

Отчёт по лабораторной работе №9

дисциплина: Архитектура компьютера

Веретенников Дмитрий Олегович

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	21
4.2	Задание для самостоятельной работы	23
5	Выводы	25
	Список литературы	26

Список иллюстраций

4.1	Создание исполняемого файла и запуск программы	8
4.2	Изменение программы	10
4.3	Запуск программы	11
4.4	Получение исполняемого файла и загрузка его в отладчик gdb . . .	11
4.5	Отладчик gdb	12
4.6	Установка брейкпоинта	13
4.7	Переключение	14
4.8	Режим псевдографики	15
4.9	Информация о точках останова	16
4.10	Содержание регистров	17
4.11	Значения переменных	18
4.12	Изменения первых символов переменных	19
4.13	Вывод ebx в разных форматах	20
4.14	Изменение значения регистра ebx	21
4.15	Создание исполняемого файла	22
4.16	Загрузка файла, установка точки останова и просмотр позиции стека	23
4.17	Измененная программа	24

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).

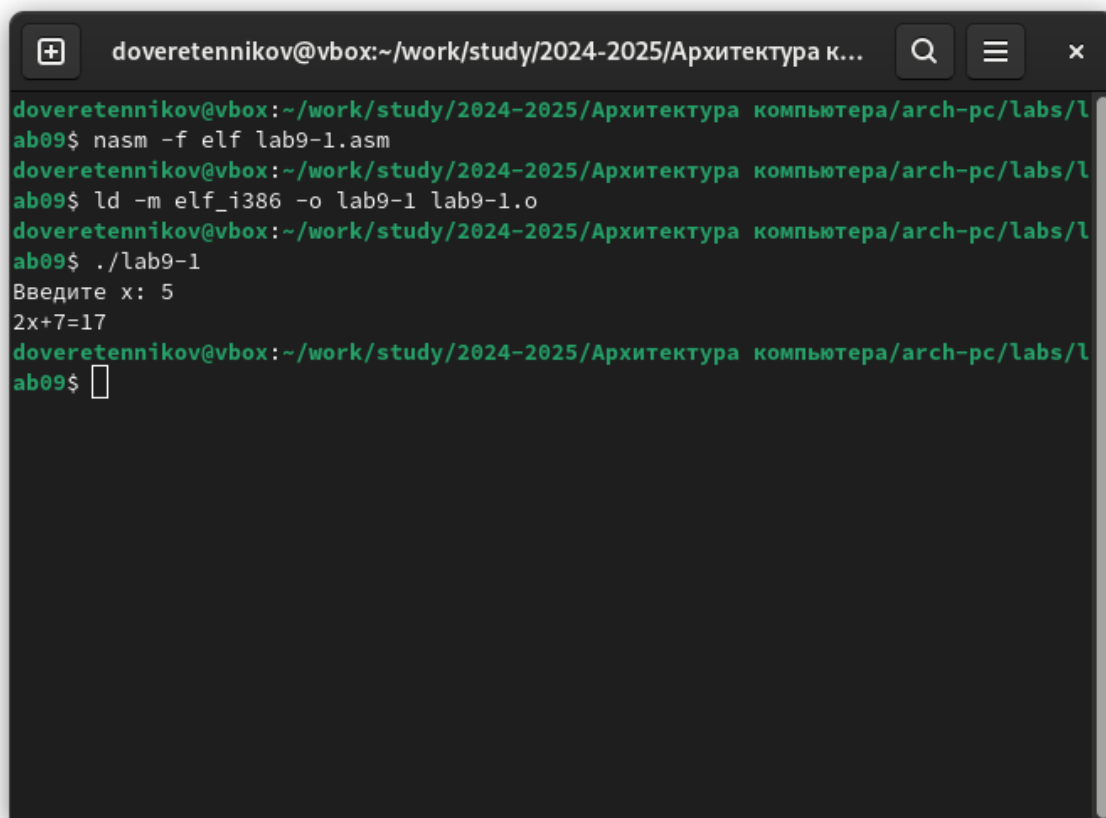
Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

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 Релазиация подпрограмм в NASM

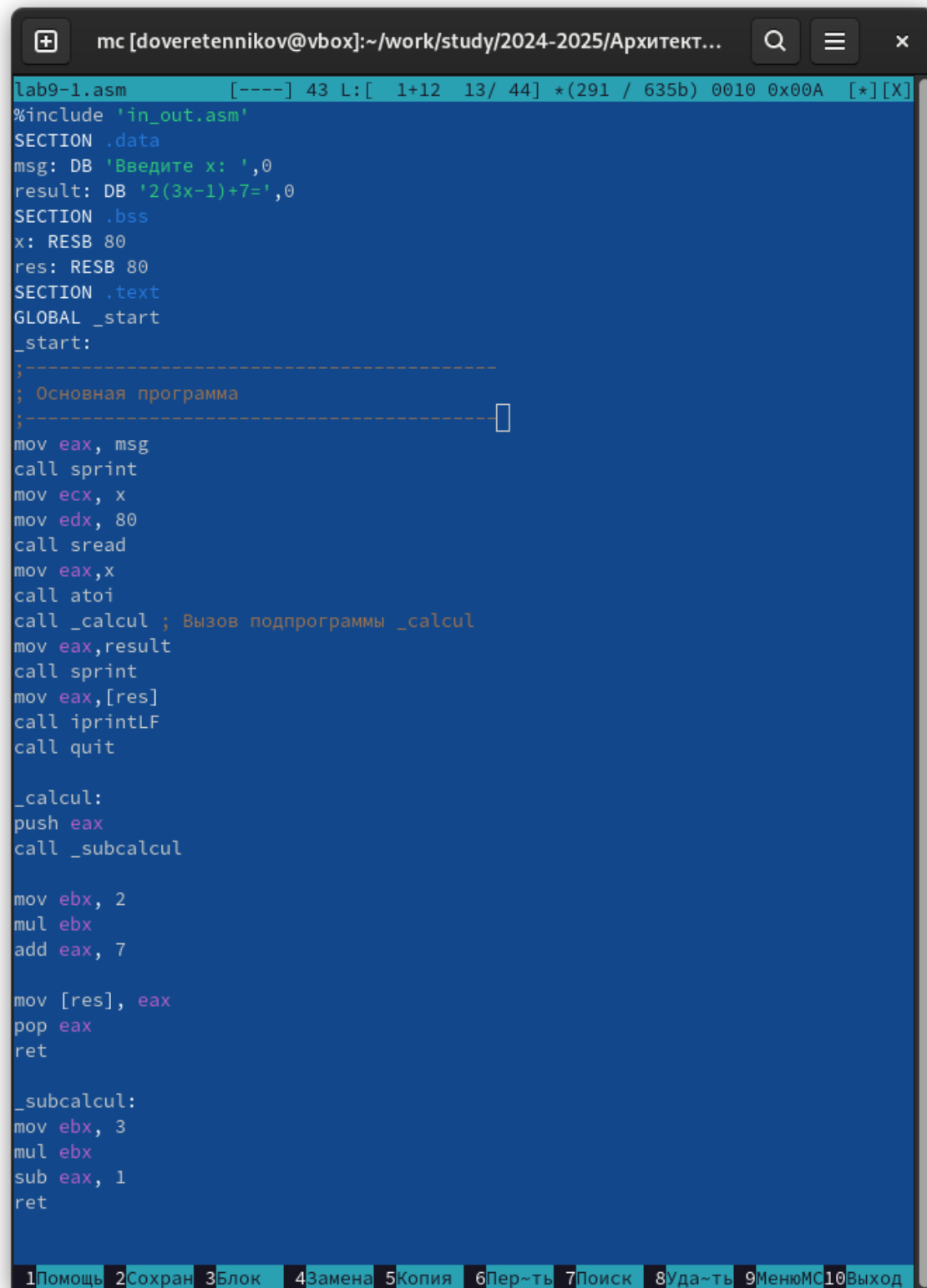
Ввожу в файл lab9-1.asm программу из листинга 9.1, создаю исполняемый файл и проверяю работу программы (рис. 4.1).



```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab09$ nasm -f elf lab9-1.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab09$ ./lab9-1
Введите x: 5
2x+7=17
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab09$
```

Рис. 4.1: Создание исполняемого файла и запуск программы

Изменяю текст программы, добавив подпрограмму для вычисления выражения $f(g(x))$ (рис. 4.2).



```
lab9-1.asm [----] 43 L: [ 1+12 13/ 44] *(291 / 635b) 0010 0x00A [*][X]
#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:
;-----
; 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

_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
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход

Рис. 4.2: Изменение программы

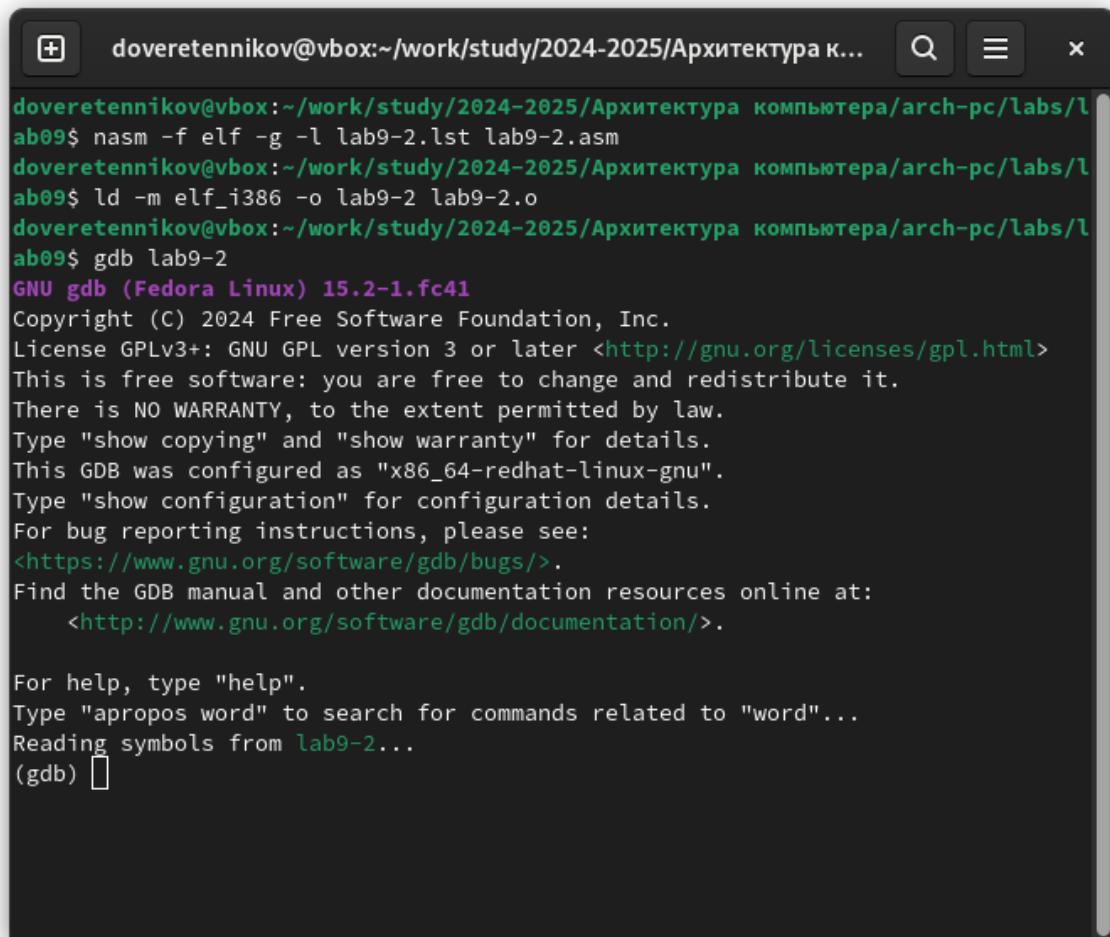
Запускаю измененную программу и проверяю правильность ее работы (рис. 4.3).

Запуск программы

Рис. 4.3: Запуск программы

4.1.1 Отладка программ с помощью GDB

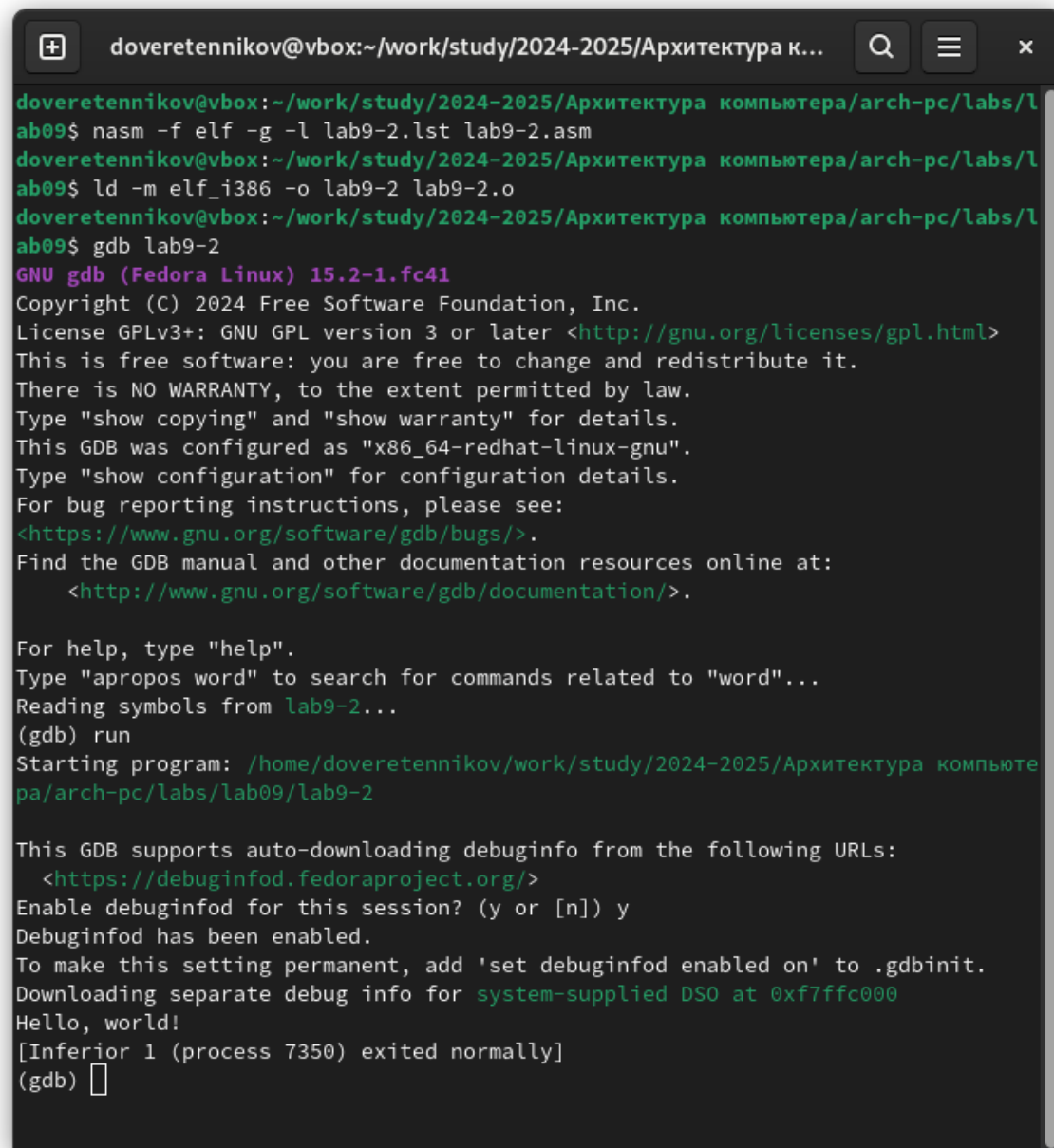
Создаю файл lab9-2.asm ввожу в него текст из листинга 9.2, получаю исполняемый файл и загружаю файл в отладчик gdb (рис. 4.4).



```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.2-1.fc41
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: Получение исполняемого файл и загрузка его в отладчик gdb



```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.2-1.fc41
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/doveretennikov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/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 7350) exited normally]
(gdb) □
```

Рис. 4.5: Отладчик gdb

Устанавливаю брейкпоинт на метку `_start` (рис. 4.5).

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...
ab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.2-1.fc41
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) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/doveretennikov/work/study/2024-2025/Архитектура компьюте
pa/arch-pc/labs/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.

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

Рис. 4.6: Установка брейкпоинта

Просматриваю дисассимилированный код программы (рис. 4.6).

!Дисассимилированный код программы[/home/doveretennikov/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09/Report/image/Вставленное изображение (7).png]{#fig:007}

Переключаюсь на отображение команд с Intel'овским синтаксисом (рис. 4.7).

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...
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: Переключение

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).

Включаю режим псевдографики (рис. 4.8).

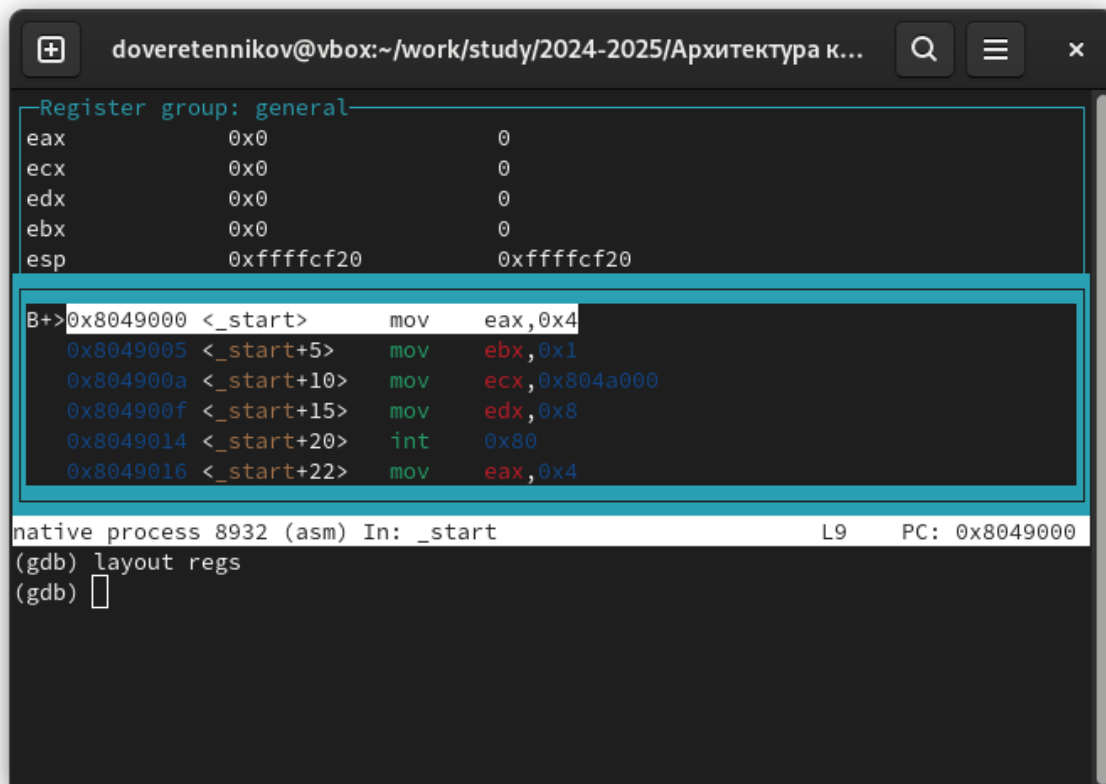


Рис. 4.8: Режим псевдографики

4.1.2 Добавление точек останова

Устанавливаю еще одну точку останова по адресу инструкции и просматриваю информацию о всех установленных точках останова (рис. 4.9).

The screenshot shows a GDB terminal window with the title bar "doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...". The window is divided into three main sections. The top section, titled "Register group: general", displays the values of registers: `eax` (0x0), `ecx` (0x0), `edx` (0x0), `ebx` (0x0), and `esp` (0xffffcf20). The middle section, titled "B+>0x8049000 <_start>", lists assembly instructions with their addresses and the instructions themselves: `mov eax,0x4` at 0x8049000, `mov ebx,0x1` at 0x8049005, `mov ecx,0x804a000` at 0x804900a, `mov edx,0x8` at 0x804900f, `int 0x80` at 0x8049014, and `mov eax,0x4` at 0x8049016. The bottom section shows the GDB prompt and the command `break *0x8049031`, followed by the output: "Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20." and the command `i b`, which displays a table of breakpoints.

Num	Type	Disp	Enb	Address	What
1	breakpoint	keep y		0x08049000	lab9-2.asm:9
					breakpoint already hit 1 time
2	breakpoint	keep y		0x08049031	lab9-2.asm:20

Рис. 4.9: Информация о точках останова

4.1.3 Работа с данными программы в GDB

Просматриваю содержимое регистров (рис. 4.10).


```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcf20 0xffffcf20

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 8932 (asm) In: _start L9 PC: 0x8049000
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcf20 0xffffcf20
ebp      0x0      0x0
esi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 4.10: Содержание регистров

Просматриваю значение переменной msg1 по имени и переменной msg2 по адресу (рис. 4.11).

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcf20 0xffffcf20

0x804a0f6  add  al,BYTE PTR [eax]
0x804a0f8  and  DWORD PTR [eax],eax
0x804a0fa  add  BYTE PTR [eax],al
0x804a0fc  add  DWORD PTR [ecx],eax
0x804a0fe  sti
0x804a0ff  push cs

native process 8932 (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
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) 
```

Рис. 4.11: Значения переменных

Изменяю первый символ переменной msg1 и переменной msg2 (рис. 4.12).

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcf20 0xffffcf20

0x804a0f6  add  al,BYTE PTR [eax]
0x804a0f8  and  DWORD PTR [eax],eax
0x804a0fa  add  BYTE PTR [eax],al
0x804a0fc  add  DWORD PTR [ecx],eax
0x804a0fe  sti
0x804a0ff  push cs

native process 8932 (asm) In: _start L9 PC: 0x8049000
warning: Expression is not an assignment (and might have no effect)
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}&msg2='z'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "zorld!\n\034"
(gdb) 
```

Рис. 4.12: Изменения первых символов переменных

Вывожу в различных форматах значение регистра ebx (рис. 4.13).

The screenshot shows a debugger window with the title bar 'doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...'. The main pane displays assembly code for the 'general' register group. The registers are listed as follows:

Register	Value	Comment
eax	0x0	0
ecx	0x0	0
edx	0x0	0
ebx	0x0	0
esp	0xffffcf20	0xffffcf20

Below the register list, a block of assembly code is highlighted with a red border:

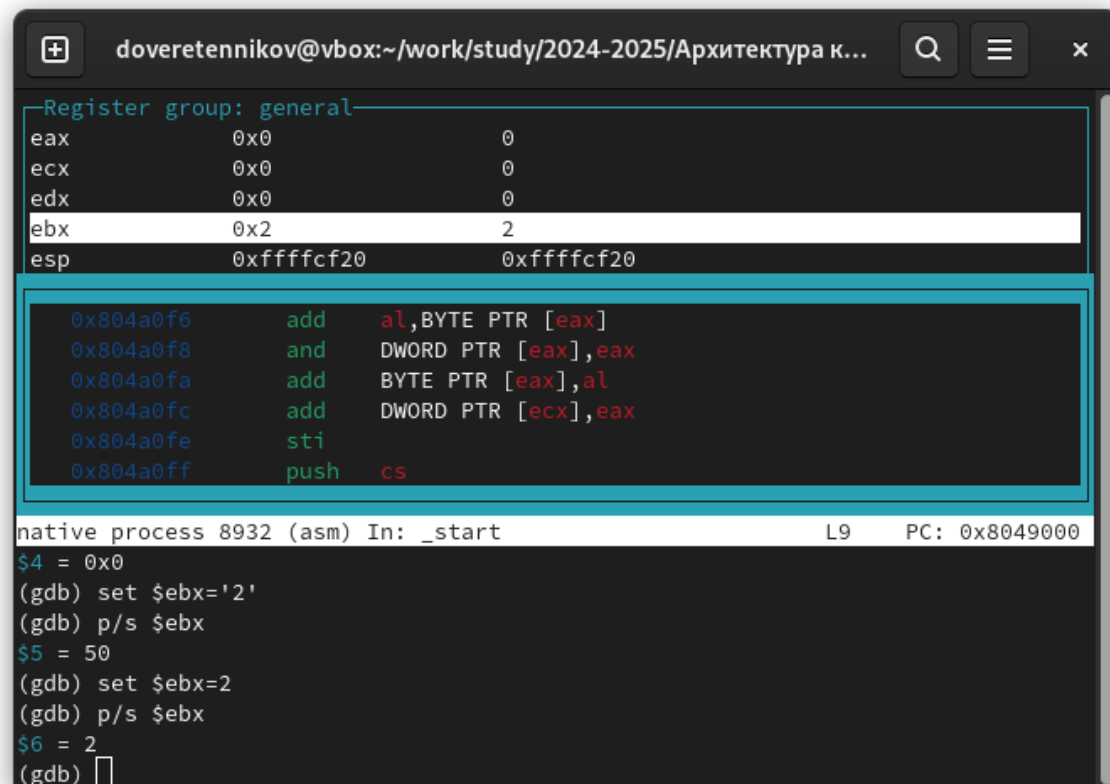
```
0x804a0f6  add  al,BYTE PTR [eax]
0x804a0f8  and  DWORD PTR [eax],eax
0x804a0fa  add  BYTE PTR [eax],al
0x804a0fc  add  DWORD PTR [ecx],eax
0x804a0fe  sti
0x804a0ff  push cs
```

The bottom pane shows the native process 8932 (asm) In: _start, L9, PC: 0x8049000. The GDB console shows the following commands and output:

```
$1 = 0
(gdb) p/t $edx
$2 = 0
(gdb) p/s $edx
$3 = 0
(gdb) p/x $edx
$4 = 0x0
(gdb) 
```

Рис. 4.13: Вывод ebx в разных форматах

С помощью команды set изменяю значение регистра ebx (рис. 4.14).



```
dovertennikov@vbox:~/work/study/2024-2025/Архитектура к...
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x2      2
esp      0xffffcf20 0xffffcf20

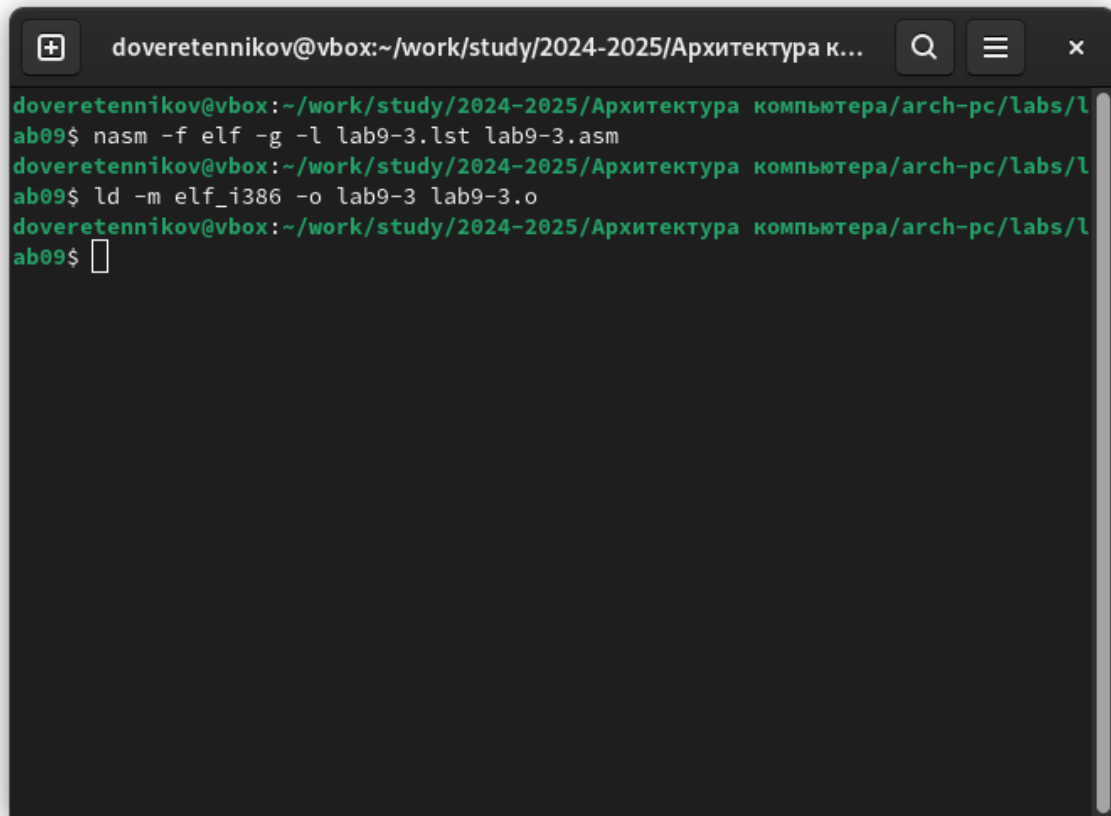
0x804a0f6  add  al, BYTE PTR [eax]
0x804a0f8  and  DWORD PTR [eax], eax
0x804a0fa  add  BYTE PTR [eax], al
0x804a0fc  add  DWORD PTR [ecx], eax
0x804a0fe  sti
0x804a0ff  push cs

native process 8932 (asm) In: _start L9 PC: 0x8049000
$4 = 0x0
(gdb) set $ebx='2'
(gdb) p/s $ebx
$5 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb) 
```

Рис. 4.14: Изменение значения регистра ebx

4.1.4 Обработка аргументов командной строки в GDB

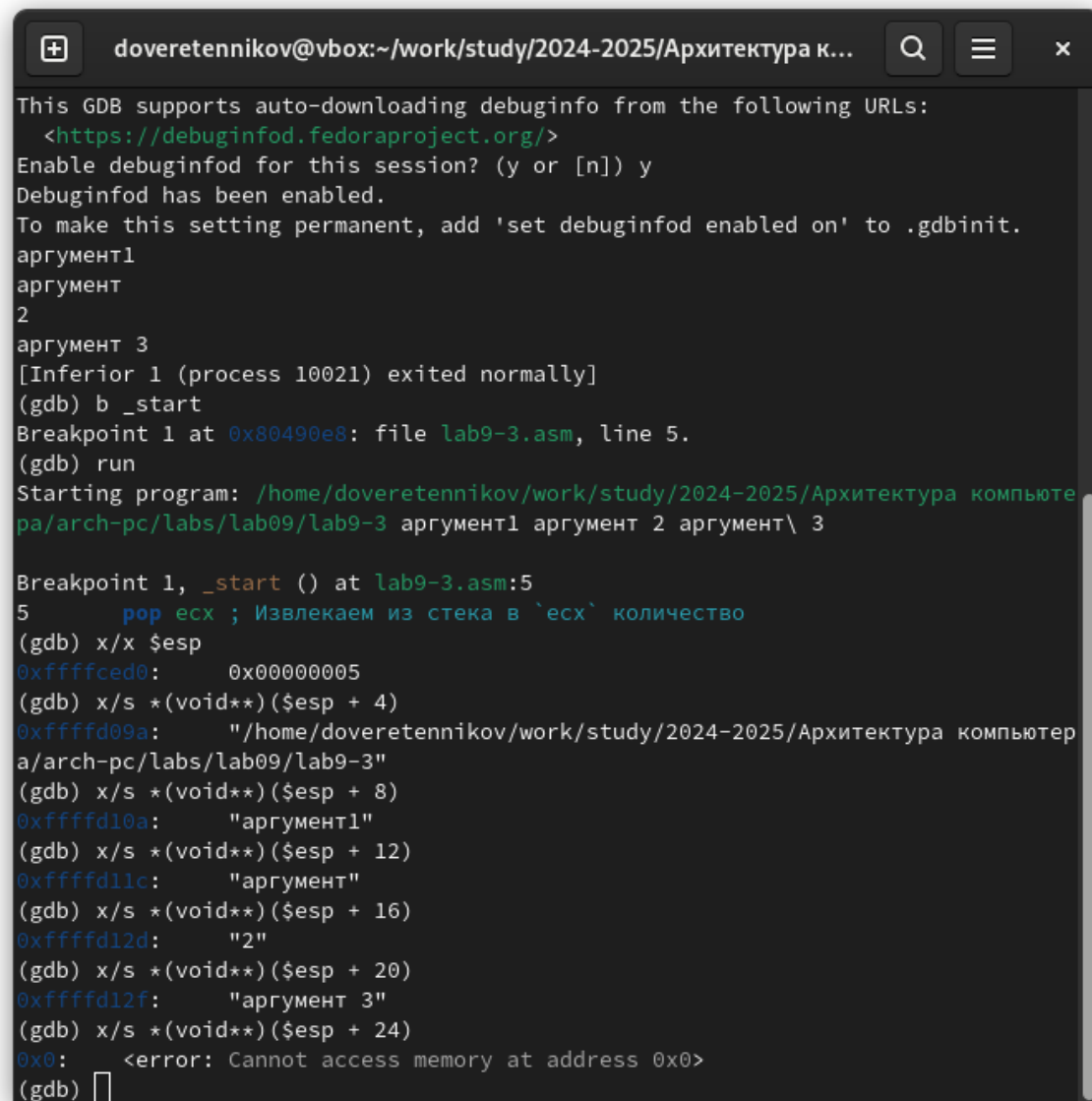
Копирую файл lab8-2.asm в файл с именем lab9-3.asm и создаю исполняемый файл (рис. 4.15).

A terminal window with a dark background and green text. The window title is "doveretennikov@vbox:~/work/study/2024-2025/Архитектура к...". The terminal shows the following commands and output:

```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$
```

Рис. 4.15: Создание исполняемого файла

Загружая исполняемый файл в отладчик указываю аргументы, далее устанавливаю точку останова перед первой инструкцией и затем просматриваю позиции стека (рис. 4.16).



```
doveretennikov@vbox:~/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.
аргумент1
аргумент
2
аргумент 3
[Inferior 1 (process 10021) exited normally]
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /home/doveretennikov/work/study/2024-2025/Архитектура компьюте
pa/arch-pc/labs/lab09/lab9-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffced0: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffd09a: "/home/doveretennikov/work/study/2024-2025/Архитектура компьютер
a/arch-pc/labs/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd10a: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd11c: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd12d: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd12f: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb) □
```

Рис. 4.16: Загрузка файла, установка точки останова и просмотр позиции стека

4.2 Задание для самостоятельной работы

Изменяю программу из 8 лабораторной работы чтобы она вычисляла значения как подпрограмма (рис. 4.17).

```
lab9-4.asm [----] 9 L: [ 1+26 27/ 31] *(390 / 435b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .data
msg_func db "Функция: f(x) = 7 + 2x", 0
msg_result db "Результат: ", 0
SECTION .text
GLOBAL _start
_start:
mov eax, msg_func
call sprintf
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 sprintf
mov eax, esi
call iprintfLF
call quit
_calculate_fx:
mov ebx, 2
mul ebx
add eax, 7
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 4.17: Измененная программа

5 Выводы

После 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.

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/Learning-bash-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,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. 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.
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
 17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
 18. — 1120 с. — (Классика Computer Science).