

# **Лабораторная работа №9**

**Архитектура компьютера**

Мурашов Иван Вячеславович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>6</b>
<b>2</b>	<b>Задание</b>	<b>7</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
3.1	Реализация подпрограмм в NASM . . . . .	8
3.2	Отладка программ с помощью GDB . . . . .	14
3.3	Добавление точек останова . . . . .	20
3.4	Работа с данными программы в GDB . . . . .	21
3.5	Обработка аргументов командной строки в GDB . . . . .	26
3.6	Выполнение заданий для самостоятельной работы . . . . .	29
<b>4</b>	<b>Выводы</b>	<b>38</b>

## Список иллюстраций

3.1	Создание каталога и файла в нём . . . . .	8
3.2	Редактирование файла . . . . .	9
3.3	Трансляция, компоновка и запуск файлов . . . . .	11
3.4	Редактирование файла . . . . .	12
3.5	Трансляция, компоновка и запуск файлов . . . . .	14
3.6	Создание файла . . . . .	15
3.7	Редактирование файла . . . . .	15
3.8	Трансляция, компоновка и запуск файлов . . . . .	16
3.9	Загрузка исполняемого файла в отладчик gdb . . . . .	17
3.10	Запуск программы в оболочке GDB . . . . .	17
3.11	Установка брейкпоинта в оболочке GDB . . . . .	17
3.12	Запуск программы в оболочке GDB . . . . .	18
3.13	Просмотр дизассимилированного кода программы в оболочке GDB	18
3.14	Переключение синтаксиса при отображении команд в оболочке GDB	18
3.15	Просмотр дизассимилированного кода программы в оболочке GDB	19
3.16	Режим псевдографики GDB . . . . .	20
3.17	Получение информации о точках останова в оболочке GDB . . . . .	20
3.18	Установка точки останова по адресу инструкции в оболочке GDB .	21
3.19	Получение информации о точках останова в оболочке GDB . . . . .	21
3.20	Получение информации о точках останова в оболочке GDB . . . . .	22
3.21	Выполнение 5 шагов программы в оболочке GDB . . . . .	22
3.22	Получение информации о точках останова в оболочке GDB . . . . .	23
3.23	Работа с переменными в оболочке GDB . . . . .	23
3.24	Работа с переменными в оболочке GDB . . . . .	24
3.25	Работа с переменными в оболочке GDB . . . . .	24
3.26	Работа с переменными в оболочке GDB . . . . .	24
3.27	Работа с регистрами в оболочке GDB . . . . .	25
3.28	Работа с регистрами в оболочке GDB . . . . .	26
3.29	Работа в оболочке GDB . . . . .	26
3.30	Копирование файла . . . . .	27
3.31	Трансляция и компоновка файлов . . . . .	27
3.32	Загрузка исполняемого файла в отладчик gdb . . . . .	27
3.33	Установка точки останова и запуск программы в оболочке GDB . .	28
3.34	Просмотр позиций стека в оболочке GDB . . . . .	28
3.35	Копирование файла . . . . .	29
3.36	Редактирование файла . . . . .	30
3.37	Трансляция, компоновка и запуск файлов . . . . .	32

3.38	Создание файла . . . . .	32
3.39	Редактирование файла . . . . .	33
3.40	Трансляция, компоновка и запуск файлов . . . . .	34
3.41	Загрузка исполняемого файла в отладчик gdb . . . . .	34
3.42	Запуск программы в оболочке GDB . . . . .	35
3.43	Установка брейкпоинтов в оболочке GDB . . . . .	35
3.44	Работа с переменными в оболочке GDB . . . . .	36
3.45	Редактирование файла . . . . .	37
3.46	Трансляция, компоновка и запуск файлов . . . . .	37

## **Список таблиц**

# 1 Цель работы

Целью данной лабораторной работы является приобретение навыков написания программ с использованием подпрограмм, знакомство с методами отладки при помощи GDB и его основными возможностями.

## 2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Добавление точек останова
4. Работа с данными программы в GDB
5. Обработка аргументов командной строки в GDB
6. Выполнение заданий для самостоятельной работы

## 3 Выполнение лабораторной работы

### 3.1 Реализация подпрограмм в NASM

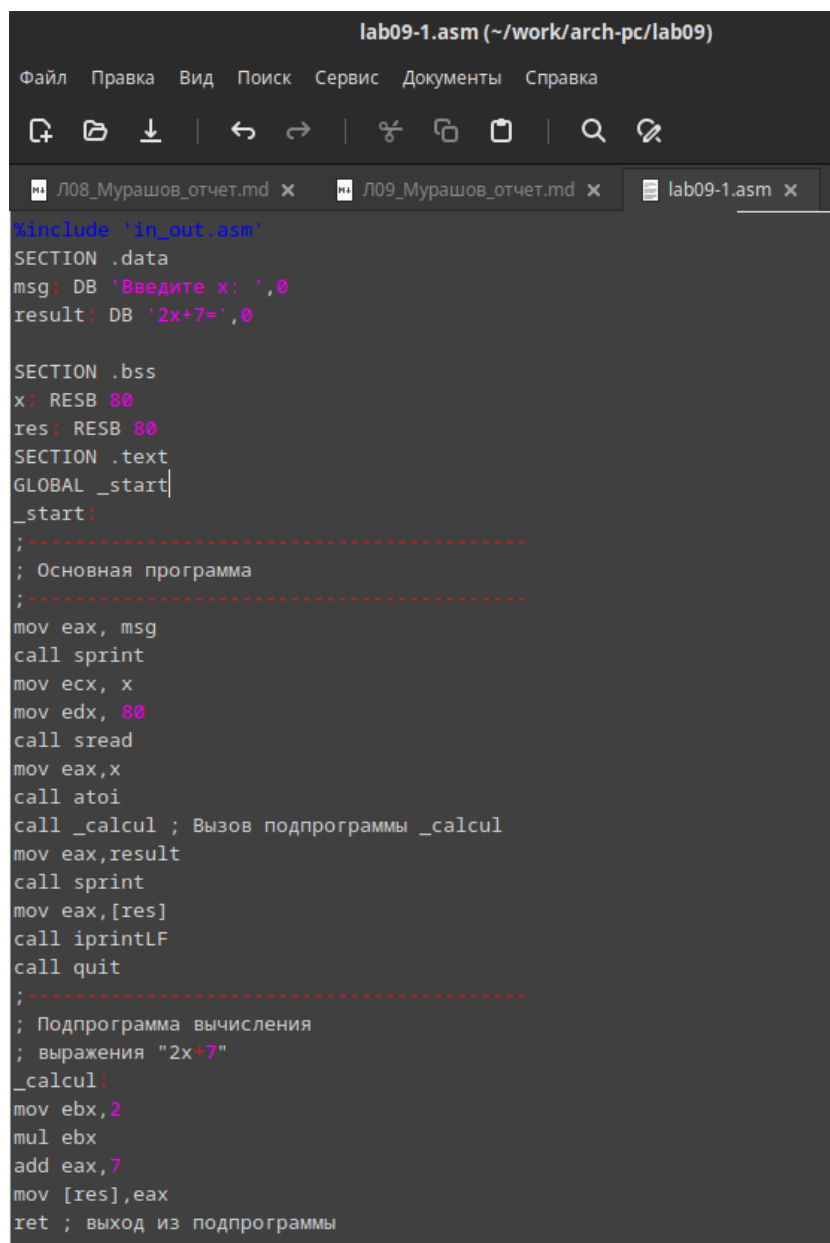
Создаю каталог для программ лабораторной работы №9, перехожу в него и создаю файл lab09-1.asm (рис. [3.1]).

```
[ivmurashov@fedora ~]$ mkdir ~/work/arch-pc/lab09  
[ivmurashov@fedora ~]$ cd ~/work/arch-pc/lab09  
[ivmurashov@fedora lab09]$ touch lab09-1.asm
```

Рис. 3.1: Создание каталога и файла в нём

Ввожу в файл lab09-1.asm текст программы из листинга 9.1 (рис. [3.2]).





```
lab09-1.asm (~/work/arch-pc/lab09)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Icons]
lab09-1.asm x
%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:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы
```

Рис. 3.2: Редактирование файла

### Листинг 1. Пример программы с использованием вызова подпрограммы

```
%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:

;-----
; Основная программа
;-----

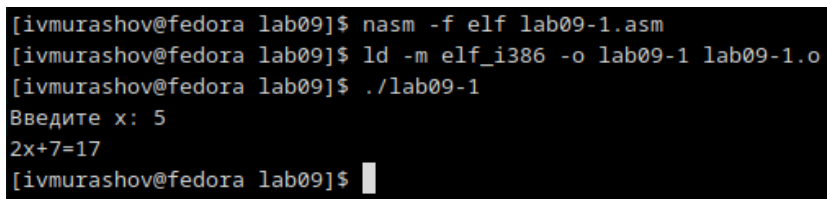
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit

;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx

```

```
add eax,7
mov [res],eax
ret ; выход из подпрограммы
```

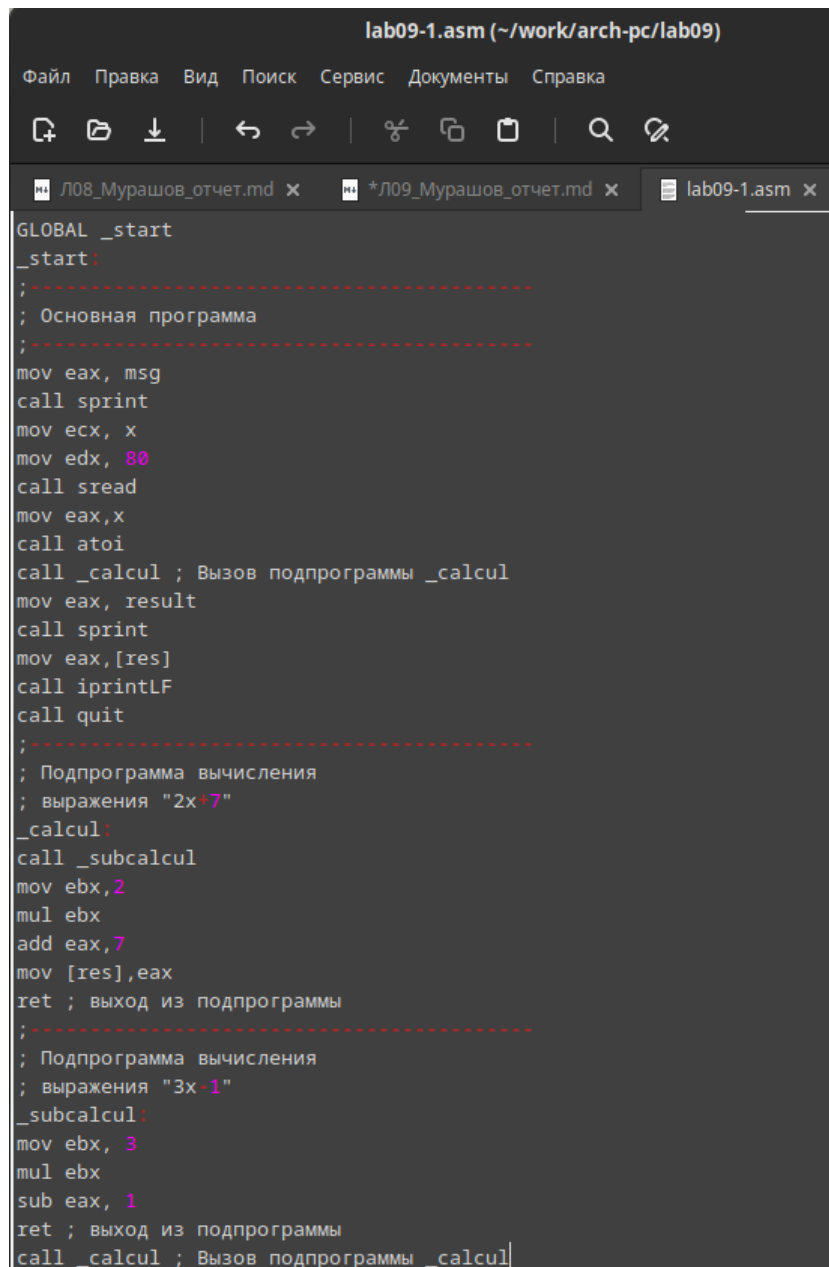
Создаю исполняемый файл и запускаю его. Проверяю работу программы, вручную посчитав искомое значение. Результаты совпали (рис. [3.3]).



```
[ivmurashov@fedora lab09]$ nasm -f elf lab09-1.asm
[ivmurashov@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[ivmurashov@fedora lab09]$ ./lab09-1
Введите x: 5
2x+7=17
[ivmurashov@fedora lab09]$
```

Рис. 3.3: Трансляция, компоновка и запуск файлов

Изменяю 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. [3.4]).



```
lab09-1.asm (~/work/arch-pc/lab09)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Иконки]
lab08_Мурашов_отчет.md x  *lab09_Мурашов_отчет.md x  lab09-1.asm x
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы
;-----
; Подпрограмма вычисления
; выражения "3x-1"
_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret ; выход из подпрограммы
call _calcul ; Вызов подпрограммы _calcul
```

Рис. 3.4: Редактирование файла

**Листинг 1.1. Программа с использованием вызова подпрограммы и её подпрограммы**

```
%include 'in_out.asm'
SECTION .data
```

```
msg: DB 'Введите x: ', 0
result: DB 'f(g(x))=', 0
```

```
SECTION .bss
```

```
x: RESB 80
```

```
res: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
;-----
```

```
; Основная программа
```

```
;-----
```

```
mov eax, msg
```

```
call sprint
```

```
mov ecx, x
```

```
mov edx, 80
```

```
call sread
```

```
mov eax, x
```

```
call atoi
```

```
call _calcul ; Вызов подпрограммы _calcul
```

```
mov eax, result
```

```
call sprint
```

```
mov eax, [res]
```

```
call iprintLF
```

```
call quit
```

```
;-----
```

```
; Подпрограмма вычисления
```

```
; выражения "2x+7"
```

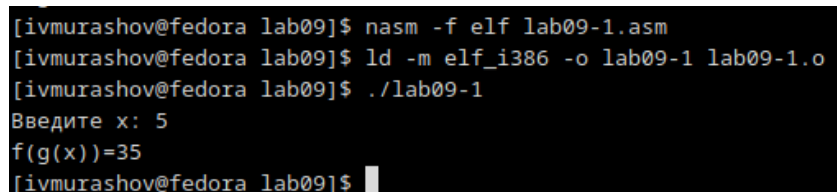
```
_calcul:
```

```

call _subcalcul
mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret ; выход из подпрограммы
;-----
; Подпрограмма вычисления
; выражения "3x-1"
_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret ; выход из подпрограммы
call _calcul ; Вызов подпрограммы _calcul

```

Создаю исполняемый файл и запускаю его. Проверяю работу программы, вручную посчитав искомое значение. Результаты совпали(рис. [3.5]).



```

[ivmurashov@fedora lab09]$ nasm -f elf lab09-1.asm
[ivmurashov@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[ivmurashov@fedora lab09]$ ./lab09-1
Введите x: 5
f(g(x))=35
[ivmurashov@fedora lab09]$

```

Рис. 3.5: Трансляция, компоновка и запуск файлов

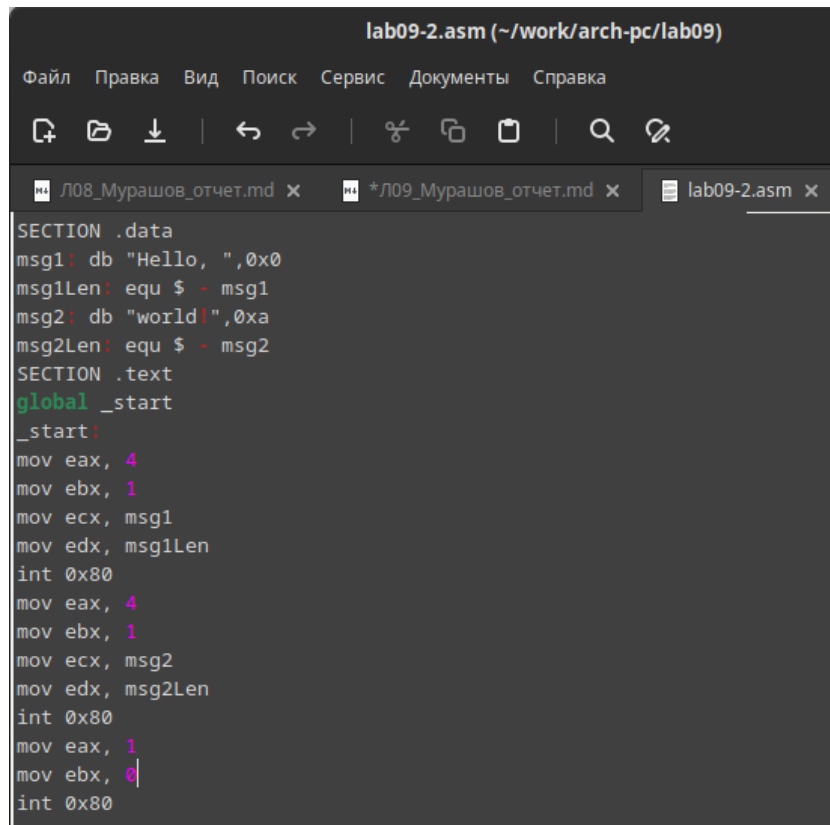
## 3.2 Отладка программ с помощью GDB

Создаю файл lab09-2.asm в каталоге ~/work/arch-pc/lab09 (рис. [3.6]).

```
[ivmurashov@fedora lab09]$ touch lab09-2.asm
```

Рис. 3.6: Создание файла

Ввожу в файл lab09-2.asm текст программы из листинга 9.2 (рис. [3.7]).



```
lab09-2.asm (~/work/arch-pc/lab09)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[иконка] Л08_Мурашов_отчет.md x  [иконка] *Л09_Мурашов_отчет.md x  [иконка] lab09-2.asm x
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
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. 3.7: Редактирование файла

## Листинг 2. Программа вывода сообщения Hello world!

```
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
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

```

Получаю исполняемый файл. Для работы с GDB добавляю в исполняемый файл отладочную информацию, для этого провожу трансляцию программ с ключом ‘-g’ (рис. [3.8]).

```

[ivmurashov@fedora lab09]$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
[ivmurashov@fedora lab09]$ ld -m elf_i386 -o lab09-2 lab09-2.o

```

Рис. 3.8: Трансляция, компоновка и запуск файлов

Загружаю исполняемый файл в отладчик gdb (рис. [3.9]).



```
[ivmurashov@fedora lab09]$ gdb lab09-2
GNU gdb (GDB) Fedora Linux 13.2-6.fc38
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. 3.9: Загрузка исполняемого файла в отладчик gdb

Проверяю работу программы, запустив её в оболочке GDB с помощью команды `run` (сокращённо `r`) (рис. [3.10]).

```
(gdb) run
Starting program: /home/ivmurashov/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 3695) exited normally]
(gdb) 
```

Рис. 3.10: Запуск программы в оболочке GDB

Для более подробного анализа программы устанавливаю брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы (рис. [3.11]).

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
```

Рис. 3.11: Установка брейкпоинта в оболочке GDB

Запускаю программу (рис. [3.12]).

```
(gdb) run
Starting program: /home/ivmurashov/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
```

Рис. 3.12: Запуск программы в оболочке GDB

Просматриваю дизассимилированный код программы с помощью команды `disassemble`, начиная с метки `_start` (рис. [3.13]).

```
(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.
```

Рис. 3.13: Просмотр дизассимилированного кода программы в оболочке GDB

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. [3.14]).

```
(gdb) set disassembly-flavor intel
```

Рис. 3.14: Переключение синтаксиса при отображении команд в оболочке GDB

Просматриваю дизассемблированный код программы, начиная с метки `_start` (рис. [3.15]).

```
(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.
```

Рис. 3.15: Просмотр дизассемблированного кода программы в оболочке GDB

Синтаксис машинных команд в режиме АТТ включает в себя использование значка '\$' перед операндами, значка '%' перед регистрами в то время как синтаксис в режиме Intel этих значков нет. Также в режиме АТТ сначала указываются операнды, а потом регистры. В Intel наоборот - сначала регистры, затем операнды.

Включаю режим псевдографики для более удобного анализа программы рис. [3.16]).

```
ivmurashov@fedora:~  
Файл Правка Вид Поиск Терминал Справка  
[ 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  
native process 4865 In: _start L9 PC: 0x8049000  
(gdb) layout regs  
(gdb)
```

Рис. 3.16: Режим псевдографики GDB

### 3.3 Добавление точек останова

На предыдущих шагах была установлена точка останова по имени метки (`_start`). Проверяю это с помощью команды `info breakpoints` (кратко `i b`) (рис. [3.17]).

```
ivmurashov@fedora:~  
Файл Правка Вид Поиск Терминал Справка  
[ 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  
native process 4865 In: _start L9 PC: 0x8049000  
(gdb) layout regs  
(gdb) i b  
Num    Type           Disp Enb Address      What  
1      breakpoint      keep y  0x08049000 lab09-2.asm:9  
      breakpoint already hit 1 time  
(gdb)
```

Рис. 3.17: Получение информации о точках останова в оболочке GDB

Определяю адрес предпоследней инструкции (mov ebx, 0x0) и устанавливаю точку останова рис. [3.18]).

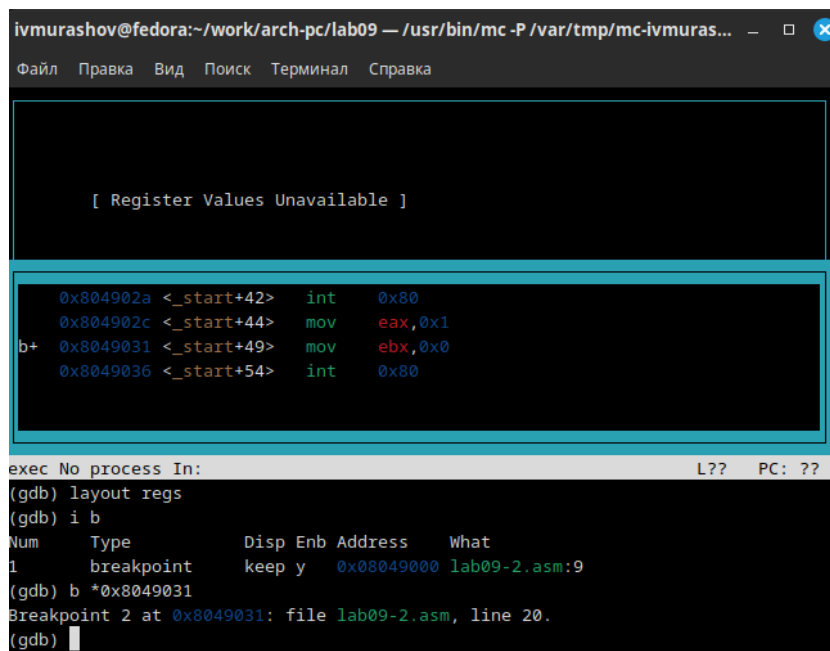


Рис. 3.18: Установка точки останова по адресу инструкции в оболочке GDB

Просматриваю информацию о всех установленных точках останова (рис. [3.19]).

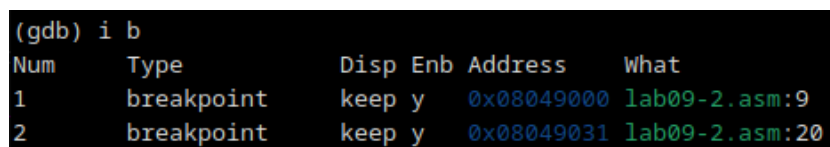


Рис. 3.19: Получение информации о точках останова в оболочке GDB

### 3.4 Работа с данными программы в GDB

Просматриваю содержимое регистров с помощью команды info registers (или i r) (рис. [3.20]).

```
ivmurashov@fedora:~/work/arch-pc/lab09 — /usr/bin/mc -P /var/tmp/mc-ivmuras...
Файл Правка Вид Поиск Терминал Справка

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd0a0 0xffffd0a0
ebp      0x0      0x0

0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
> 0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008

native process 5469 In: _start L14 PC: 0x8049016
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd0a0 0xffffd0a0
ebp      0x0      0x0
esi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 3.20: Получение информации о точках останова в оболочке GDB

Выполняю 5 инструкций с помощью команды `stepi` (или `si`) (рис. [3.21]).

```
ivmurashov@fedora:~/work/arch-pc/lab09 — /usr/bin/mc -P /var/tmp/mc-ivmuras...
Файл Правка Вид Поиск Терминал Справка

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd0a0 0xffffd0a0
ebp      0x0      0x0

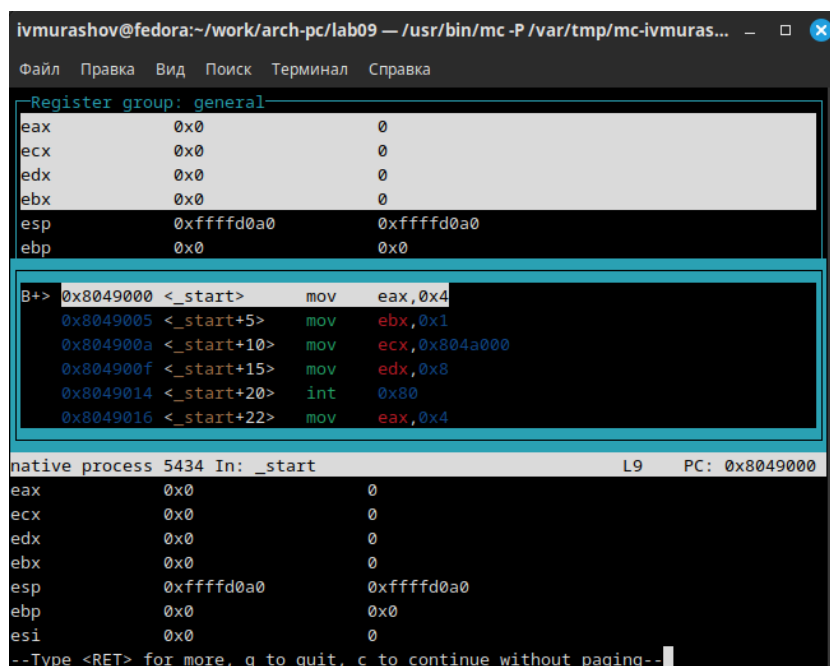
0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
> 0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008

native process 5377 In: _start L14 PC: 0x8049016
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) nDebuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab09-2.asm:9
(gdb) stepi 5
(gdb)
```

Рис. 3.21: Выполнение 5 шагов программы в оболочке GDB

Просматриваю содержимое регистров с помощью команды `info registers` (или `info r`) (рис. [3.22]).



The screenshot shows the GDB interface with the following content:

```
ivmurashov@fedora:~/work/arch-pc/lab09 — /usr/bin/mc -P /var/tmp/mc-ivmuras...
Файл  Правка  Вид  Поиск  Терминал  Справка

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0a0 0xffffd0a0
ebp      0x0      0x0

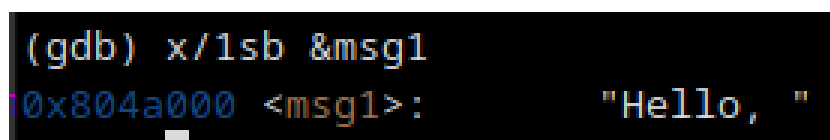
B> 0x8049000 <_start>      mov     eax,0x4
0x8049005 <_start+5>      mov     ebx,0x1
0x804900a <_start+10>     mov     ecx,0x804a000
0x804900f <_start+15>     mov     edx,0x8
0x8049014 <_start+20>     int     0x80
0x8049016 <_start+22>     mov     eax,0x4

native process 5434 In: _start L9 PC: 0x8049000
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0a0 0xffffd0a0
ebp      0x0      0x0
esi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 3.22: Получение информации о точках останова в оболочке GDB

Изменились значения регистров `eax`, `ecx`, `edx`, `ebx`.

Просматриваю значение переменной `msg1` по имени (рис. [3.23]).



The screenshot shows the GDB interface with the following content:

```
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
```

Рис. 3.23: Работа с переменными в оболочке GDB

Просматриваю значение переменной `msg2` по адресу, определяя его по дисассемблированной инструкции (рис. [3.24]).

```
0x804900a <_start+10>  mov     ecx,0x804a000
0x804900f <_start+15>  mov     edx,0x8
0x8049014 <_start+20>  int     0x80
> 0x8049016 <_start+22>  mov     eax,0x4
0x804901b <_start+27>  mov     ebx,0x1
0x8049020 <_start+32>  mov     ecx,0x804a008

native process 3417 In: _start L14 PC: 0x8049016
ebp      0x0      0x0
esi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--qQuit
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb)
```

Рис. 3.24: Работа с переменными в оболочке GDB

Изменяю первый символ переменной msg1 и просматриваю значение msg1 (рис. [3.25]).

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

Рис. 3.25: Работа с переменными в оболочке GDB

Заменяю 3 символ в переменной msg2 и просматриваю значение msg2 (рис. [3.26]).

```
(gdb) set {char}0x804a00a='W'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "woWld!\n\034"
```

Рис. 3.26: Работа с переменными в оболочке GDB

Вывожу в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx (рис. [3.27]).



```
(gdb) p/x $edx
$1 = 0x8

(gdb) p/t $edx
$2 = 1000

(gdb) p/c $edx
$3 = 8 '\b'

(gdb) █
```

Рис. 3.27: Работа с регистрами в оболочке GDB

С помощью команды set изменяю значение регистра ebx (рис. [3.28]).

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
```

Рис. 3.28: Работа с регистрами в оболочке GDB

При изменении значения регистра ebx на '2' мы переводим символ в строковый вид (по таблице ASCII символ '2' соответствует 50 а десятичном представлении), а при изменении на 2 число остаётся в строковом виде.

Завершаю выполнение программы с помощью команды continue (сокращенно c) и выхожу из GDB с помощью команды quit (сокращенно q) (рис. [3.29]).

```
(gdb) continue
Continuing.
[Inferior 1 (process 3417) exited normally]
```

Рис. 3.29: Работа в оболочке GDB

### 3.5 Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8 с программой, выводящей на экран аргументы командной строки (Листинг

8.2) в файл с именем lab09-3.asm (рис. [3.30]).

```
[ivmurashov@fedora ~]$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
```

Рис. 3.30: Копирование файла

Создаю исполняемый файл (рис. [3.31]).

```
[ivmurashov@fedora lab09]$ nasm -f elf -g -l lab09-3.lst lab09-3.asm  
[ivmurashov@fedora lab09]$ ld -m elf_i386 -o lab09-3 lab09-3.o
```

Рис. 3.31: Трансляция и компоновка файлов

Загружаю исполняемый файл в отладчик, указав аргументы с помощью ключа `-args` (рис. [3.32]).

```
[ivmurashov@fedora lab09]$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'  
GNU gdb (GDB) Fedora Linux 13.2-6.fc38  
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. 3.32: Загрузка исполняемого файла в отладчик gdb

Устанавливаю точку останова перед первой инструкцией в программе и запускаю её (рис. [3.33]).

```

ivmurashov@fedora:~/work/arch-pc/lab09 — /usr/bin/mc -P /var/tmp/mc-ivmurashov...
Файл Правка Вид Поиск Терминал Справка

B+> 0x80490e8 <_start> pop ecx
0x80490e9 <_start+1> pop edx
0x80490ea <_start+2> sub ecx,0x1
0x80490ed <next> cmp ecx,0x0
0x80490f0 <next+3> je 0x80490fa <_end>
0x80490f2 <next+5> pop eax
0x80490f3 <next+6> call 0x804902d <sprintfLF>
0x80490f8 <next+11> loop 0x80490ed <next>
0x80490fa <_end> call 0x80490db <quit>
0x80490ff sbb al,0x0
0x8049101 add BYTE PTR [eax],al
0x8049103 add al,BYTE PTR [eax]
0x8049105 add BYTE PTR [eax],al

native process 4675 In: _start L8 PC: 0x80490e8
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 8.
(gdb) run
Starting program: /home/ivmurashov/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2
аргумент\ 3

Breakpoint 1, _start () at lab09-3.asm:8
(gdb)

```

Рис. 3.33: Установка точки останова и запуск программы в оболочке GDB

Просматриваю позиции стека (рис. [3.34]).

```

0xffffd060: 0x05
(gdb) x/s *(void**)(esp + 4)
0xffffd247: "/home/ivmurashov/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd273: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd285: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd296: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd298: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>

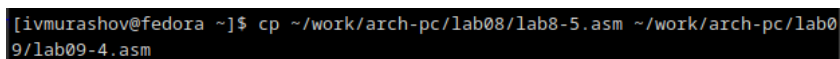
```

Рис. 3.34: Просмотр позиций стека в оболочке GDB

Шаг изменения адреса равен 4, так как

## 3.6 Выполнение заданий для самостоятельной работы

1. Копирую файл lab8-5.asm в каталог lab09 с новым именем lab09-4.asm (рис. [3.35]).

A terminal window with a black background and white text. The prompt is [ivmurashov@fedora ~]\$. The command being entered is cp ~/work/arch-pc/lab08/lab8-5.asm ~/work/arch-pc/lab09/lab09-4.asm.

```
[ivmurashov@fedora ~]$ cp ~/work/arch-pc/lab08/lab8-5.asm ~/work/arch-pc/lab09/lab09-4.asm
```

Рис. 3.35: Копирование файла

Изменяю программу, реализовав вычисление значения функции  $f(x)$  как подпрограмму (рис. [3.36]).

```
lab09-4.asm (~/work/arch-pc/lab09)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка

_ start:
pop ecx ; Извлекаем из стека в ecx количество
        ; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в edx имя программы
        ; (второе значение в стеке)
sub ecx, 1 ; Уменьшаем ecx на 1 (количество
        ; аргументов без названия программы)
mov esi, 0 ; Используем esi для хранения
        ; промежуточных произведений

next:
cmp ecx, 0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
        ; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
        ; след. аргумент esi=esi*eax
call _calcul
loop next ; переход к обработке следующего аргумента

_end:
mov eax, msg ; вывод сообщения "Функция: f(x) = 8*x - 3 "
call sprintf
mov eax, msg1 ; вывод сообщения "Результат: "
call sprintf
mov eax, esi ; записываем произведение в регистр eax
call iprintfLF ; печать результата
call quit ; завершение программы

_calcul:
mov ebx, 8
mul ebx ; "eax = eax*ebx = x*8"
add eax, 3 ; "eax = eax + 3 = 8*x - 3"
sub esi, eax
ret
```

Рис. 3.36: Редактирование файла

**Листинг 3. Программа вычисления суммы функций для аргументов командной строки**

```

%include 'in_out.asm'

SECTION .data
msg db "Функция: f(x) = 8*x - 3",0
msg1 db "Результат: ",0

SECTION .text
global _start

_start:

pop ecx ; Извлекаем из стека в ecx количество
        ; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в edx имя программы
        ; (второе значение в стеке)
sub ecx,1 ; Уменьшаем ecx на 1 (количество
        ; аргументов без названия программы)
mov esi, 0 ; Используем esi для хранения
        ; промежуточных произведений

next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
        ; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
        ; след. аргумент esi=esi*eax
call _calcul
loop next ; переход к обработке следующего аргумента

_end:
mov eax, msg ; вывод сообщения "Функция: f(x) = 8*x - 3 "

```

```

call sprintf
mov eax, msg1 ; вывод сообщения "Результат: "
call sprintf
mov eax, esi ; записываем произведение в регистр eax
call iprintLF ; печать результата
call quit ; завершение программы

```

```

_calcul:
mov ebx, 8
mul ebx ; "eax = eax*ebx = x*8"
sub eax, 3 ; "eax = eax - 3 = 8*x - 3"
add esi, eax
ret

```

Проверяю корректность работы программы (рис. [3.37]).

```

[ivmurashov@fedora lab09]$ nasm -f elf lab09-4.asm
[ivmurashov@fedora lab09]$ ld -m elf_i386 -o lab09-4 lab09-4.o
[ivmurashov@fedora lab09]$ ./lab09-4 1 2 3
Функция: f(x) = 8*x - 3
Результат: 39

```

Рис. 3.37: Трансляция, компоновка и запуск файлов

Программа работает корректно.

2. Создаю файл lab09-5.asm в каталоге lab09 (рис. [3.38]).

```

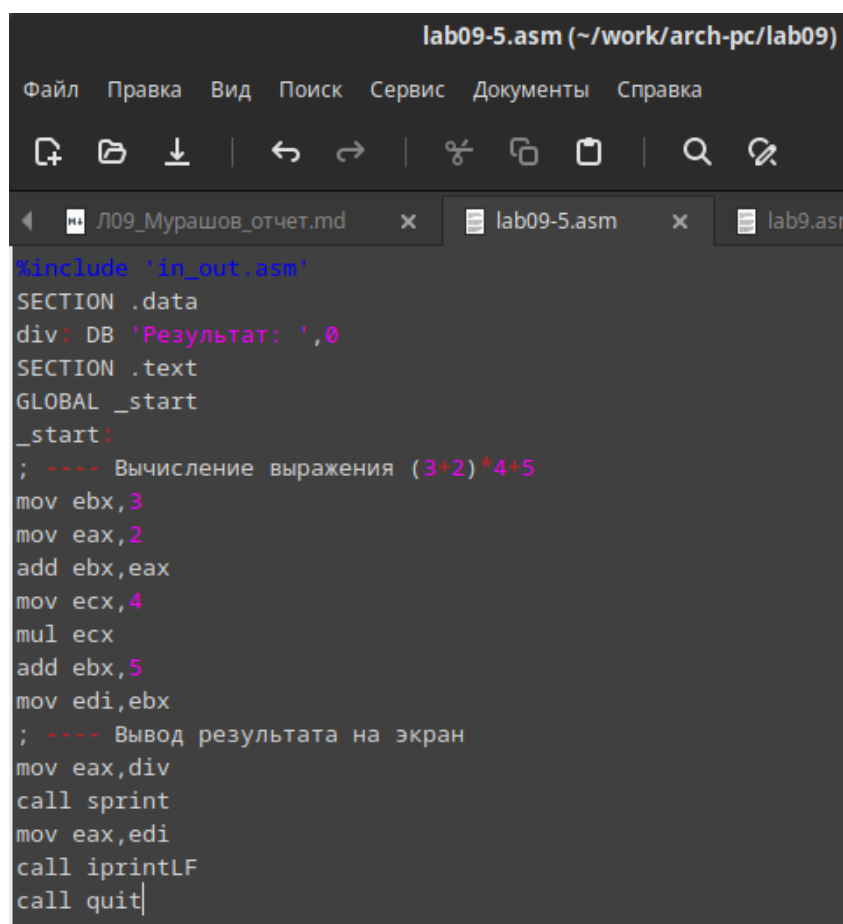
[ivmurashov@fedora lab09]$ touch lab09-5.asm

```

Рис. 3.38: Создание файла

Ввожу в данный файл программу вычисления выражения  $(3+2)^4+5$  из листинга 9.3 (рис. [3.39]).





```
lab09-5.asm (~/.work/arch-pc/lab09)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Icons]
Л09_Мурашов_отчет.md  lab09-5.asm  lab9.asm

%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```

Рис. 3.39: Редактирование файла

**\*\*Листинг 4. Программа вычисления выражения  $(3+2)*4+5$ \*\***

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
```

```

add ebx, eax
mov ecx, 4
mul ecx
add ebx, 5
mov edi, ebx
; ---- Вывод результата на экран
mov eax, div
call sprint
mov eax, edi
call iprintLF
call quit

```

Получаю исполняемый файл (рис. [3.40]).

```

[ivmurashov@fedora lab09]$ nasm -f elf -g -l lab09-5.lst lab09-5.asm
[ivmurashov@fedora lab09]$ ld -m elf_i386 -o lab09-5 lab09-5.o

```

Рис. 3.40: Трансляция, компоновка и запуск файлов

Загружаю исполняемый файл в отладчик gdb (рис. [3.41]).

```

[ivmurashov@fedora lab09]$ gdb lab09-5
GNU gdb (GDB) Fedora Linux 13.2-6.fc38
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 lab09-5...

```

Рис. 3.41: Загрузка исполняемого файла в отладчик gdb

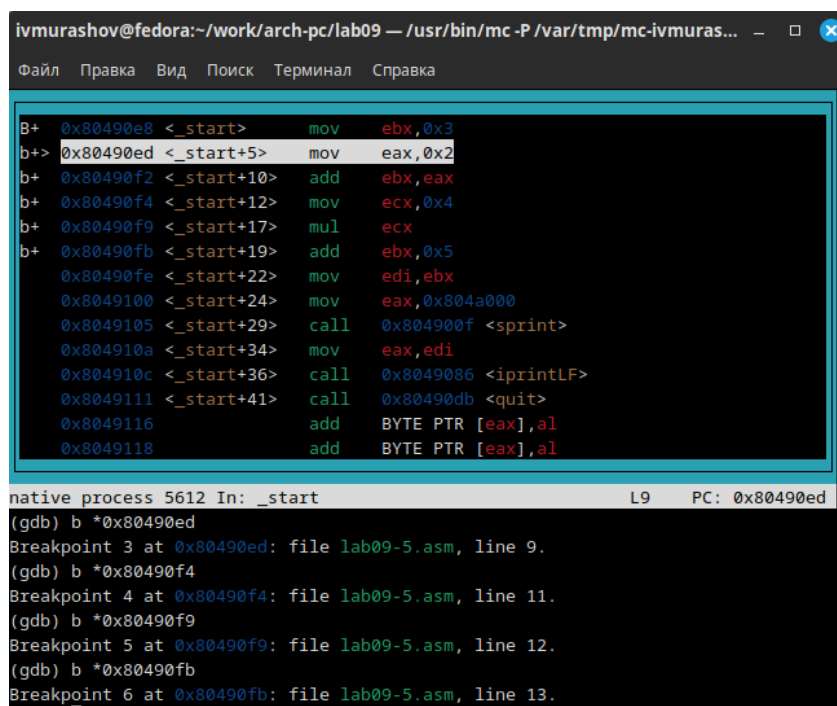
Запускаю программы. рис. [3.42]).

```
(gdb) run
Starting program: /home/ivmurashov/work/arch-pc/lab09/lab09-5
Результат: 10
[Inferior 1 (process 5449) exited normally]
```

Рис. 3.42: Запуск программы в оболочке GDB

В результате выводится '10', а должно получиться '25'. Следовательно, программа работает некорректно.

Устанавливаю брейкпоинты во всех местах, где происходит вычисление значений (рис. [3.43]).



```
ivmurashov@fedora:~/work/arch-pc/lab09 — /usr/bin/mc -P /var/tmp/mc-ivmuras...
Файл  Правка  Вид  Поиск  Терминал  Справка

B+  0x80490e8 <_start>      mov     ebx, 0x3
b+> 0x80490ed <_start+5>    mov     eax, 0x2
b+  0x80490f2 <_start+10>   add     ebx, eax
b+  0x80490f4 <_start+12>   mov     ecx, 0x4
b+  0x80490f9 <_start+17>   mul     ecx
b+  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

native process 5612 In: _start                               L9      PC: 0x80490ed
(gdb) b *0x80490ed
Breakpoint 3 at 0x80490ed: file lab09-5.asm, line 9.
(gdb) b *0x80490f4
Breakpoint 4 at 0x80490f4: file lab09-5.asm, line 11.
(gdb) b *0x80490f9
Breakpoint 5 at 0x80490f9: file lab09-5.asm, line 12.
(gdb) b *0x80490fb
Breakpoint 6 at 0x80490fb: file lab09-5.asm, line 13.
```

Рис. 3.43: Установка брейкпоинтов в оболочке GDB

С помощью команды continue (сокращённо 'c') просматриваю значения и нахожу ошибки (рис. [3.44]).

```
B+> 0x80490e8 <_start> mov ebx,0x3
b+ 0x80490ed <_start+5> mov eax,0x2
b+ 0x80490f2 <_start+10> add ebx,eax
b+ 0x80490f4 <_start+12> mov ecx,0x4
b+ 0x80490f9 <_start+17> mul ecx
b+ 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

native process 7771 In: _start L8 PC: 0x80490e8

Breakpoint 3, _start () at lab09-5.asm:9
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) yStarting program: /home/ivmurashov/work/arch-pc/lab09/lab09-5

Breakpoint 1, _start () at lab09-5.asm:8
(gdb) |
```

Рис. 3.44: Работа с переменными в оболочке GDB

При выполнении инструкции `mul` значение регистра `ecx` умножается на значение регистра `eax`. Добавляю инструкцию `mov eax, ebx`. Изменяю инструкцию `add ebx, 5` на `add eax, 5` и `mov edi, ebx` на `mov edi, eax` (рис. [3.45]).



## 4 Выводы

В ходе выполнения данной лабораторной работы я приобрёл навыки написания программ с использованием подпрограмм, познакомился с методами отладки при помощи GDB и его основными возможностями.