

Отчет по лабораторной работе №9

Дисциплина: Архитектура компьютеров

Воронов Александр Валерьевич

Содержание

| | | |
|----------|---|-----------|
| 1 | Цель работы | 5 |
| 2 | Задание | 6 |
| 3 | Теоретическое введение | 7 |
| 4 | Выполнение лабораторной работы | 8 |
| 4.1 | Релаксация подпрограмм в NASM | 8 |
| 4.1.1 | Отладка программ с помощью GDB | 11 |
| 4.1.2 | Добавление точек останова | 15 |
| 4.1.3 | Работа с данными программы в GDB | 16 |
| 4.1.4 | Обработка аргументов командной строки в GDB | 18 |
| 4.2 | Задание для самостоятельной работы | 19 |
| 5 | Выводы | 24 |
| 6 | Список литературы | 25 |

Список иллюстраций

| | | |
|------|---|----|
| 4.1 | Создание рабочего каталога | 8 |
| 4.2 | Запуск программы из листинга | 9 |
| 4.3 | Изменение программы первого листинга | 9 |
| 4.4 | Запуск программы в отладчике | 12 |
| 4.5 | Проверка программы отладчиком | 12 |
| 4.6 | Запуск отладчика с брейкпоинтом | 13 |
| 4.7 | Дисассимилирование программы | 14 |
| 4.8 | Режим псевдографики | 14 |
| 4.9 | Список брейкпоинтов | 15 |
| 4.10 | Добавление второй точки останова | 15 |
| 4.11 | Просмотр содержимого регистров | 16 |
| 4.12 | Просмотр содержимого переменных двумя способами | 16 |
| 4.13 | Изменение содержимого переменных двумя способами | 17 |
| 4.14 | Просмотр значения регистра разными представлениями | 17 |
| 4.15 | Примеры использования команды set | 18 |
| 4.16 | Подготовка новой программы | 18 |
| 4.17 | Проверка работы стека | 19 |
| 4.18 | Измененная программа предыдущей лабораторной работы | 20 |
| 4.19 | Поиск ошибки в программе через пошаговую отладку | 22 |
| 4.20 | Проверка корректировок в программе | 22 |

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. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Самостоятельное выполнение заданий по материалам лабораторной работы

3 Теоретическое введение

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, отрабатывает, 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).

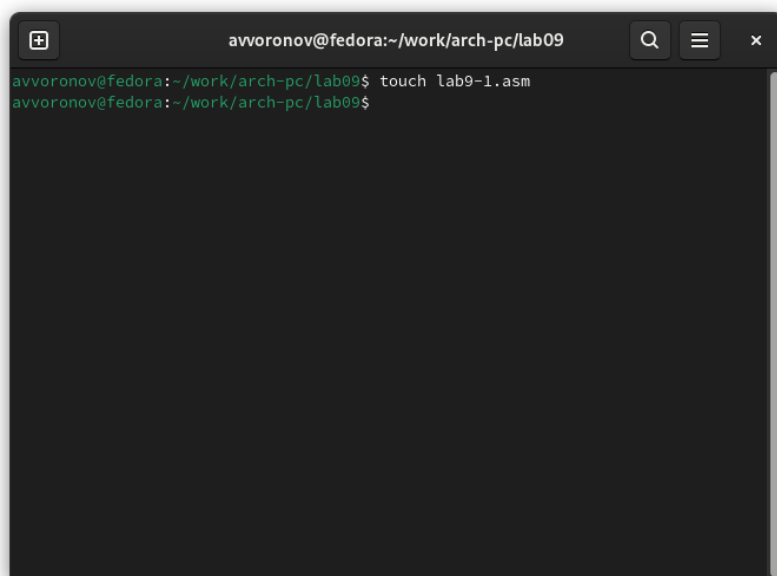
Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

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.

4 Выполнение лабораторной работы

4.1 Релаксация подпрограмм в NASM

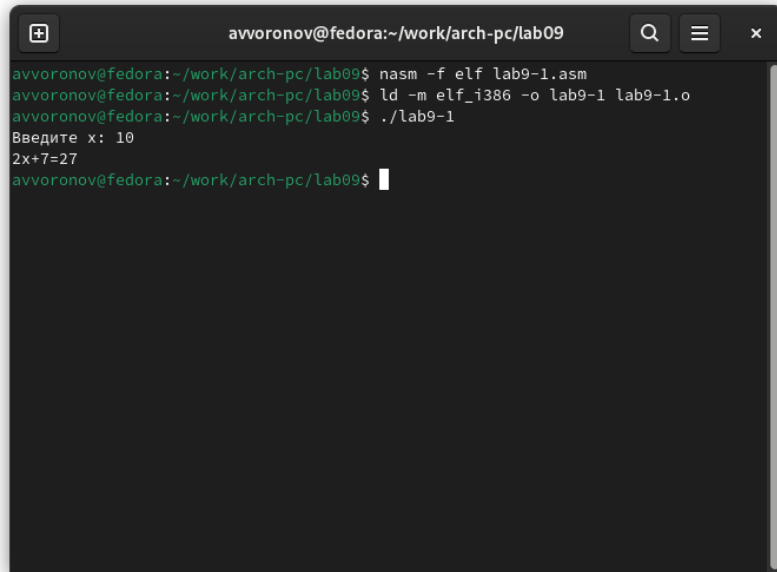
Создаю каталог для выполнения лабораторной работы №9 (рис. -fig. 4.1).

A terminal window with a dark background. The title bar shows the user 'avvoronov' on a 'fedora' machine in the directory '~/work/arch-pc/lab09'. The terminal shows two commands: 'touch lab9-1.asm' and its execution. The prompt changes from '\$' to '\$#' after the command is run.

```
avvoronov@fedora:~/work/arch-pc/lab09$ touch lab9-1.asm
avvoronov@fedora:~/work/arch-pc/lab09$
```

Рис. 4.1: Создание рабочего каталога

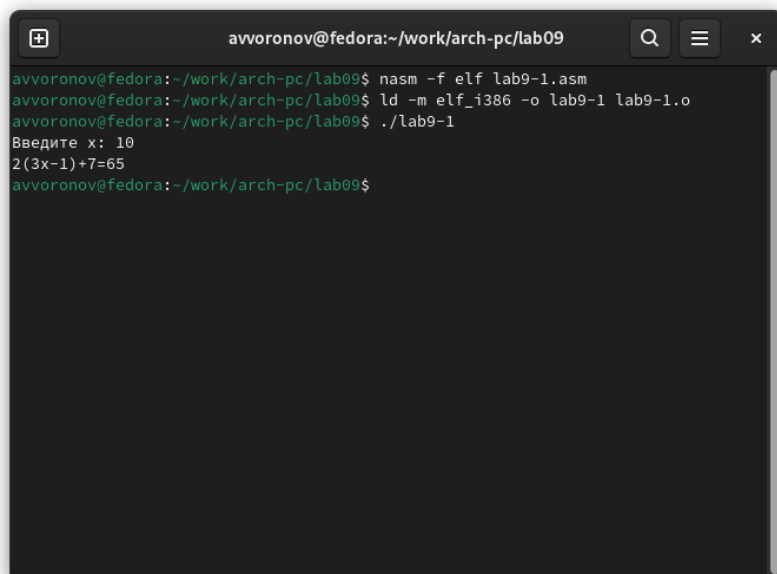
Копирую в файл код из листинга, компилирую и запускаю его, данная программа выполняет вычисление функции (рис. -fig. 4.2).



```
avvoronov@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
avvoronov@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
avvoronov@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 10
2x+7=27
avvoronov@fedora:~/work/arch-pc/lab09$
```

Рис. 4.2: Запуск программы из листинга

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения $f(g(x))$ (рис. -fig. 4.3).



```
avvoronov@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
avvoronov@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
avvoronov@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 10
2(3x-1)+7=65
avvoronov@fedora:~/work/arch-pc/lab09$
```

Рис. 4.3: Изменение программы первого листинга

Код программы:

```

#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

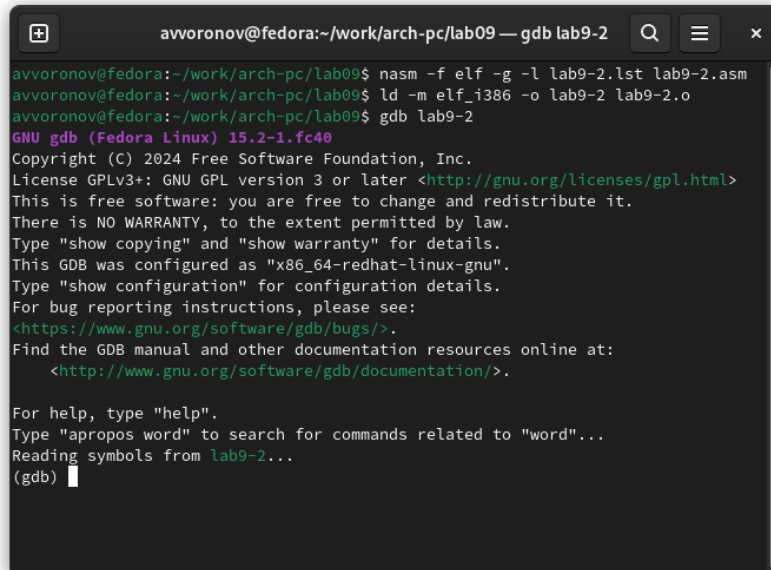
mov [res], eax
pop eax
ret

_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret

```

4.1.1 Отладка программ с помощью GDB

В созданный файл копирую программу второго листинга, транслирую с созданием файла листинга и отладки, компоную и запускаю в отладчике (рис. -fig. 4.4).

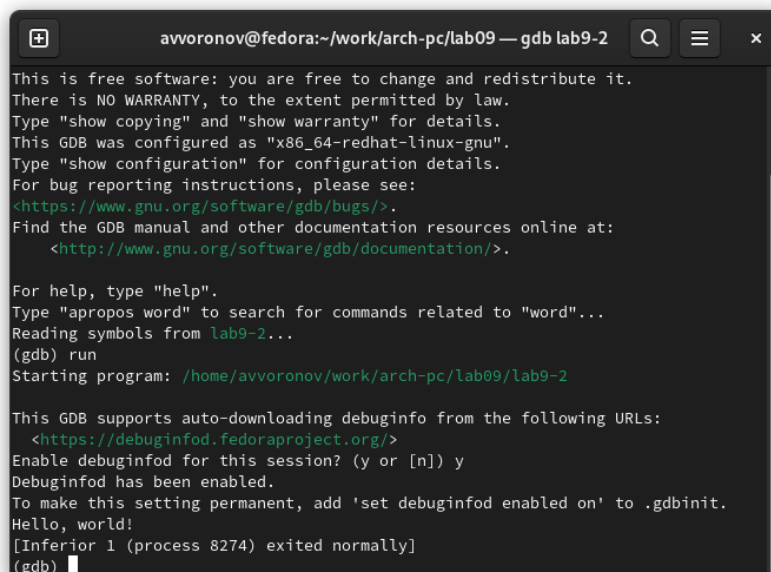


```
avvoronov@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
avvoronov@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
avvoronov@fedora:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.2-1.fc40
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: 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. 4.4: Запуск программы в отладчике

Запустив программу командой `run`, я убедился в том, что она работает исправно (рис. -fig. 4.5).



```
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/>.

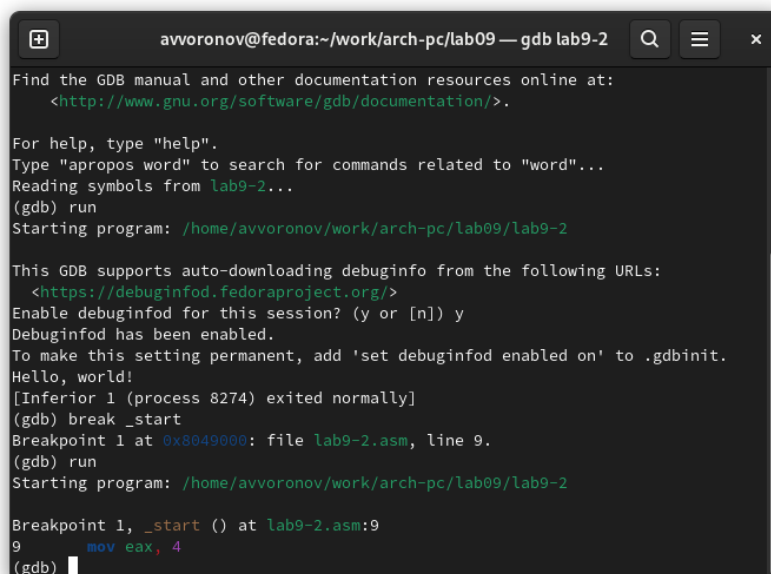
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) run
Starting program: /home/avvoronov/work/arch-pc/lab09/lab9-2

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.
Hello, world!
[Inferior 1 (process 8274) exited normally]
(gdb)
```

Рис. 4.5: Проверка программы отладчиком

Для более подробного анализа программы добавляю брейкпоинт на метку

_start и снова запускаю отладку (рис. -fig. 4.6).



```
avvoronov@fedora:~/work/arch-pc/lab09 — gdb lab9-2
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/avvoronov/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.
Hello, world!
[Inferior 1 (process 8274) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/avvoronov/work/arch-pc/lab09/lab9-2

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

Рис. 4.6: Запуск отладчика с брейкпоинтом

Далее смотрю дисассимилированный код программы, перевожу на команд с синтаксисом Intel *amd тончик* (рис. -fig. 4.7).

Различия между синтаксисом АТТ и Intel заключаются в порядке операндов (АТТ - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (АТТ - размер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом \$; Intel - Размер операндов неявно определяется контекстом, как ах, еах, непосредственные операнды пишутся напрямую), именах регистров (АТТ - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

```
avoronov@fedora:~/work/arch-pc/lab09 — gdb lab9-2
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) set disassembly-flavor intel
(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. 4.7: Дисассимилирование программы

Включаю режим псевдографики для более удобного анализа программы (рис. -fig. 4.8).

```
avoronov@fedora:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd070 0xffffd070
ebp      0x0      0x0

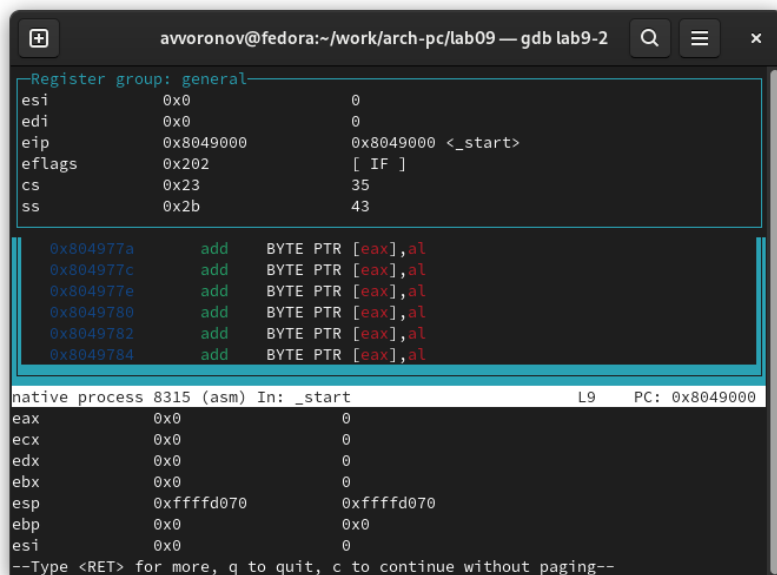
B+>0x08049000 <_start>  mov    eax,0x4
0x08049005 <_start+5>  mov    ebx,0x1
0x0804900a <_start+10> mov    ecx,0x804a000
0x0804900f <_start+15> mov    edx,0x8
0x08049014 <_start+20> int     0x80
0x08049016 <_start+22> mov    eax,0x4

native process 8315 (asm) In: _start      L9    PC: 0x08049000
(gdb) layout regs
(gdb) 
```

Рис. 4.8: Режим псевдографики

4.1.2 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. -fig. 4.9).



The screenshot shows the GDB interface with the following content:

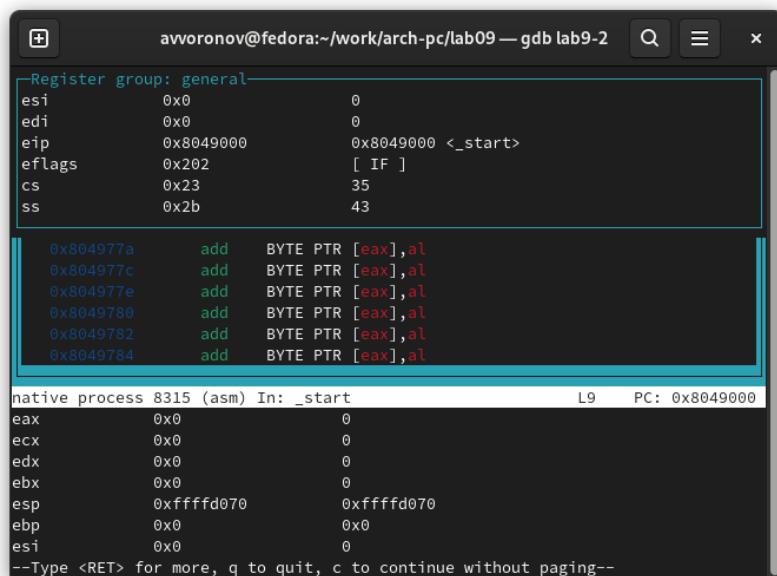
```
Register group: general
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x804977a add BYTE PTR [eax],al
0x804977c add BYTE PTR [eax],al
0x804977e add BYTE PTR [eax],al
0x8049780 add BYTE PTR [eax],al
0x8049782 add BYTE PTR [eax],al
0x8049784 add BYTE PTR [eax],al

native process 8315 (asm) In: _start L9 PC: 0x8049000
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd070 0xffffd070
ebp      0x0      0x0
esi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 4.9: Список брейкпоинтов

Устанавливаю еще одну точку останова по адресу инструкции (рис. -fig. 4.10).

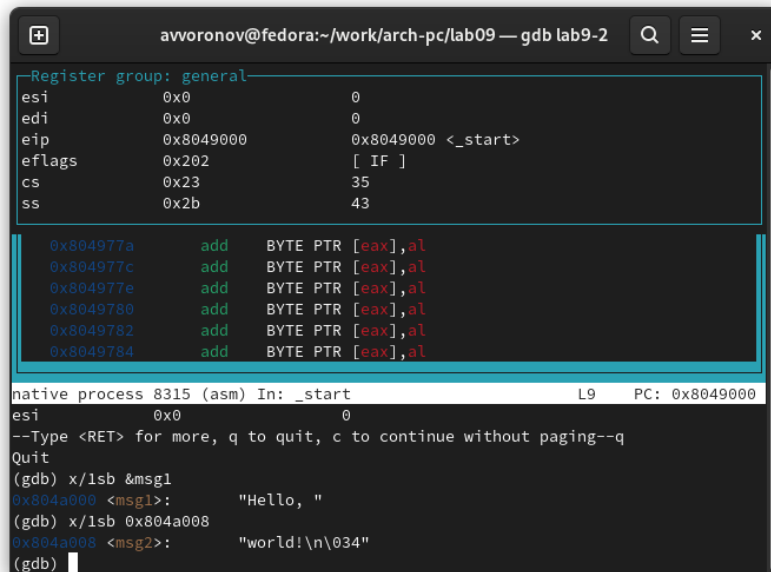


The screenshot shows the GDB interface with the same content as Figure 4.9, indicating that the breakpoints have been successfully added or are still present.

Рис. 4.10: Добавление второй точки останова

4.1.3 Работа с данными программы в GDB

Просматриваю содержимое регистров командой `info registers` (рис. -fig. 4.11).



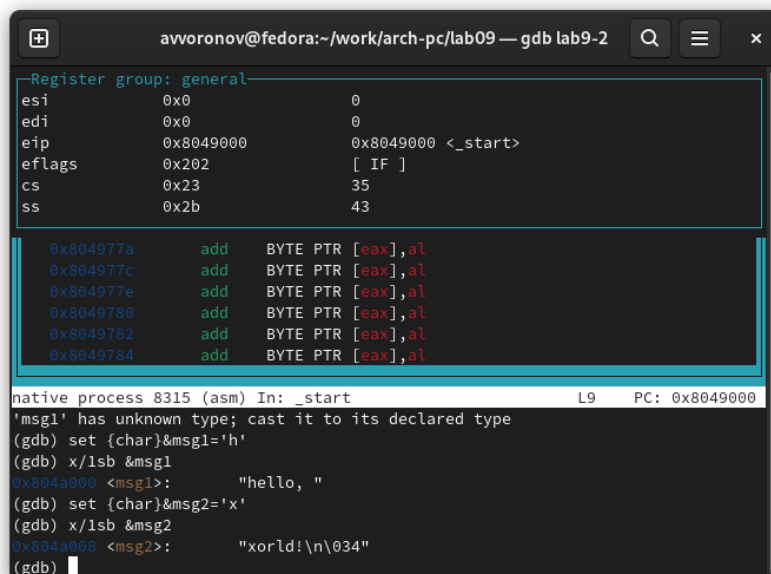
```
aworonov@fedora:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x804977a add BYTE PTR [eax],al
0x804977c add BYTE PTR [eax],al
0x804977e add BYTE PTR [eax],al
0x8049780 add BYTE PTR [eax],al
0x8049782 add BYTE PTR [eax],al
0x8049784 add BYTE PTR [eax],al

native process 8315 (asm) In: _start L9 PC: 0x8049000
esi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) x/1sb &msg1
0x804a008 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)
```

Рис. 4.11: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. -fig. 4.12).



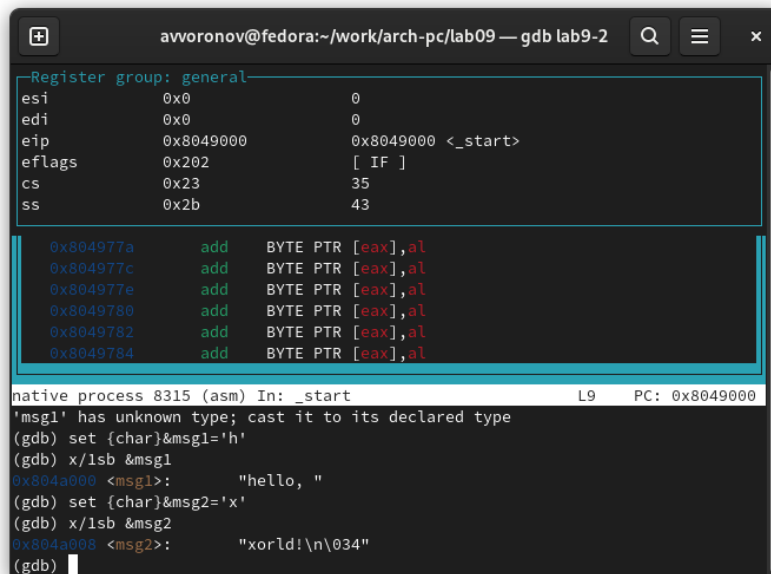
```
aworonov@fedora:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x804977a add BYTE PTR [eax],al
0x804977c add BYTE PTR [eax],al
0x804977e add BYTE PTR [eax],al
0x8049780 add BYTE PTR [eax],al
0x8049782 add BYTE PTR [eax],al
0x8049784 add BYTE PTR [eax],al

native process 8315 (asm) In: _start L9 PC: 0x8049000
'msg1' has unknown type; cast it to its declared type
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a008 <msg1>: "hello, "
(gdb) set {char}&msg2='x'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "xorld!\n\034"
(gdb)
```

Рис. 4.12: Просмотр содержимого переменных двумя способами

Меняю содержимое переменных по имени и по адресу (рис. -fig. 4.13).



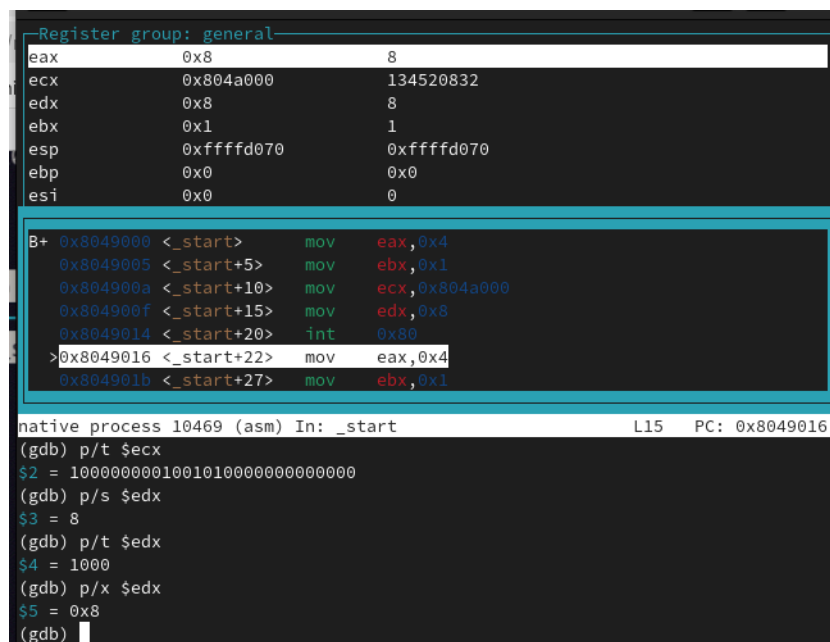
```
avvoronov@fedora:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x804977a add BYTE PTR [eax],al
0x804977c add BYTE PTR [eax],al
0x804977e add BYTE PTR [eax],al
0x8049780 add BYTE PTR [eax],al
0x8049782 add BYTE PTR [eax],al
0x8049784 add BYTE PTR [eax],al

native process 8315 (asm) In: _start L9 PC: 0x8049000
'msg1' has unknown type; cast it to its declared type
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}&msg2='x'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "xorld!\n\034"
(gdb)
```

Рис. 4.13: Изменение содержимого переменных двумя способами

Вывожу в различных форматах значение регистра edx (рис. -fig. 4.14).



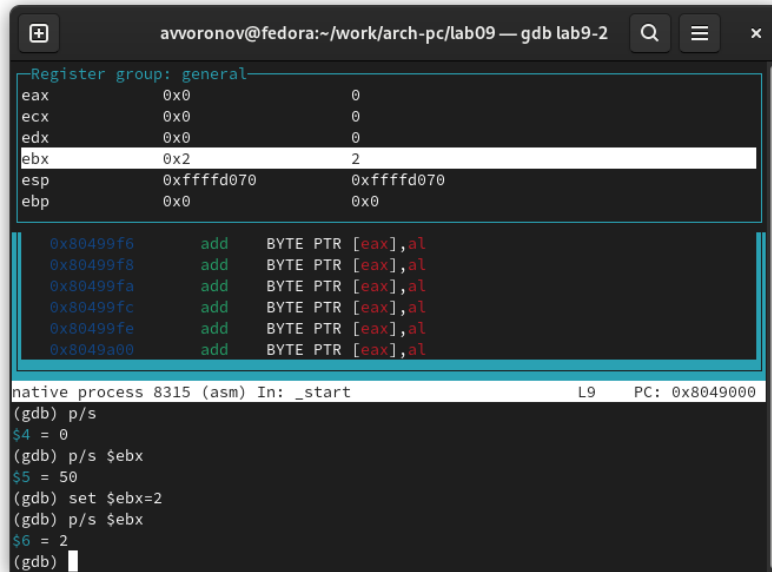
```
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd070 0xffffd070
ebp      0x0      0
esi      0x0      0

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
0x804901b <_start+27> mov ebx,0x1

native process 10469 (asm) In: _start L15 PC: 0x8049016
(gdb) p/t $ecx
$2 = 100000000100101000000000000000
(gdb) p/s $edx
$3 = 8
(gdb) p/t $edx
$4 = 1000
(gdb) p/x $edx
$5 = 0x8
(gdb)
```

Рис. 4.14: Просмотр значения регистра разными представлениями

С помощью команды set меняю содержимое регистра ebx (рис. -fig. 4.15).



The screenshot shows the GDB interface with the title bar 'avvoronov@fedora:~/work/arch-pc/lab09 — gdb lab9-2'. The 'Register group: general' section displays the following values:

| Register | Value (hex) | Value (decimal) |
|----------|-------------|-----------------|
| eax | 0x0 | 0 |
| ecx | 0x0 | 0 |
| edx | 0x0 | 0 |
| ebx | 0x2 | 2 |
| esp | 0xffffd070 | 0xffffd070 |
| ebp | 0x0 | 0x0 |

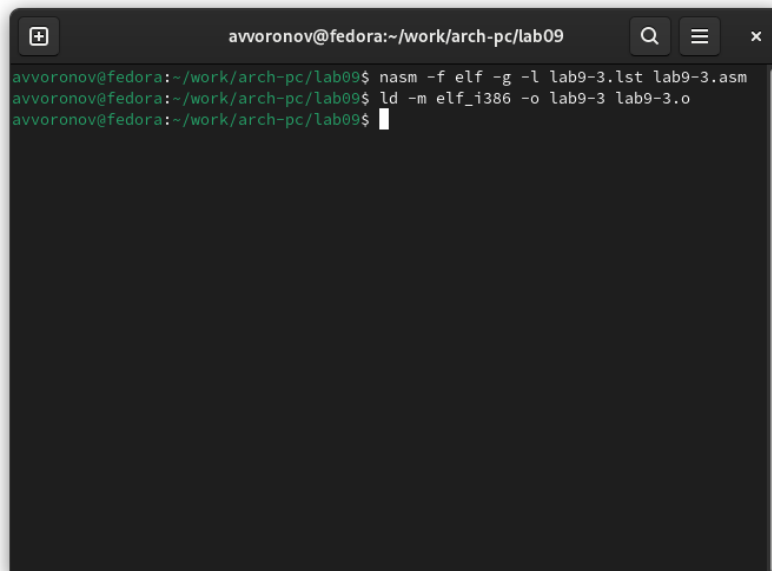
Below the registers, a list of assembly instructions is shown, all of which are 'add BYTE PTR [eax], al' at various memory addresses from 0x80499f6 to 0x8049a00. The status bar at the bottom indicates 'native process 8315 (asm) In: _start L9 PC: 0x8049000'. The command history shows the following sequence of commands:

```
(gdb) p/s
$4 = 0
(gdb) p/s $ebx
$5 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb)
```

Рис. 4.15: Примеры использования команды set

4.1.4 Обработка аргументов командной строки в GDB

Копирую программу из предыдущей лабораторной работы в текущий каталог и создаю исполняемый файл с файлом листинга и отладки (рис. -fig. 4.16).

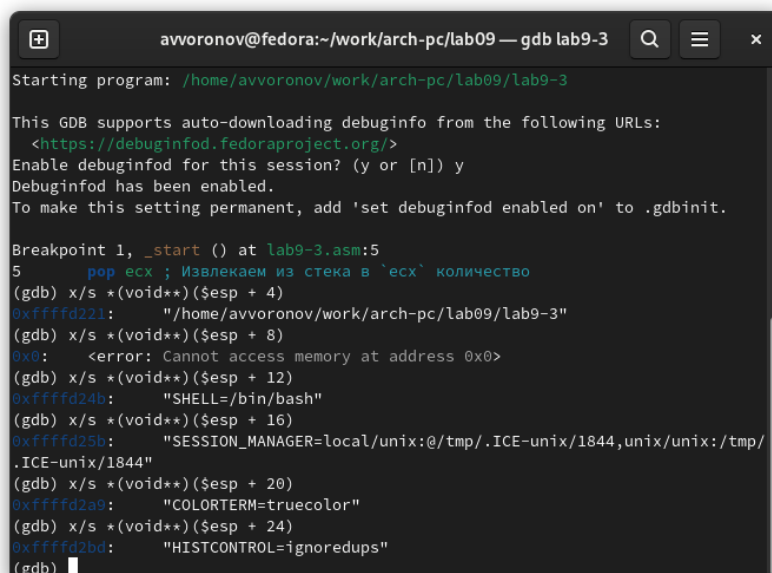


The screenshot shows a terminal window with the title bar 'avvoronov@fedora:~/work/arch-pc/lab09'. The following commands are executed:

```
avvoronov@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
avvoronov@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
avvoronov@fedora:~/work/arch-pc/lab09$
```

Рис. 4.16: Подготовка новой программы

Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпоинт и запускаю отладку. Проверяю работу стека, изменяя аргумент команды просмотра регистра `esp` на `+4`, число обусловлено разрядностью системы, а указатель `void` занимает как раз 4 байта, ошибка при аргументе `+24` означает, что аргументы на вход программы закончились. (рис. -fig. 4.17).



```
avvoronov@fedora:~/work/arch-pc/lab09 — gdb lab9-3
Starting program: /home/avvoronov/work/arch-pc/lab09/lab9-3
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 lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/s *(void**)($esp + 4)
0xffffd221: "/home/avvoronov/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)($esp + 8)
0x0: <error: Cannot access memory at address 0x0>
(gdb) x/s *(void**)($esp + 12)
0xffffd24b: "SHELL=/bin/bash"
(gdb) x/s *(void**)($esp + 16)
0xffffd25b: "SESSION_MANAGER=local/unix:@/tmp/.ICE-unix/1844,unix/unix:/tmp/
.ICE-unix/1844"
(gdb) x/s *(void**)($esp + 20)
0xffffd2a9: "COLORTERM=truecolor"
(gdb) x/s *(void**)($esp + 24)
0xffffd2bd: "HISTCONTROL=ignoredups"
(gdb)
```

Рис. 4.17: Проверка работы стека

4.2 Задание для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. -fig. 4.18).

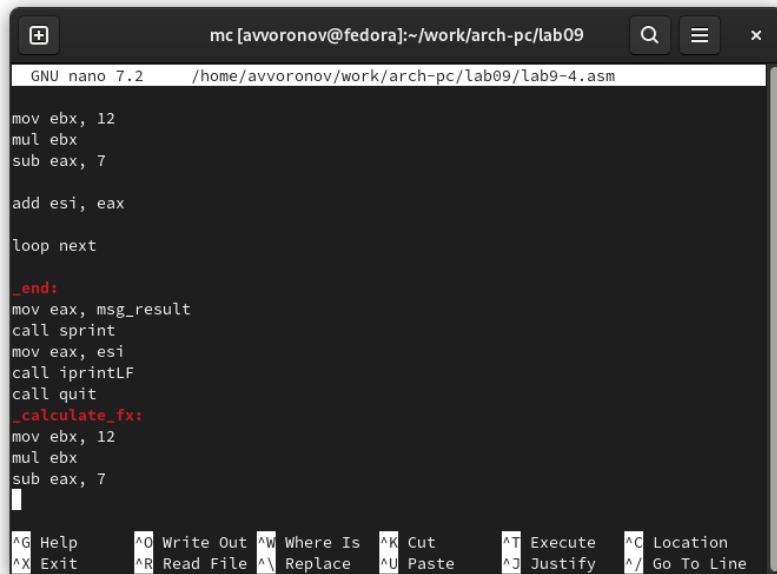


Рис. 4.18: Измененная программа предыдущей лабораторной работы

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 10x - 4", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintLF

pop ecx
pop edx
```

```

sub ecx, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end
pop eax
call atoi

call _calculate_fx

add esi, eax
loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit

_calculate_fx:
mov ebx, 10
mul ebx
sub eax, 4

```

2. Запускаю программу в режиме отладчика и пошагово через si просматриваю изменение значений регистров через i r. При выполнении инструкции mul ecx можно заметить, что результат умножения записывается в регистр eax, но также меняет и edx. Значение регистра ebx не обновляется напря-

мую, поэтому результат программа неверно подсчитывает функцию (рис. -fig. 4.19).

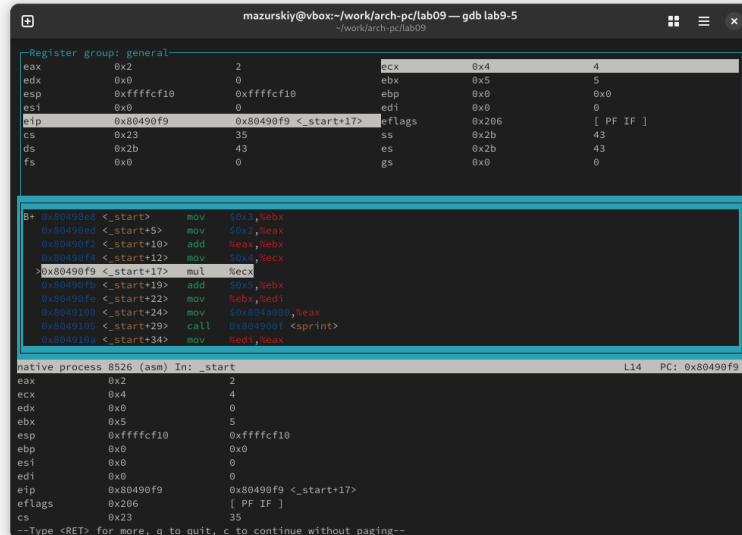


Рис. 4.19: Поиск ошибки в программе через пошаговую отладку

Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. -fig. 4.20).

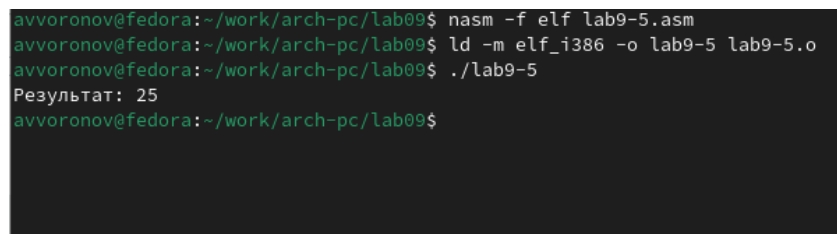


Рис. 4.20: Проверка корректировок в программе

Код измененной программы:

```
%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
```

5 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием подпрограмм, а так же познакомился с методами отладки при помощи GDB и его основными возможностями.

6 Список литературы

1. Курс на ТУИС
2. Лабораторная работа №9
3. Программирование на языке ассемблера NASM Столяров А. В.