РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

дисциплина: Архитектура компьютера

Студент: Безходарнова А.В

Группа: НКАбд-01-25

МОСКВА

2025 г.

СОДЕРЖАНИЕ

1.Цель работы

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

2. Выполнение лабораторной работы

2.1 Реализация подпрограмм в NASM

Создаю каталог для программам лабораторной работы №9, перейдя в него создаю файл lab9-1.asm:.(рис.2.1.1)

```
avbezkhodarnova1@avbezkhodarnova1:~$ mkdir ~/work/arch-pc/lab09
avbezkhodarnova1@avbezkhodarnova1:~$ cd ~/work/arch-pc/lab09
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ touch lab9-1.asm
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$
```

Рис. 2.1.1. Создание файла

В созданном файле ввожу программу из листинга. (рис.2.1.3)

Open • lab9-1.asm
~/work/arch-pc/lab09

```
%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 ; Вызов подпрограммы _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. 2.1.2 Загрузка текста из листинга

Далее запускаю исходный код, который вычисляет значение функции. (рис.2.1.3)

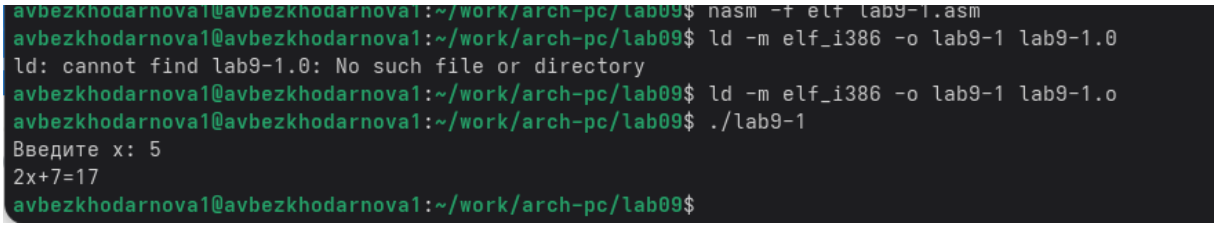

Рис. 2.1.3 Запуск программы

Изменяю текст программы, добавив подпрограмму _subcalcul в подпрограмму _calcul, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$. (Рис.2.1.4) и (рис.2.1.5)

Open • lab9-1.asm
~/work/arch-pc/lab09

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ', 0
result: DB '2(3x-1)+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:
push eax
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
pop eax
ret
_subcalcul:
mov ebx, 3
mul ebx
```

Рис.2.1.4 Изменение текста

```
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 5
2(3x-1)+7=35
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$
```

Рис.2.1.5 Запуск программы

2.2 Отладка программам с помощью GDB

Создаю новый файл и ввожу в него код из листинга. (рис. 2.2.1)

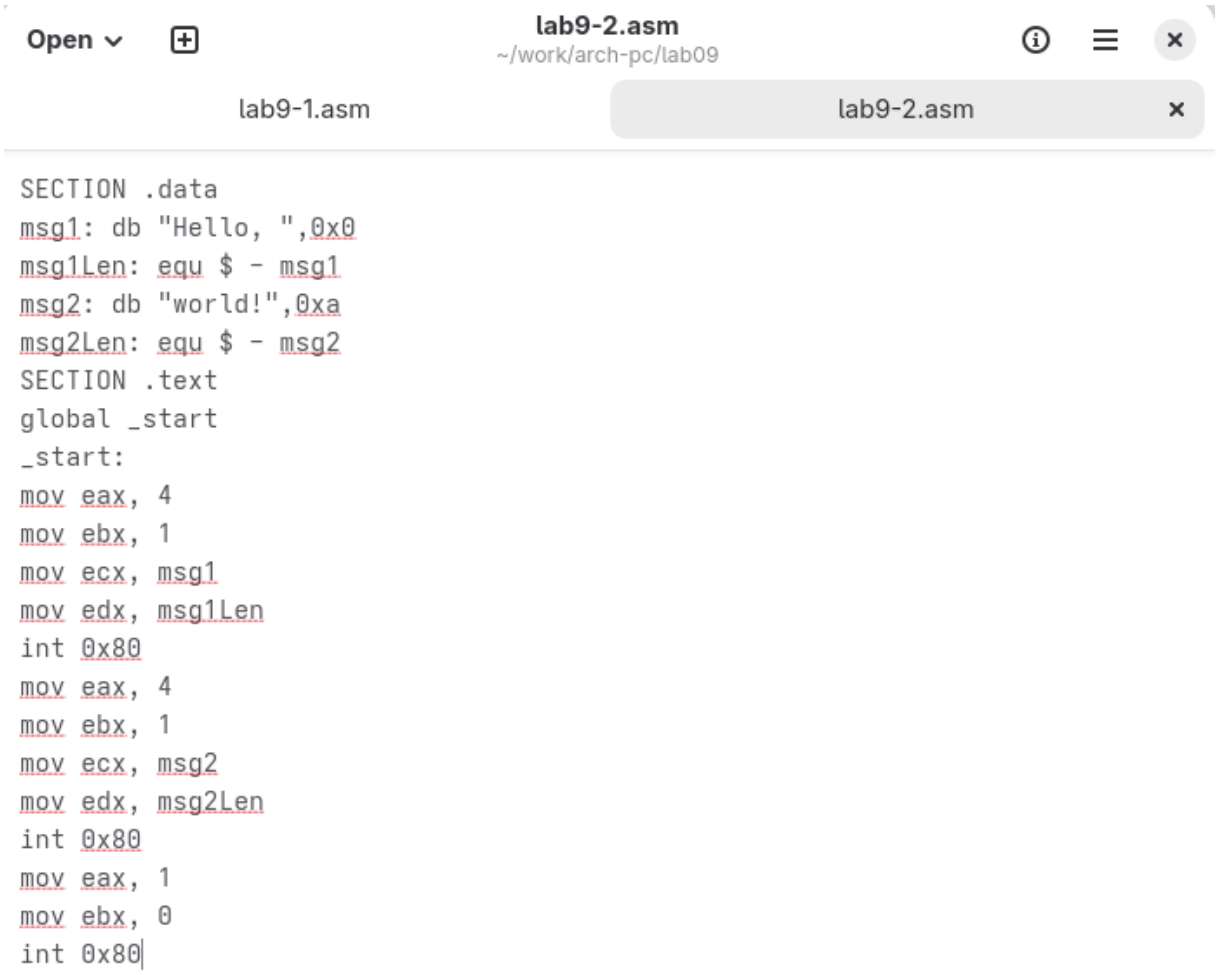

Рис.2.2.1 Ввод кода из листинга.

Запускаю программу и получаю исполняемый файл. (Рис.2.2.2)

```
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 16.2-3.fc42
Copyright (C) 2024 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 lab9-2...
(gdb)
```

Рис. 2.2.2 Запуск программы

Проверяю работу программы, запуская ее с помощью run. (Рис.2.2.3)

```
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 16.2-3.fc42
Copyright (C) 2024 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 lab9-2...
(gdb) run
Starting program: /home/avbezkhodarnova1/work/arch-pc/lab09/lab9-2

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 7351) exited normally]
(gdb)
```

Рис. 2.2.3 Работа программы.

Далее ставлю брейкпоинт для корректной работы программы и запускаю. (Рис.2.2.4)

```
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 lab9-2...
(gdb) run
Starting program: /home/avbezkhodarnova1/work/arch-pc/lab09/lab9-2

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 7351) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8048080: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/avbezkhodarnova1/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9       mov eax, 4
(gdb)
```

Рис.2.2.4 Запуск программы.

Далее я смотрю дисассимилированный код программы с помощью команды disassemble, а после переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду setdisassembly-flavor intel. (рис.2.2.5) 1.

Различия между синтаксисом ATT и Intel заключаются в следующем: в синтаксисе AT&T первым указывается исходный операнд, а вторым — целевой. Например, инструкция addl %eax, %ebx означает сложение содержимого регистра EAX с регистром EBX и сохранение результата в EBX. В синтаксисе Intel порядок обратный: сначала указывается операнд-назначение, затем операнд-источник. Та же операция записывается как add

ebx, eax. В ATT размер обрабатываемых данных задаётся явно через суффиксы в мнемонике команды. В Intel размер данных определяется неявно — по используемым регистрам или явным указанием типа (например, byte ptr, dword ptr). В ATT непосредственные значения предваряются символом $. Например, movl $100, %eax загружает число 100 в регистр EAX. В Intel непосредственные значения записываются без дополнительных символов: mov eax, 100. В ATT для обращения к памяти используются круглые скобки: (%eax). В Intel адресация памяти обозначается квадратными скобками: [eax], а сложные адреса имеют вид [база + индекс * множитель + смещение].

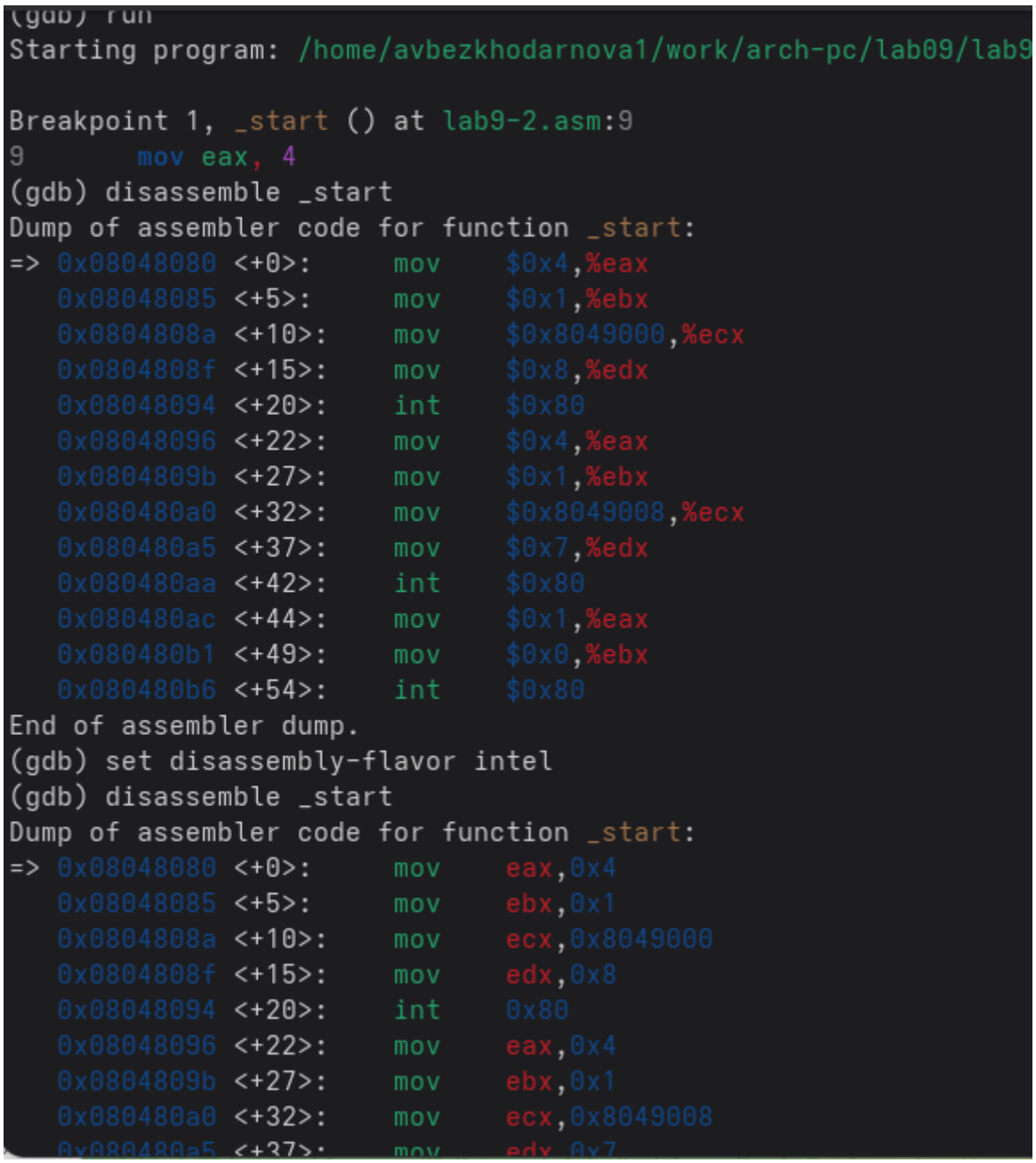

Рис.2.2.5 Изменение текста листинга.

Включаю режим псевдографики. (Рис.2.2.6)

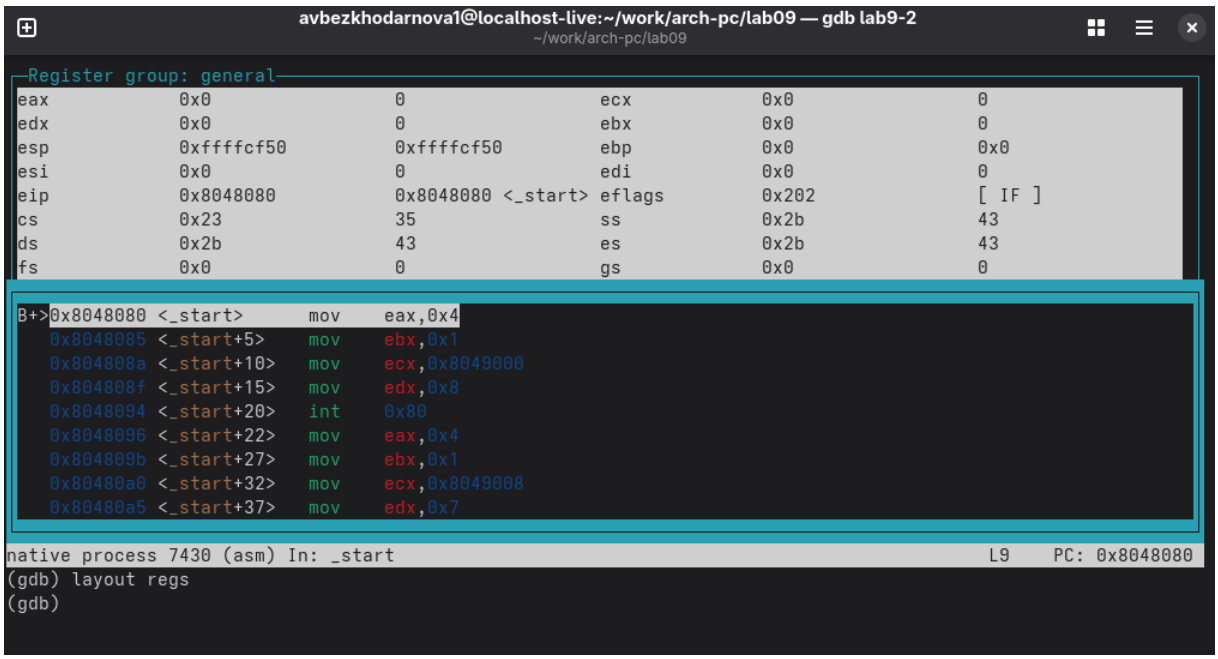

Рис.2.2.6 Режим псевдографики.

2.3 Добавление точек останова

Проверяю, установлена ли точка останова с помощью команды info breakpoints. (Рис.2.3.1)

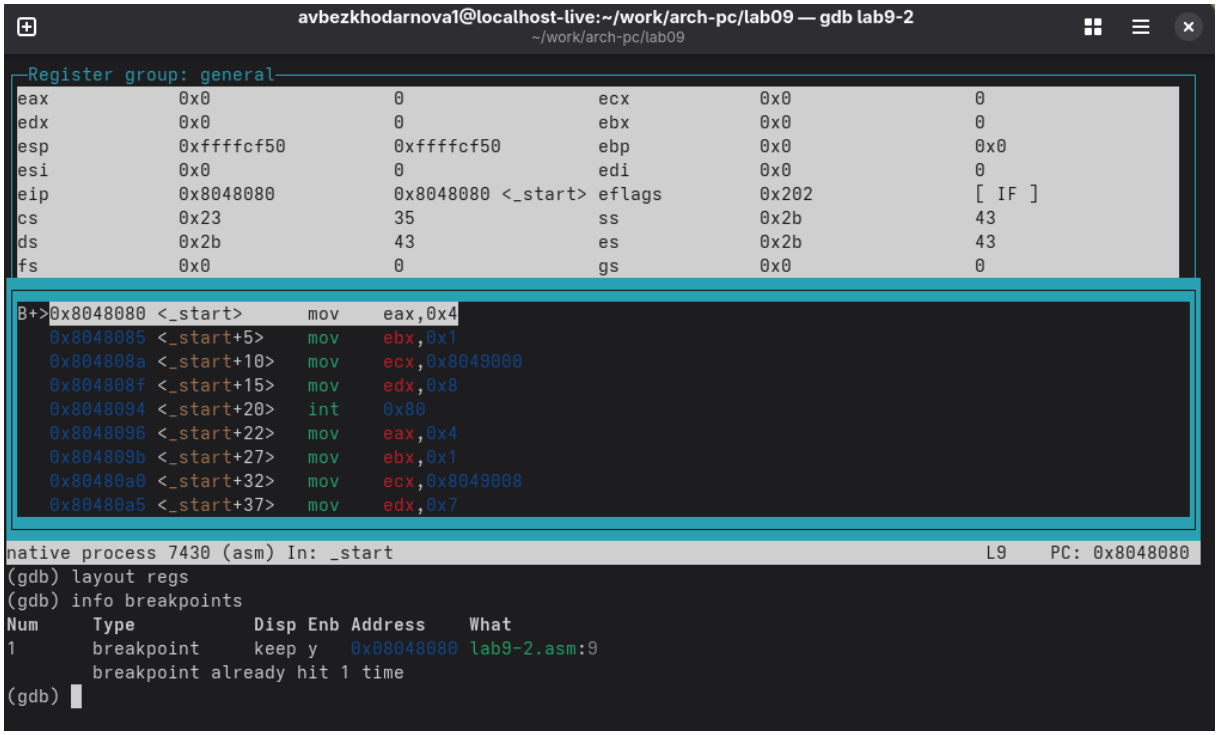

Рис.2.3.1 Проверка установки

Далее устанавливаю еще одну точку останова по адресу. Для этого смотрю, адрес инструкции и задаю точку останова. (рис.2.3.2)

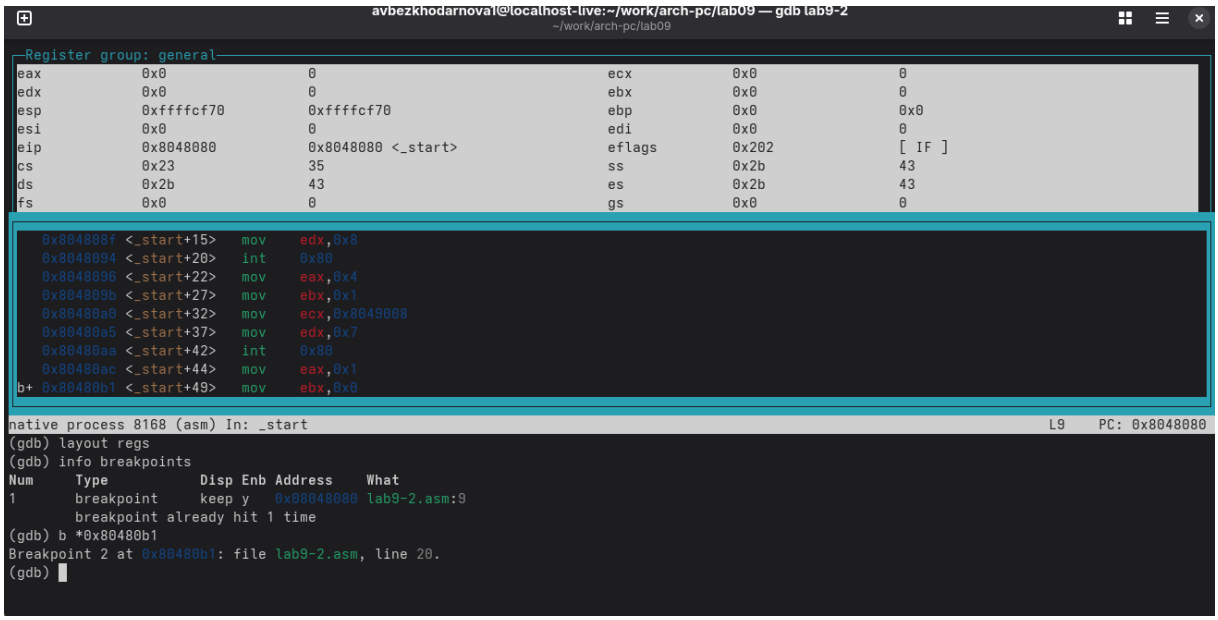

Рис.2.3.2 задача точек останова

2.4 Работа с данными программы в GDB

Смотрю содержимое регистров с помощью команды. (рис.2.4.1)

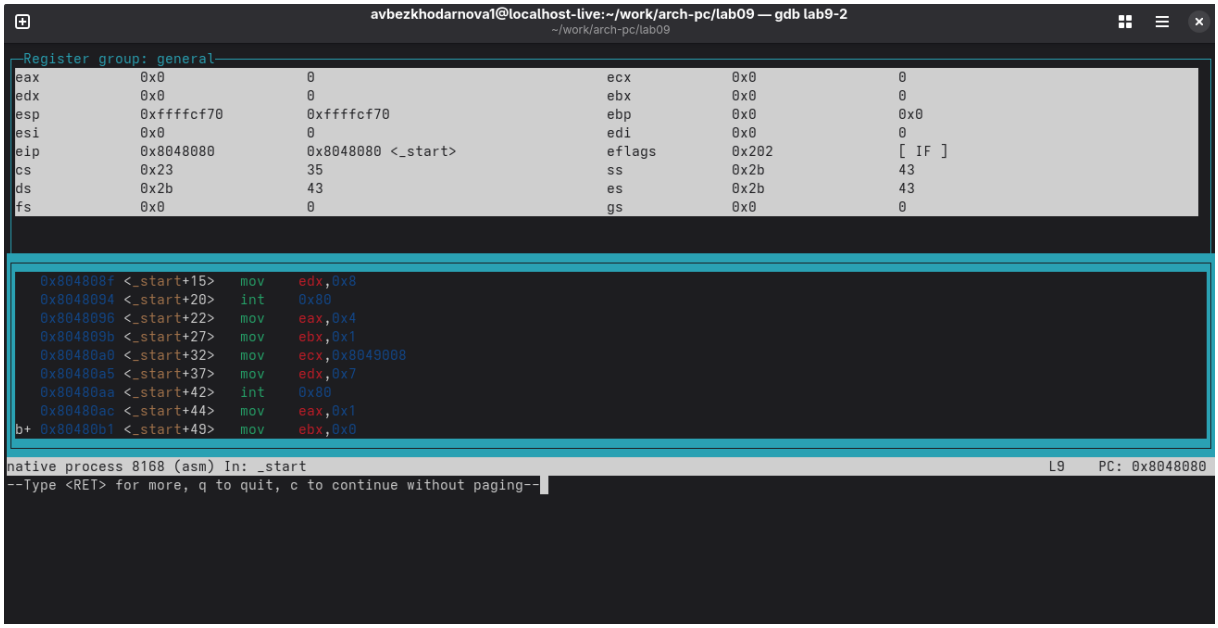

Рис.2.4.1 просмотр регистров

Смотрю содержимое переменной. (Рис.2.4.2)

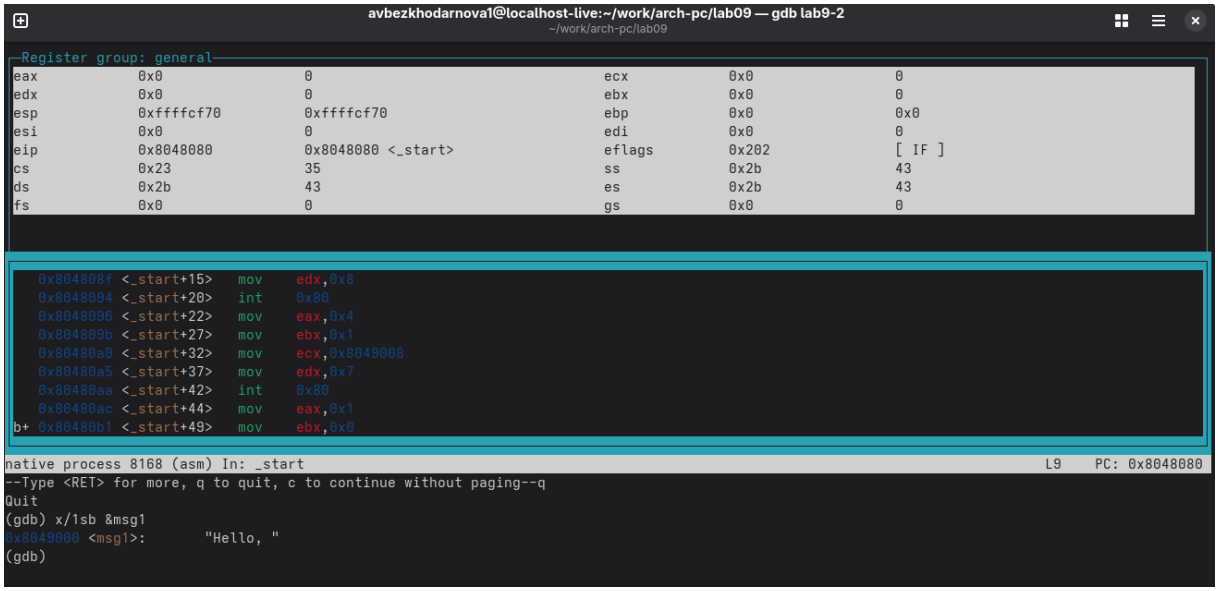

Рис.2.4.2 просмотр содержимого переменной

Далее я смотрю адрес переменной, но теперь определяю Адрес переменной можно по дизассемблированной инструкции. (Рис.2.4.3)

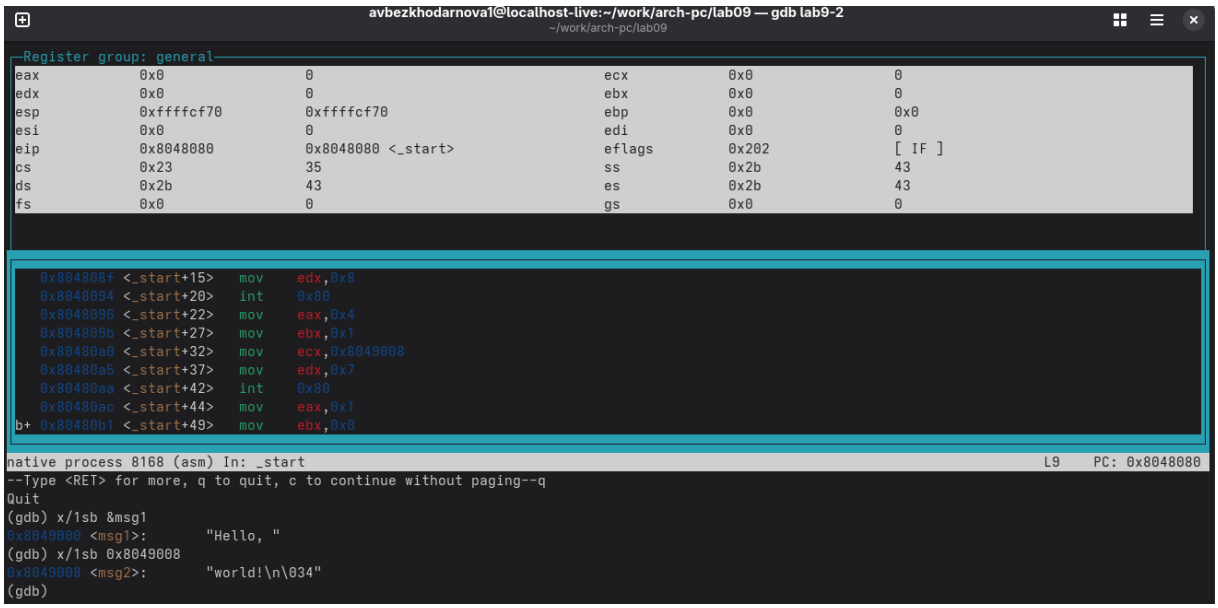

Рис.2.4.3 Просмотр содержимого переменной.

Изменяю значение для регистра или ячейки памяти с помощью команды set. (рис.2.4.4)

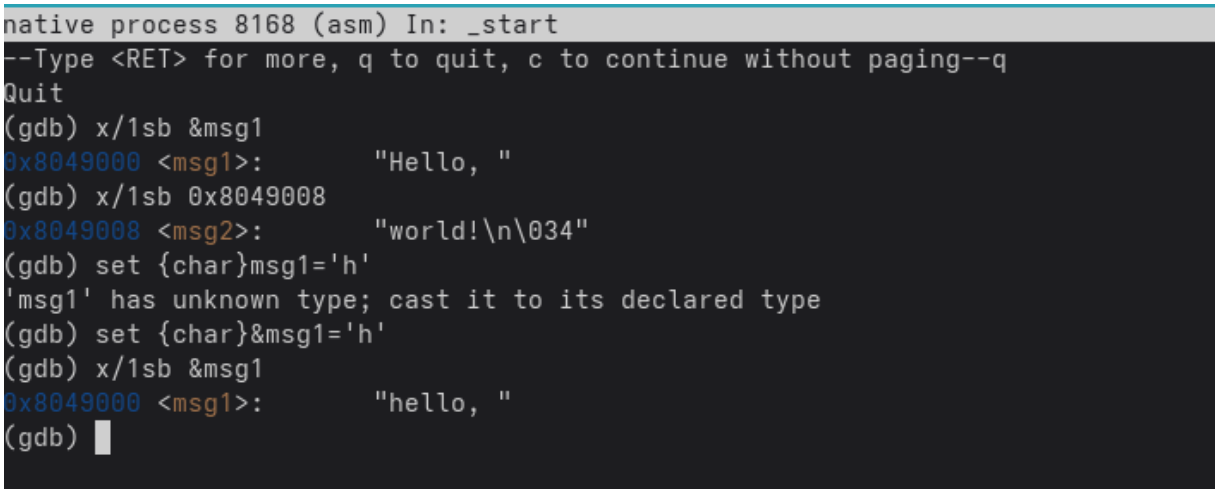

Рис.2.4.4 Изменение значение регистра.

Меняю символ в переменной msg2. (Рис.2.4.5)

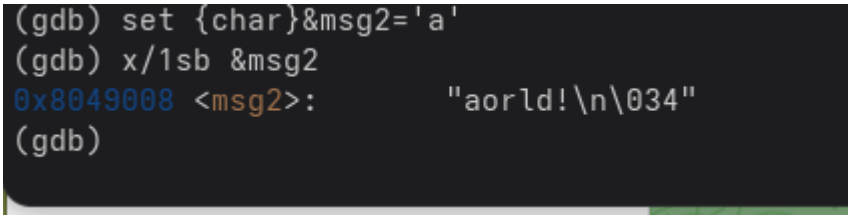

Рис.2.4.5 Изменение символа в переменной

Вывожу в различных форматах значение регистра edx. (Рис.2.4.6)

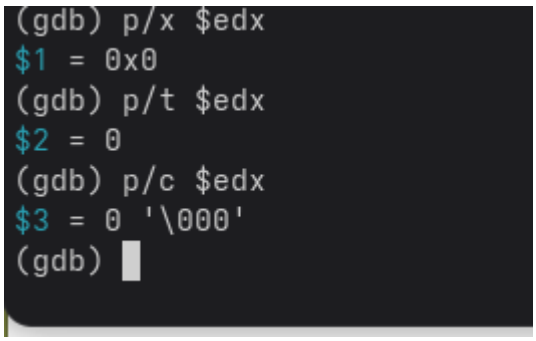

Рис.2.4.6 Значения регистра

Далее с помощью команды set изменяю значения регистра ebx. (Рис.2.4.7) и (Рис.2.4.8)

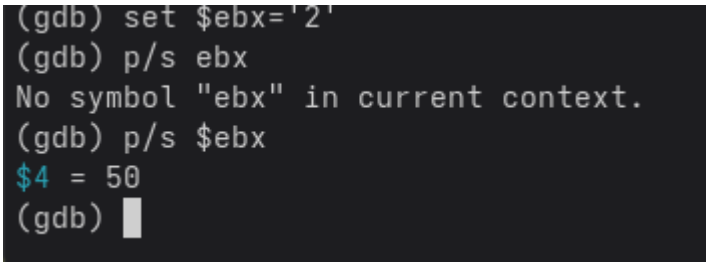

Рис.2.4.7 Изменение регистра

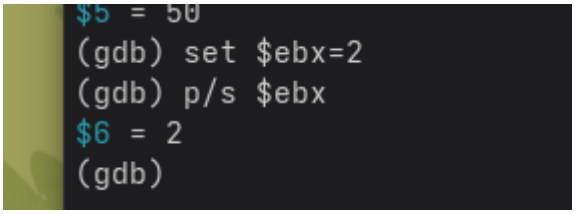

Рис.2.4.8 Изменение регистра

Кавычки '2' дают код символа (50), а просто 2 даёт число 2. Команда p/s показывает число, а не символ.

2.5 Обработка аргументов командной строки в GDB

Копирую файл из лабораторной работы №8 и создаю исполняемый файл, а также задаю аргументы, использую ключ. (Рис.2.5.1)

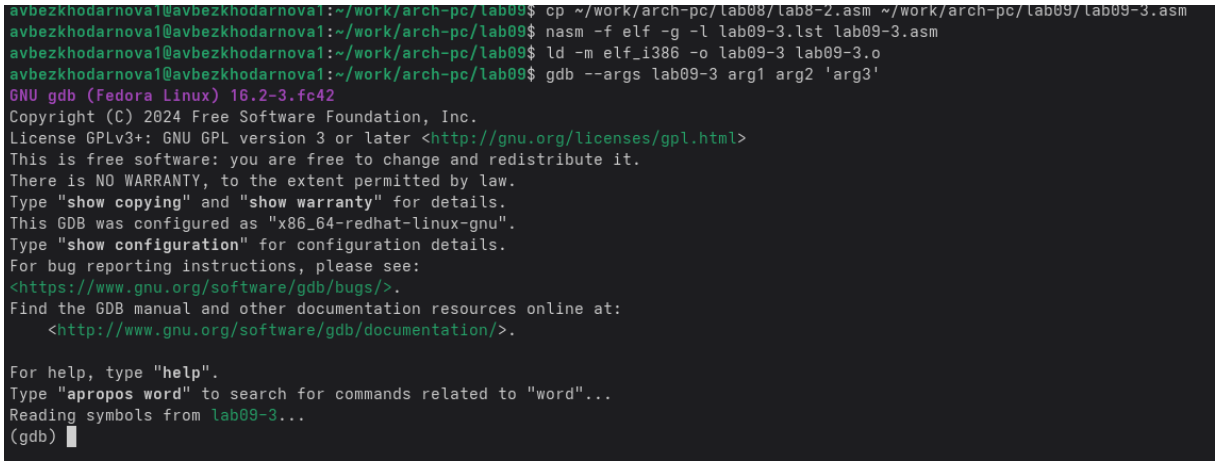

Рис.2.5.1 Копирование файла.

Далее устанавливаю точку останова перед первой инструкцией в программе и запускаю ее. (Рис.2.5.2)

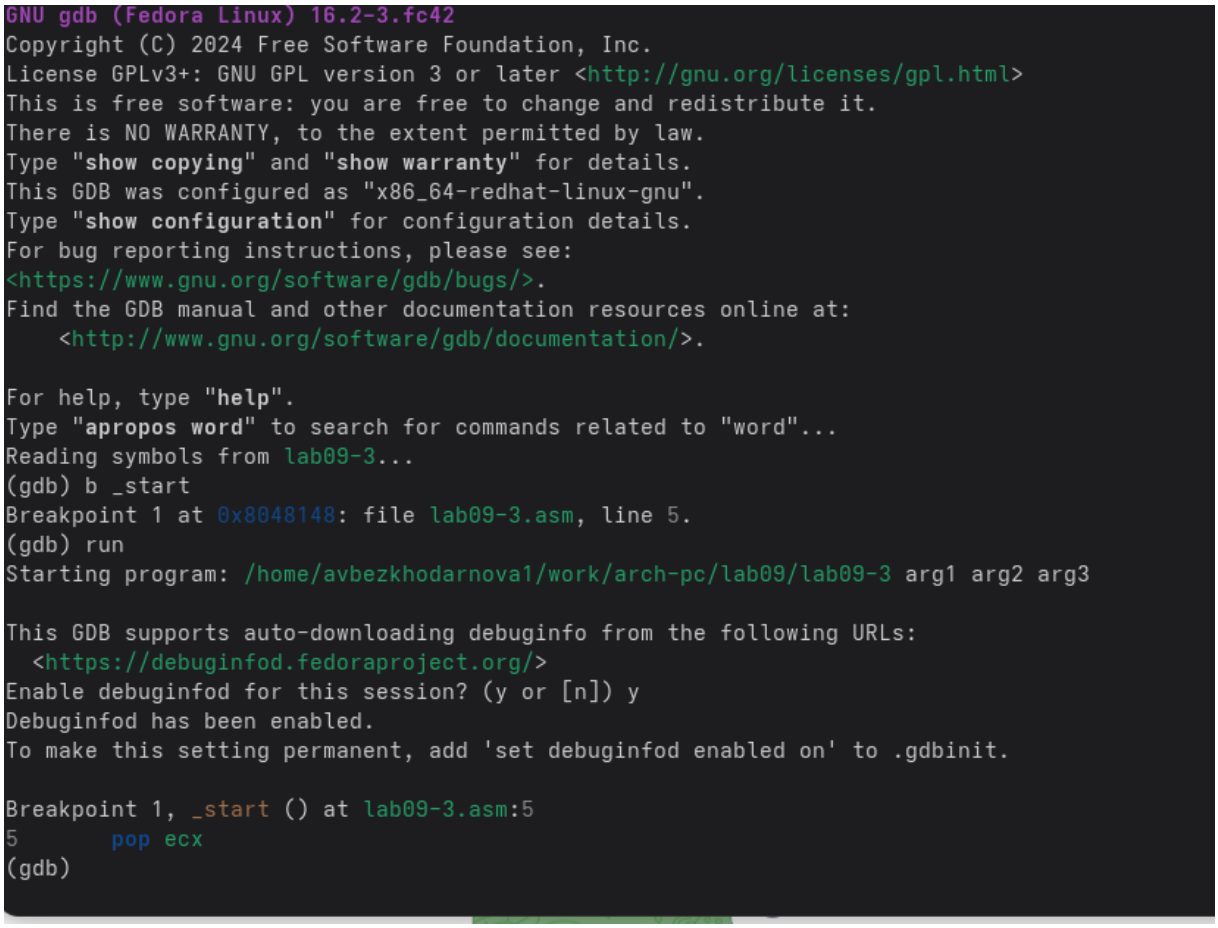

Рис.2.5.2 Запуск программы

Смотрю остальные позиции стека – по адесу [esp+4] располагается адрес в памяти где находиться имя программы, по адесу [esp+8] храниться адрес первого аргумента, по аресу [esp+12] – второго и т.д. (Рис.2.5.3)

```
0xffffcf50:     0x00000004
(gdb) x/x $esp+4
0xffffcf54:     0xffffd12e
(gdb) x/x $esp+8
0xffffcf58:     0xffffd160
(gdb) x/x $esp+12
0xffffcf5c:     0xffffd165
(gdb) x/s *(void**)($esp+4)
0xffffd12e:     "/home/avbezkhodarnova1/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)($esp+8)
0xffffd160:     "arg1"
(gdb) x/s *(void**)($esp+12)
0xffffd165:     "arg2"
(gdb) x/s *(void**)($esp+16)
0xffffd16a:     "arg3"
(gdb) x/s *(void**)($esp+20)
0x0:    <error: Cannot access memory at address 0x0>
(gdb) x/s *(void**)($esp+24)
0xffffd16f:     "SHELL=/bin/bash"
(gdb)
```

Рис.2.5.3 Проверка работы

3. Задание для самостоятельной работы

Преобразую программу из лабораторной работы №8, реализовав вычисление значения функции $f(x)$ как подпрограмму. (Рис.3.1)

Open • lab8-4.asm ~/work/arch-pc/lab09

lab9-1.asm | lab9-2.asm | • lab8-4.asm

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '4x-3=',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,4
mul ebx
sub eax,3
mov [res],eax
ret
```

Рис.3.1 Редактирование файла.

Далее я загружаю программу вычисления, с помощью отладчика нахожу ошибки и исправляю их. (Рис.3.2), (Рис.3.3), (Рис.3.4), (Рис.3.5)

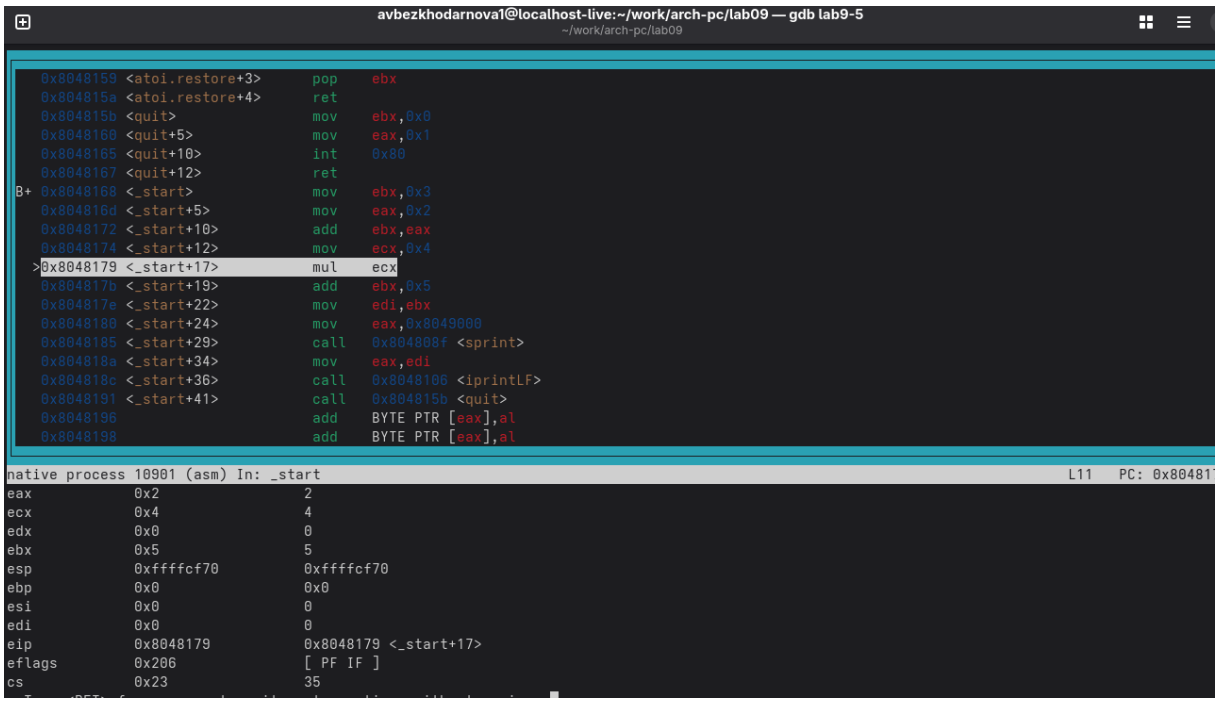

Рис.3.2 Запуск программы.

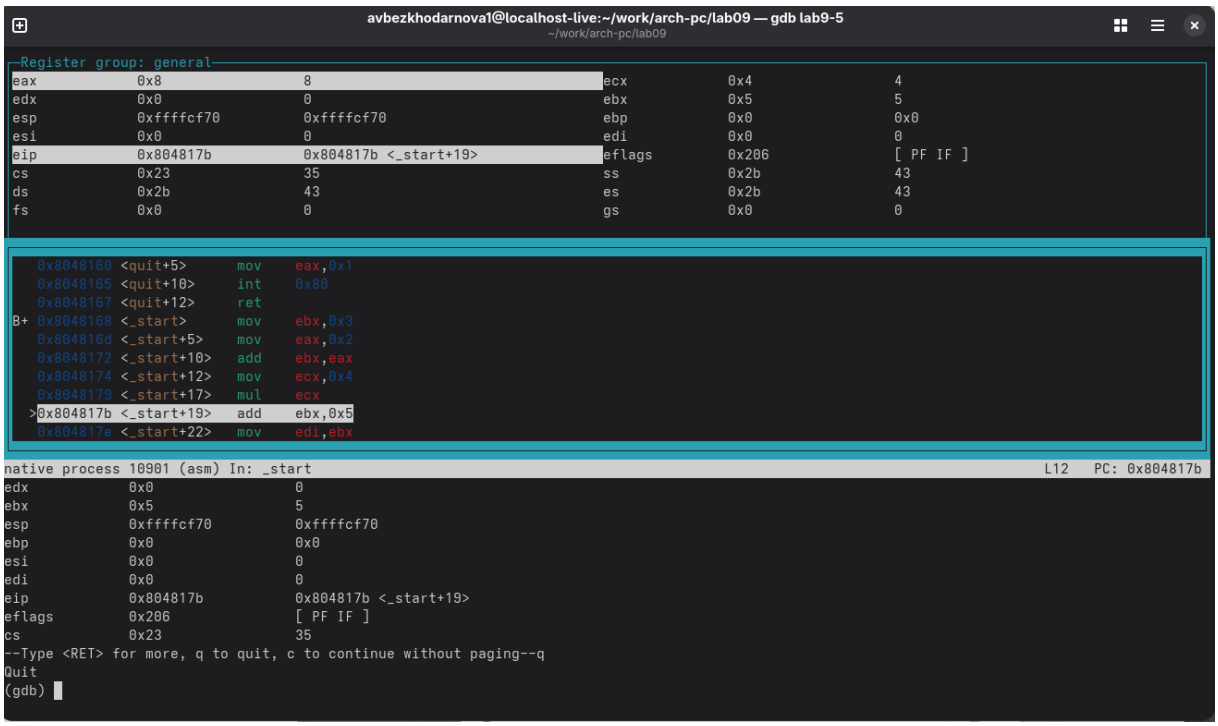

Рис.3.3

Open

lab9-5.asm
~/work/arch-pc/lab09

lab9-1.asm | lab9-2.asm | lab9-4.asm | lab9-5.asm

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
mov ebx,3
mov eax,2
add ebx,eax
mov 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.3.4

```
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ nasm -f elf lab9-5.asm
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-5 lab9-5.o
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$ ./lab9-5
Результат: 25
avbezkhodarnova1@avbezkhodarnova1:~/work/arch-pc/lab09$
```

Рис.3.5

4. Выводы

В ходе выполнения данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм и также познакомилась с методами отладки при помощи GDB и его основными возможностями.