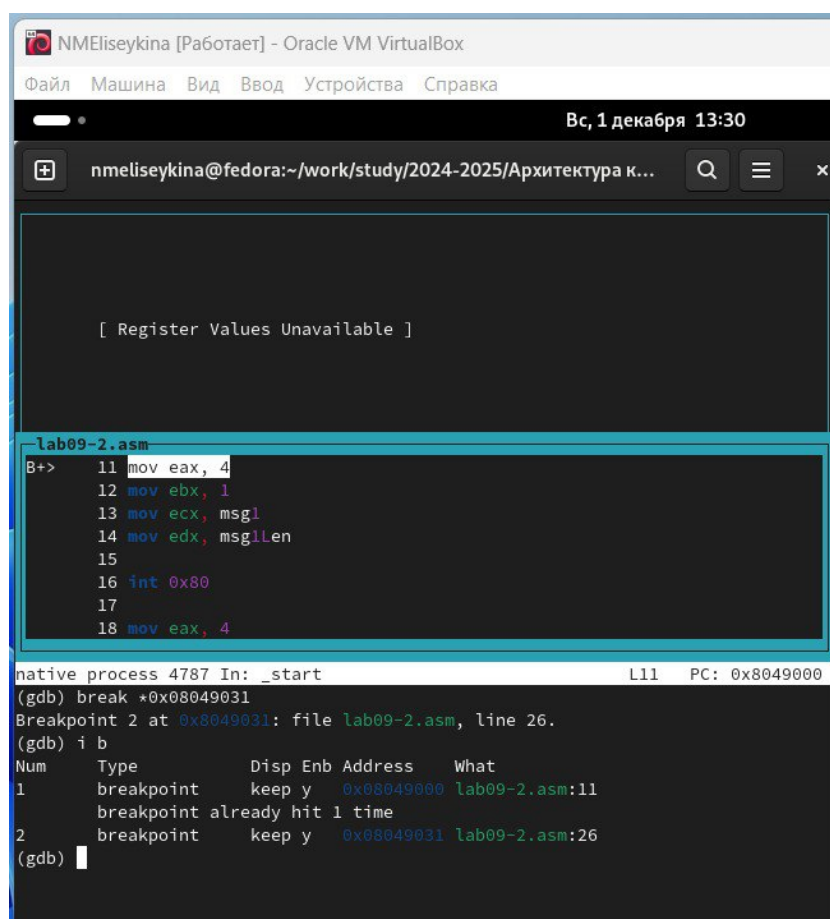


Рис. 4.13: Установка точки останова по адресу инструкции

- Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Выполнили 5 инструкций с помощью команды `si` и проследили за изменением значений регистров. Изменяются регистры `eax` и `ecx`. Посмотреть содержимое регистров также можно с помощью команды `info registers` (рис. 4.14 Просмотр регистров).

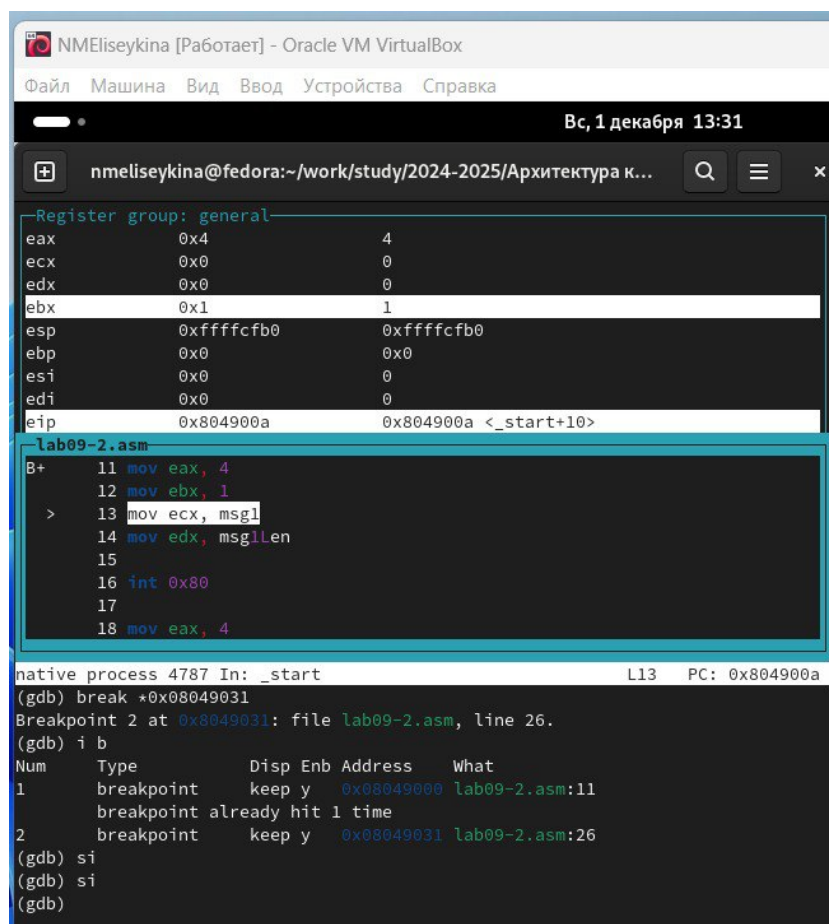
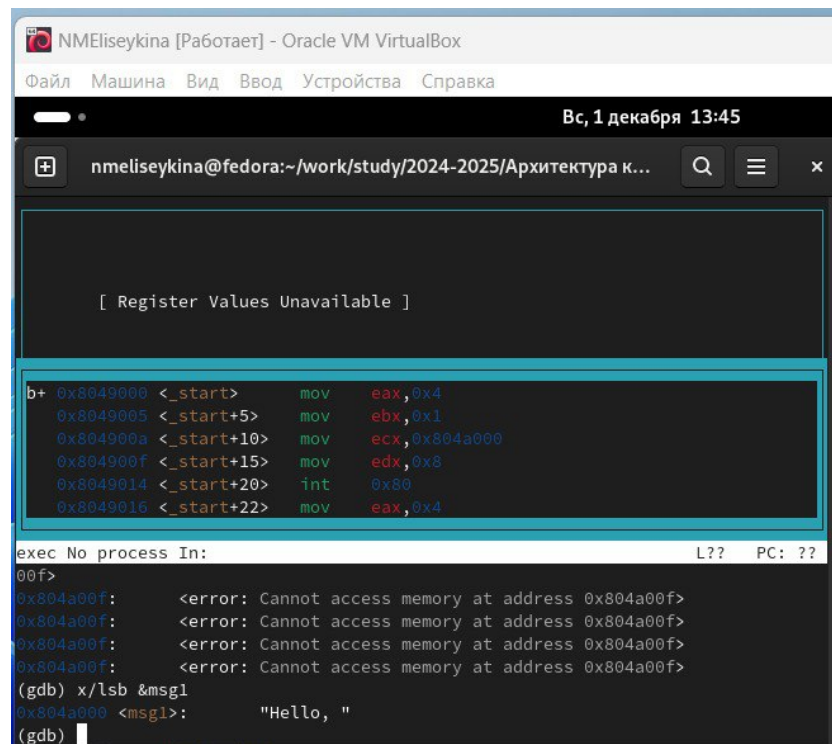


Рис. 4.14: Просмотр регистров

eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffcea0	0xffffcea0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>
eflags	0x202	[ IF ]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43
es	0x2b	43
fs	0x0	0

Рис. 4.15: Измененные регистры

6. С помощью команды посмотрели значение переменной msg1 (рис. 4.16 Значение переменной msg1).



```
NMElseykina [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Вс, 1 декабря 13:45
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
[ Register Values Unavailable ]
b+ 0x8049000 <_start>    mov     eax,0x4
0x8049005 <_start+5>    mov     ebx,0x1
0x804900a <_start+10>   mov     ecx,0x804a000
0x804900f <_start+15>   mov     edx,0x8
0x8049014 <_start+20>   int     0x80
0x8049016 <_start+22>   mov     eax,0x4
exec No process in: L?? PC: ??
00f>
0x804a00f: <error: Cannot access memory at address 0x804a00f>
0x804a00f: <error: Cannot access memory at address 0x804a00f>
0x804a00f: <error: Cannot access memory at address 0x804a00f>
0x804a00f: <error: Cannot access memory at address 0x804a00f>
(gdb) x/lb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) 
```

Рис. 4.16: Значение переменной msg1

7. Посмотрели значение переменной msg2 по адресу. (рис. 4.17 Значение переменной msg2).

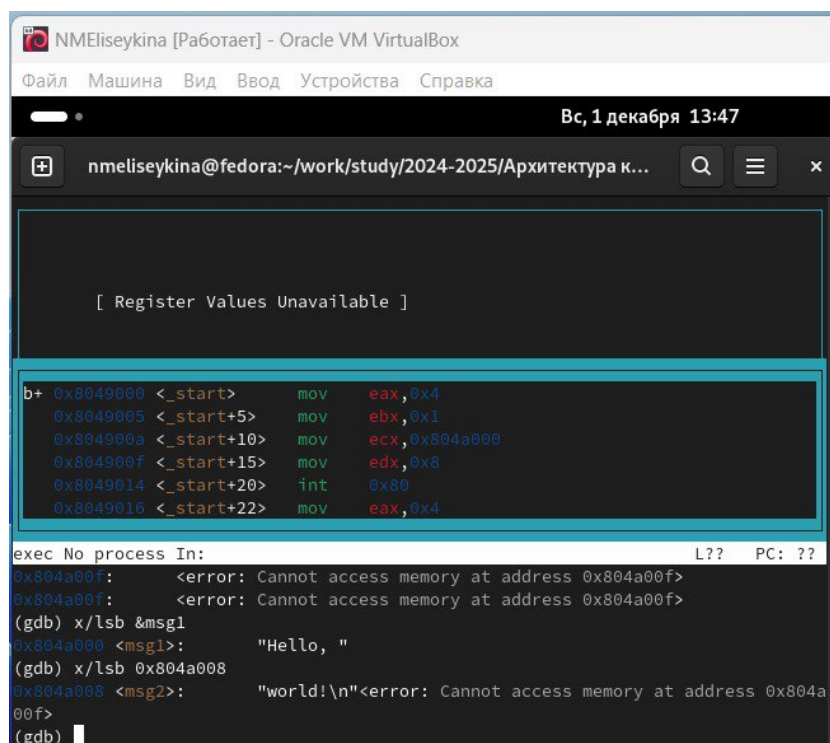


Рис. 4.17: Значение переменной msg2

8. Изменить значение для регистра или ячейки памяти можно с помощью команды `set`, задав ей в качестве аргумента имя регистра или адрес. При этом перед именем регистра ставится префикс `$`, а перед адресом нужно указать в фигурных скобках тип данных (размер сохраняемого значения; в качестве типа данных можно использовать типы языка Си). Изменили первый символ переменной `msg1` (рис. 4.18 Измененное значение переменной `msg1`) и заменили символ во второй переменной `msg2` (рис. 4.19 Измененное значение переменной `msg2`).



Рис. 4.18: Измененное значение переменной msg1

```
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hhlllo, "
```

Рис. 4.19: Измененное значение переменной msg2

9. Вывели в различных форматах значение регистра edx (рис. 4.20 Значение регистра edx в различных форматах).

```
(gdb) set {char}0x804a008='L'
(gdb) set {char}0x804a00b=' '
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "Lor d!\n\034"
```

Рис. 4.20: Значение регистра edx в различных форматах

10. С помощью команды set изменили значение регистра ebx (рис. 4.21 Значение регистра ebx).

```
(gdb) p/s $edx
$1 = 8
(gdb) p/t $edx
$2 = 1000
(gdb) p/x $edx
$3 = 0x8
(gdb)
```

Рис. 4.21: Значение регистра ebx

Команда выводит разные значения, так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум, поэтому и значения разные.

11. Завершили выполнение программы и вышли из GDB. (рис. 4.22 Завершение работы с файлов)

```
(gdb) set $ebx='2'  
(gdb) p/s $ebx  
$7 = 50  
(gdb) set $ebx=2  
(gdb) p/s $ebx  
$8 = 2
```

Рис. 4.22: Завершение работы с файлов

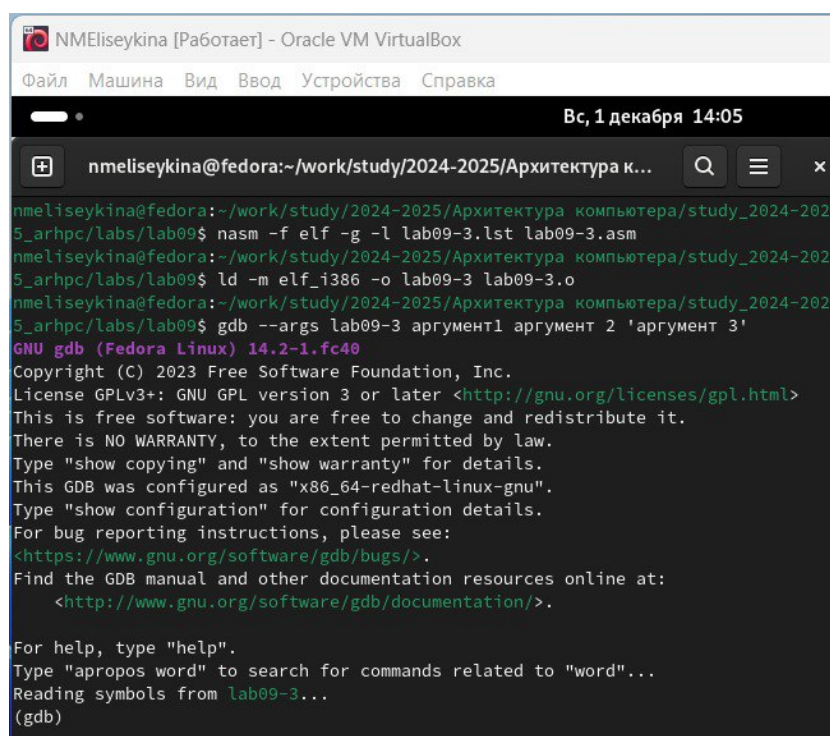
12. Скопировали файл lab8-2.asm. Создали исполняемый файл (рис. 4.23 Создание исполняемого файла lab09-3.asm).

```
[Inferior 1 (process 3985) exited normally]
```

Рис. 4.23: Создание исполняемого файла lab09-3.asm

13. Для загрузки в gdb программы с аргументами необходимо использовать ключ `-args` (рис. 4.24 Запуск файла в отладчике).





```
NMEliseykina [Работаer] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Вс, 1 декабря 14:05
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Fedora Linux) 14.2-1.fc40
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
(gdb)
```

Рис. 4.24: Запуск файла в отладчике

- Исследовали расположение аргументов командной строки в стеке после запуска программы с помощью gdb. Для начала установили точку останова перед первой инструкцией в программе и запустим ее (рис. 4.25 Точка останова перед первой инструкцией в программе).

```
NMElseykina [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Вс, 1 декабря 14:08
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 6.
(gdb) r
Starting program: /home/nmeliseykina/work/study/2024-2025/Архитектура компьютера
/study_2024-2025_arhpc/labs/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.

Breakpoint 1, _start () at lab09-3.asm:6
6      pop ecx
(gdb)
```

Рис. 4.25: Точка останова перед первой инструкцией в программе

Проверили адрес вершины стека и убедились, что там хранится 5 элементов (рис. 4.26 Адрес вершины стека).

```
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

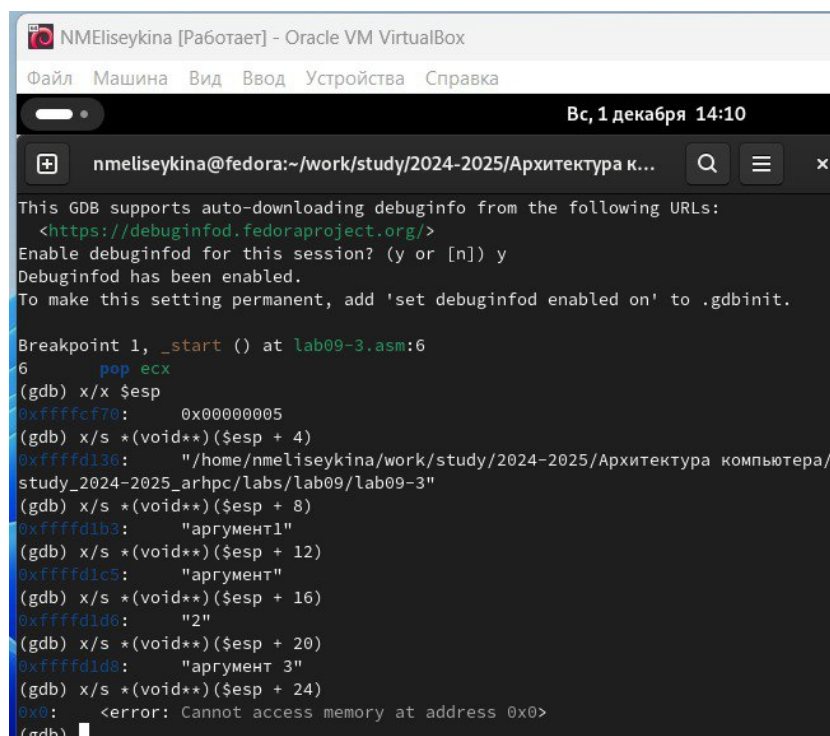
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 6.
(gdb) r
Starting program: /home/nmeliseykina/work/study/2024-2025/Архитектура компьютера
/study_2024-2025_arhpc/labs/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.

Breakpoint 1, _start () at lab09-3.asm:6
6      pop ecx
(gdb) x/x $esp
0xffffcf70: 0x00000005
(gdb)
```

Рис. 4.26: Адрес вершины стека

По первому адресу хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации (рис.4.27 Все позиции стека).



```
NMElseykina [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Вс, 1 декабря 14:10
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.

Breakpoint 1, _start () at lab09-3.asm:6
6      pop ecx
(gdb) x/x $esp
0xffffcf70:      0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffd130:      "/home/nmeliseykina/work/study/2024-2025/Архитектура компьютера/
study_2024-2025_arhpc/labs/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd1b3:      "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd1c5:      "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd1d0:      "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd1d8:      "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:      <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 4.27: Все позиции стека

#### Задание для самостоятельной работы

1. Преобразовать программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции  $f(x)$  как подпрограмму.

```
%include 'in_out.asm'

SECTION .data
prim DB 'f(x)=7+2x',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

pop ecx
pop edx
sub ecx,1
mov esi,0

mov eax,prim
call sprintf
next:
cmp ecx,0
jz _end

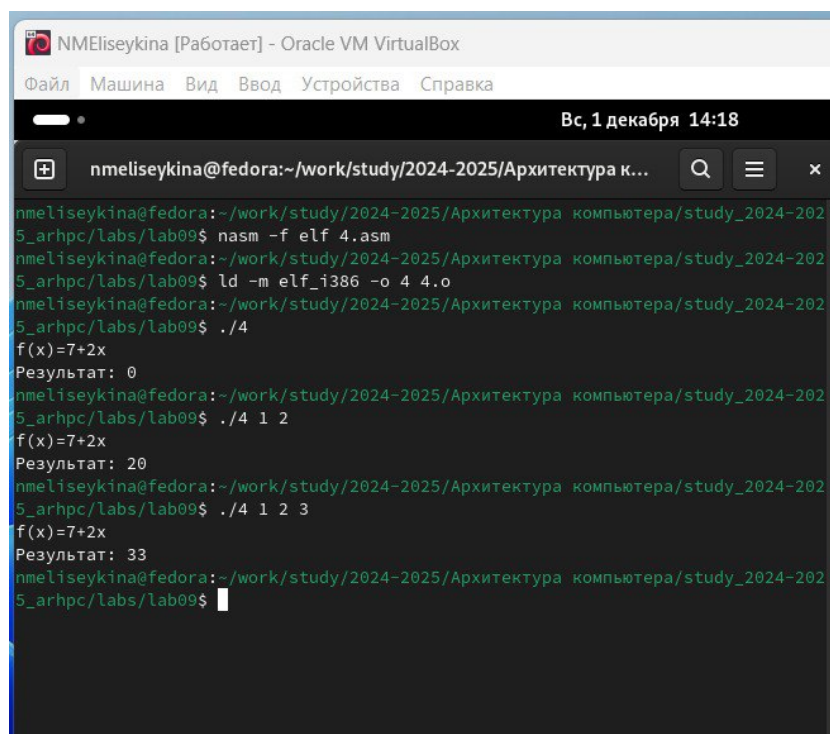
mov ebx,2
pop eax
call atoi
mul ebx

add eax,7
add esi,eax
loop next

_end:
mov eax,otv
call sprintf
mov eax,esi
call fprintf
call quit

fir:
mov ebx,2
mul ebx
add eax,7
ret
```

Рис. 4.28: Листинг 1 самостоятельного задания №1



```
nmliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ nasm -f elf 4.asm
nmliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ld -m elf_i386 -o 4 4.o
nmliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ./4
f(x)=7+2x
Результат: 0
nmliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ./4 1 2
f(x)=7+2x
Результат: 20
nmliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ./4 1 2 3
f(x)=7+2x
Результат: 33
nmliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$
```

Рис. 4.29: Результат работы программы

2. В листинге из самостоятельного задания приведена программа вычисления выражения  $(3 + 2) * 4 + 5$ . При запуске данная программа дает неверный результат. Проверить это. С помощью отладчика GDB, анализируя изменения значений регистров, определить ошибку и исправить ее.
3. 1.Переписали программу и попробовали запустить ее чтобы увидеть ошибку.

```
%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0

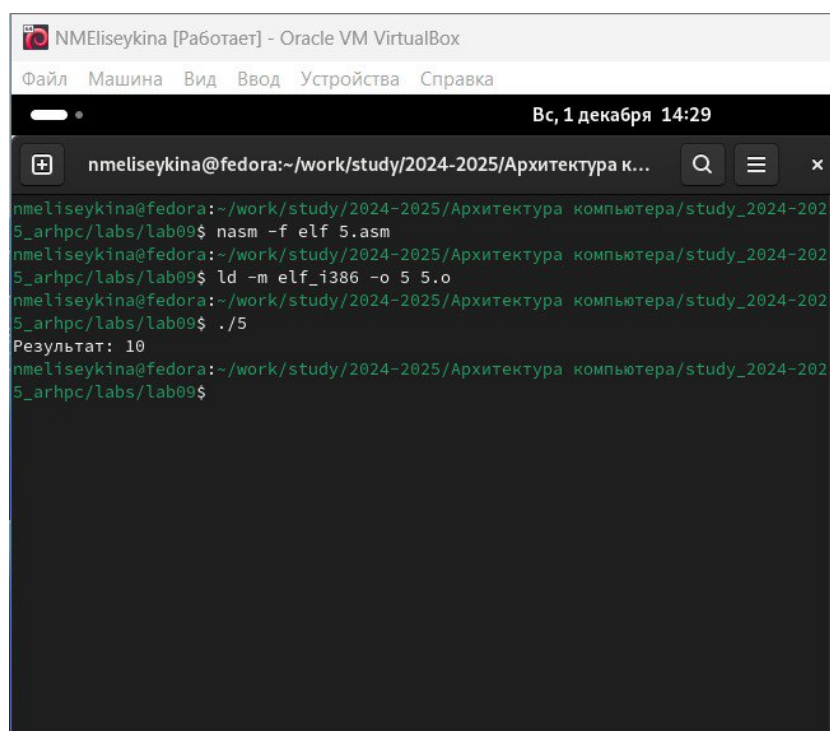
SECTION .text
GLOBAL _start
_start:

mov ebx,3
mov eax,2
add eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax

mov eax,div
call sprint
mov eax,edi
call iprintLF

call quit
```

Рис. 4.30: Листинг из самостоятельного задания №2

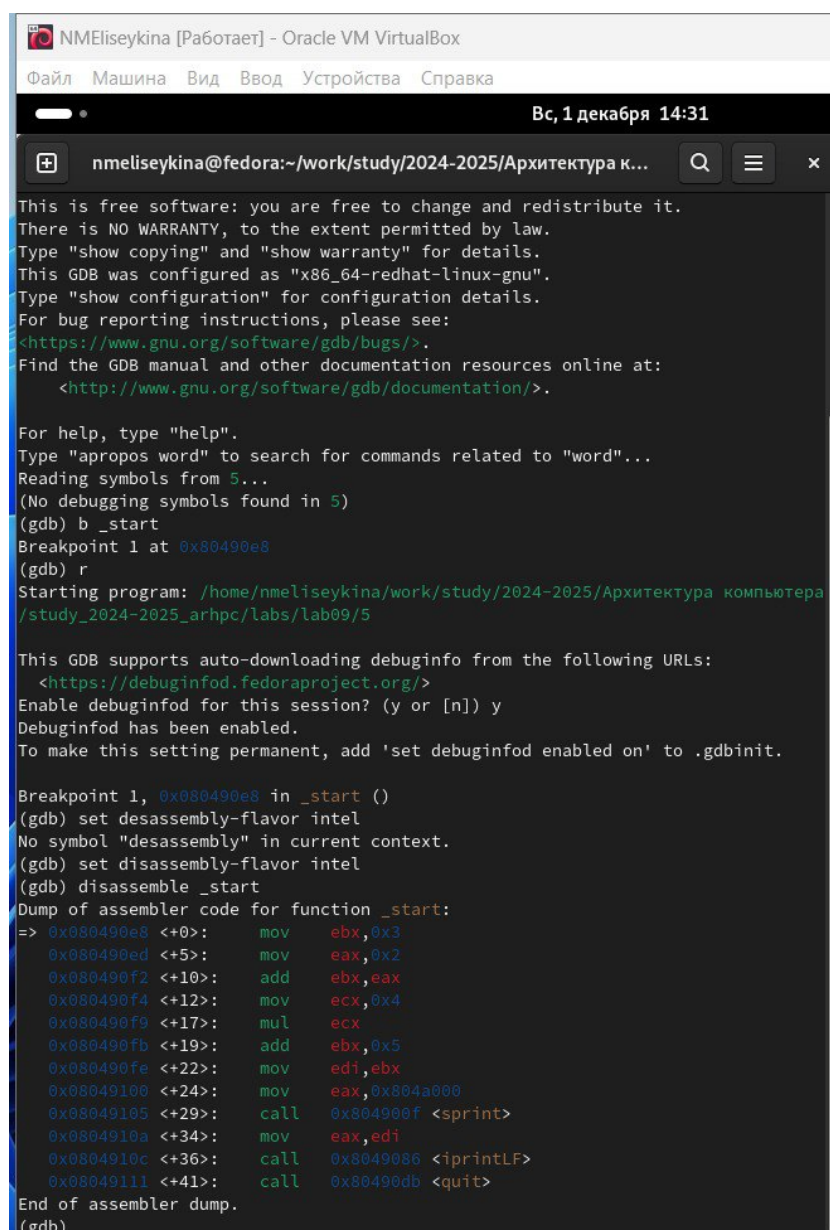


```
nmeiseykina [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Вс, 1 декабря 14:29
nmeiseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ nasm -f elf 5.asm
nmeiseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ld -m elf_i386 -o 5 5.o
nmeiseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ./5
Результат: 10
nmeiseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$
```

Рис. 4.31: Проверка результата работы программы из Листинга самостоятельного задания №2

2. После обнаружения ошибки, запустили программу в отладчике (рис. 4.32 Запуск программы в отладчике).





```
NMElseykina [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка
Вс, 1 декабря 14:31
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from 5...
(No debugging symbols found in 5)
(gdb) b _start
Breakpoint 1 at 0x80490e8
(gdb) r
Starting program: /home/nmeliseykina/work/study/2024-2025/Архитектура компьютера
/study_2024-2025_arhpc/labs/lab09/5

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.

Breakpoint 1, 0x80490e8 in _start ()
(gdb) set disassembly-flavor intel
No symbol "disassembly" in current context.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x80490e8 <+0>:    mov     ebx,0x3
0x80490ed <+5>:    mov     eax,0x2
0x80490f2 <+10>:   add     ebx,eax
0x80490f4 <+12>:   mov     ecx,0x4
0x80490f9 <+17>:   mul     ecx
0x80490fb <+19>:   add     ebx,0x5
0x80490fe <+22>:   mov     edi,ebx
0x8049100 <+24>:   mov     eax,0x804a000
0x8049105 <+29>:   call   0x804900f <sprint>
0x804910a <+34>:   mov     eax,edi
0x804910c <+36>:   call   0x8049086 <iprintLF>
0x8049111 <+41>:   call   0x80490db <quit>
End of assembler dump.
(gdb)
```

Рис. 4.32: Запуск программы в отладчике

2. 3.Открыли регистры и проанализировали их, поняли что некоторые регистры стоят не на своих местах, исправили это.

The screenshot shows a debugger window titled "NMEliseykina [Работает] - Oracle VM VirtualBox". The menu bar includes "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The status bar at the top right shows "Вс, 1 декабря 14:32". The address bar displays the path "nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...". The main window shows assembly code with the following instructions:

```
B+> 0x80490e8 <_start>    mov     ebx,0x3
0x80490ed <_start+5>    mov     eax,0x2
0x80490f2 <_start+10>   add     ebx,eax
0x80490f4 <_start+12>   mov     ecx,0x4
0x80490f9 <_start+17>   mul     ecx
0x80490fb <_start+19>   add     ebx,0x5
0x80490fe <_start+22>   mov     edi,ebx
0x8049100 <_start+24>   mov     eax,0x804a000
0x8049105 <_start+29>   call    0x804900f <sprint>
0x804910a <_start+34>   mov     eax,edi
0x804910c <_start+36>   call    0x8049086 <iprintf>
0x8049111 <_start+41>   call    0x80490db <quit>
0x8049116          add     BYTE PTR [eax],al
0x8049118          add     BYTE PTR [eax],al
0x804911a          add     BYTE PTR [eax],al
0x804911c          add     BYTE PTR [eax],al
0x804911e          add     BYTE PTR [eax],al
0x8049120          add     BYTE PTR [eax],al
0x8049122          add     BYTE PTR [eax],al
0x8049124          add     BYTE PTR [eax],al
0x8049126          add     BYTE PTR [eax],al
0x8049128          add     BYTE PTR [eax],al
0x804912a          add     BYTE PTR [eax],al
0x804912c          add     BYTE PTR [eax],al
0x804912e          add     BYTE PTR [eax],al
0x8049130          add     BYTE PTR [eax],al
0x8049132          add     BYTE PTR [eax],al
0x8049134          add     BYTE PTR [eax],al
```

Рис. 4.33: Анализ регистров

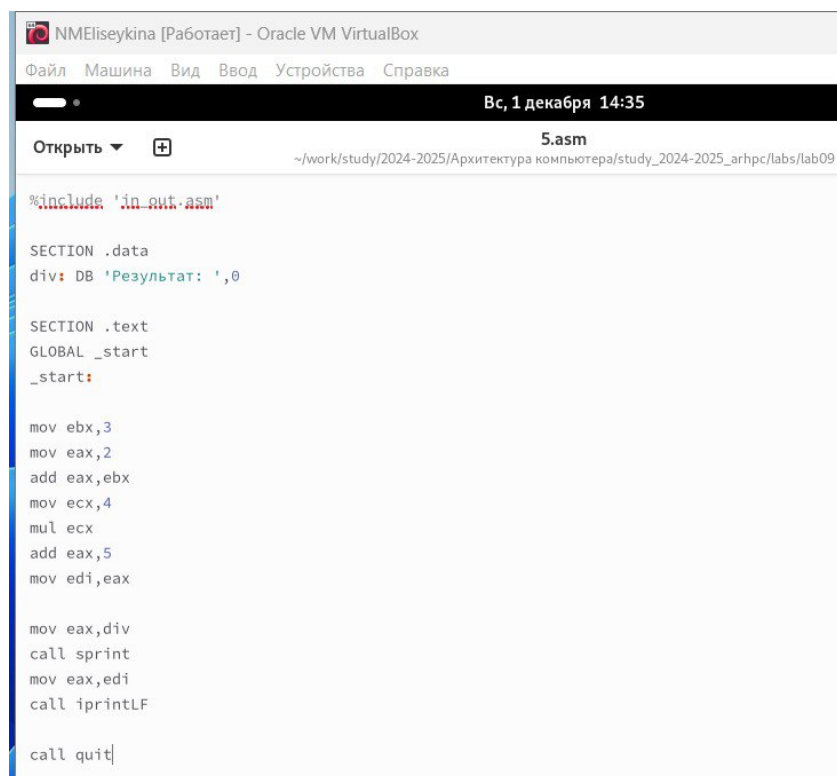
2. 4.Изменили регистры и запустили программу, программа вывела ответ 25, то есть все работает правильно (рис. 4.34 Повторный запуск программы).

```
NMElseykina [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка
Вс, 1 декабря 14:35
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ gdb 5
GNU gdb (Fedora Linux) 14.2-1.fc40
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from 5...
(No debugging symbols found in 5)
(gdb) r
Starting program: /home/nmeliseykina/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09/5

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Результат: 25
[Inferior 1 (process 6806) exited normally]
(gdb) q
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$
```

Рис. 4.34: Повторный запуск программы



The screenshot shows a virtual machine window titled "NMElseykina [Работает] - Oracle VM VirtualBox". The menu bar includes "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The system clock shows "Вс, 1 декабря 14:35". The text editor displays the file "5.asm" at the path "~/work/study/2024-2025/Архитектура компьютера/study\_2024-2025\_arhpc/labs/lab09". The assembly code is as follows:

```
%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0

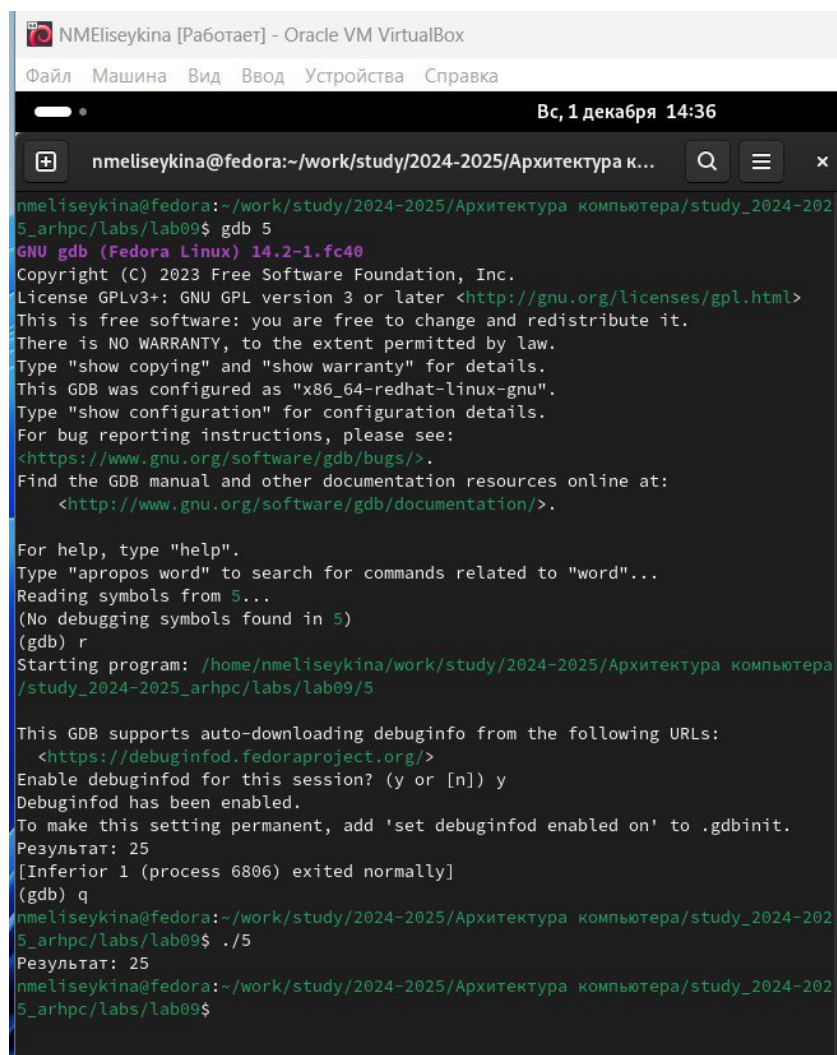
SECTION .text
GLOBAL _start
_start:

mov ebx,3
mov eax,2
add eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax

mov eax,div
call sprint
mov eax,edi
call iprintLF

call quit|
```

Рис. 4.35: Листинг программы



```
nmeliseykina [Работает] - Oracle VM VirtualBox
Файл Машина Вид Ввод Устройства Справка
Вс, 1 декабря 14:36
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура к...
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ gdb 5
GNU gdb (Fedora Linux) 14.2-1.fc40
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from 5...
(No debugging symbols found in 5)
(gdb) r
Starting program: /home/nmeliseykina/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09/5
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Результат: 25
[Inferior 1 (process 6806) exited normally]
(gdb) q
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$ ./5
Результат: 25
nmeliseykina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab09$
```

Рис. 4.36: Проверка результата работы программы

## 5 Выводы

Приобрели навыки написания программ с использованием подпрограмм. Познакомились с методами отладки при помощи GDB и его основными возможностями.

## Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).