

# **Отчет по лабораторной работе №9**

**Понятие подпрограммы. Отладчик GDB**

Карачевцева Елизавета

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Самостоятельная работа</b>	<b>19</b>
	<b>Вывод</b>	<b>24</b>

# Список иллюстраций

2.1	Создание каталога и файла . . . . .	6
2.2	Текст программы . . . . .	7
2.3	Работа программы . . . . .	8
2.4	Измененный текст программы . . . . .	8
2.5	Проверка работы программы . . . . .	8
2.6	Текст второй программы . . . . .	9
2.7	Отладка второго файла . . . . .	10
2.8	Брежпоинт на метку _start . . . . .	10
2.9	Дисассимплированный код . . . . .	11
2.10	Intel’овское отображение . . . . .	12
2.11	Псевдографика . . . . .	13
2.12	Наличие меток . . . . .	13
2.13	Просмотр регистров . . . . .	14
2.14	Измененные регистры . . . . .	14
2.15	Просмотр значения переменной . . . . .	15
2.16	Значение переменной msg2 . . . . .	15
2.17	Изменение значения переменной . . . . .	15
2.18	Изменение msg2 . . . . .	15
2.19	Значение регистров ехх и еах . . . . .	16
2.20	Значение регистров ебх . . . . .	17
2.21	Завершение работы с файлов . . . . .	17
2.22	Запуск файла в отладчике . . . . .	17
2.23	Запуск файла lab9-3 через метку . . . . .	18
2.24	Адрес вершины стека . . . . .	18
2.25	Все позиции стека . . . . .	18
3.1	Текст программы . . . . .	20
3.2	Запуск программы . . . . .	21
3.3	Текст программы . . . . .	22
3.4	Запуск программы . . . . .	23
3.5	Запуск программы в отладчике . . . . .	23

## Список таблиц

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

1) Я создала каталог lab9 и создал файл lab9-1.asm

```
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ mkdir lab9
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd lab9
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ touch lab9-1.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ls
```

Рис. 2.1: Создание каталога и файла

2) Я ввела текст листинга в файл и запустила программу.

```

GNU nano 7.2 /home/evkarachevtseva/work/study
#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. 2.2: Текст программы

```

evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ nasm -f elf lab9-1.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ld -m elf_i386 -o lab9-1 lab9-1.o
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ./lab9-1
Введите x: 5
2x+7=17

```

Рис. 2.3: Работа программы

3) Я изменила текст программы, чтобы она решала выражение  $f(g(x))$ .

```

GNU nano 7.2 /home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9/lab9-1.asm
#include "in_out.asm"
SECTION .data
msg: DB 'Введите x: ',0
prim1: DB 'f(x)=2x+7',0
prim2: DB 'g(x)=3x-1',0
result: DB 'f(g(x))=',0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, prim1
call sprintf
mov eax, prim2
call sprintf
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul
mov eax, result
call sprintf
mov eax, [res]
call iprintf
call quit
calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret
subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret

```

Рис. 2.4: Измененный текст программы

```

evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ nasm -f elf lab9-1.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ld -m elf_i386 -o lab9-1 lab9-1.o
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ./lab9-1
f(x)=2x+7
g(x)=3x-1
Введите x: 1
f(g(x))=11

```

Рис. 2.5: Проверка работы программы

4) Я создала файл lab9-2.asm и вписала туда программу.



```

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. 2.6: Текст второй программы

5) Я загрузила и запустила файл второй программы в отладчик gdb.

```
evkarachevtseva@fedora: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
evkarachevtseva@fedora: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ld -m elf_i386 -o lab9-2 lab9-2.o
evkarachevtseva@fedora: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ gdb lab9-2
GNU gdb (Fedora Linux) 14.2-1.fc40
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) r
quit
Starting program: /home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9/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 0x7ffff000
Hello, world!
[Inferior 1 (process 14914) exited normally]
(gdb) █
```

Рис. 2.7: Отладка второго файла

6) Я поставила брекпоинт на метку \_start и запустила программу.

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 10.
(gdb) r
Starting program: /home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9/lab9-2
Breakpoint 1, _start () at lab9-2.asm:10
10      mov eax, 4
(gdb) █
```

Рис. 2.8: Брекпоинт на метку \_start

7) Я просмотрела дисассимплированный код программы начиная с метки.

```

Undefined command: "disassaemble". Try "help".
(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. 2.9: Дисассимплированный код

- 8) С помощью команды я переключилась на intel'овское отображение синтаксиса. Отличие заключается в командах, в диссимилированном отображении в командах используют % и \$, а в Intel отображение эти символы не используются. На такое отображение удобнее смотреть.

```

(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.

```

Рис. 2.10: Intel'овское отображение

9) Для удобства я включила режим псевдографики.

```

[ 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
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edx,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
0x8049031 <_start+49> mov ebx,0x0
0x8049036 <_start+54> int 0x80
0x8049038 add BYTE PTR [eax],al
0x804903a add BYTE PTR [eax],al
0x804903c add BYTE PTR [eax],al

native process 15048 In: _start
(gdb) layout regs
(gdb)

```

Рис. 2.11: Псевдографика

- 10) Я посмотрела наличие меток и добавила еще одну метку на предпоследнюю инструкцию.

```

0x804903a add BYTE PTR [eax],al
0x804903c add BYTE PTR [eax],al

native process 15048 In: _start
(gdb) layout regs
(gdb) info breakpoints
Num      Type           Disp Enb Address      What
1        breakpoint      keep y  0x08049000  lab9-2.asm:10
          breakpoint already hit 1 time
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 21.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint      keep y  0x08049000  lab9-2.asm:10
          breakpoint already hit 1 time
2        breakpoint      keep y  0x08049031  lab9-2.asm:21
(gdb)

```

Рис. 2.12: Наличие меток

- 11) С помощью команды si я посмотрела регистры и изменила их.

```

--Register group: General
eax 0x4 4 ebx 0x0 0
edx 0x0 0 ebx 0x0 0
esp 0xffffcfc0 0xffffcfc0 ebp 0x0 0x0
esi 0x0 0 edi 0x0 0
eip 0x8049005 0x8049005 <_start+5> eflags 0x202 [ IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

0x8049005 <_start+5> mov eax, 4
0x8049006 <_start+10> mov ebx, 0x1000
0x8049007 <_start+15> mov edx, 0x5
0x8049014 <_start+20> int 0x0
0x8049016 <_start+22> mov eax, 0x4
0x804901b <_start+27> mov ebx, 0x1
0x8049020 <_start+32> mov ecx, 0x8049005
0x8049025 <_start+37> mov edx, 0x7
0x804902a <_start+42> int 0x0
0x804902c <_start+44> mov eax, 0x1
0x8049031 <_start+49> mov ebx, 0x5
0x8049036 <_start+54> int 0x0
0x8049038 add BYTE PTR [eax], al
0x804903a add BYTE PTR [eax], al
0x804903c add BYTE PTR [eax], al

native process 16376 In: _start L11 PC: 0x8049005
esi 0x0 0
edi 0x0 0
ebp 0x804900a 0x804900a <_start+10>
eflags 0x202 [ IF ]
cs 0x23 35
ss 0x2b 43
ds 0x2b 43
es 0x2b 43
fs 0x0 0
gs 0x0 0

(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/evkarachevtsevs/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9/lab9-2

Breakpoint 1, _start () at lab9-2.asm:10
(gdb) si
(gdb)

```

Рис. 2.13: Просмотр регистров

```

eax 0x4 4
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xffffcfc0 0xffffcfc0
ebp 0x0 0x0
esi 0x0 0
edi 0x0 0
eip 0x8049005 0x8049005 <_start+5>
eflags 0x202 [ IF ]
cs 0x23 35
ss 0x2b 43
ds 0x2b 43
es 0x2b 43
fs 0x0 0
gs 0x0 0

```

Рис. 2.14: Измененные регистры

12) С помощью команды я посмотрела значение переменной msg1.

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

Рис. 2.15: Просмотр значения переменной

13) Следом я посмотрела значение второй переменной msg2.

```
(gdb) x/lsb 0x804a008
0x804a008 <msg2>: "world!\n\034"
```

Рис. 2.16: Значение переменной msg2

14) С помощью команды set я изменила значение переменной msg1.

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

Рис. 2.17: Изменение значения переменной

15) Я изменила переменную msg2.

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

Рис. 2.18: Изменение msg2

16) Я вывела значение регистров ехх и еах.

```
(gdb) p/f $msg1
$1 = void
(gdb) p/s $eax
$2 = 4
(gdb) p/t $eax
$3 = 100
(gdb) p/c $ecx
$4 = 0 '\000'
(gdb) p/x $ecx
$5 = 0x0
```

Рис. 2.19: Значение регистров ecx и eax

- 17) Я изменила значение регистра ebx. Команда выводит два разных значения так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум, поэтому и значения разные.



```

(gdb) set $ebx='2'
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2

```

Рис. 2.20: Значение регистров ebx

18) Я завершила работу с файлом и вышла.

```

[Inferior 1 (process 16904) exited normally]

```

Рис. 2.21: Завершение работы с файлов

19) Я скопировала файл lab9-2.asm и переименовала его. Запустила файл в отладчике и указала аргументы.

```

evkarachevtseva@fedora: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
evkarachevtseva@fedora: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab$ ld -m elf_i386 -o lab9-3 lab9-3.o
evkarachevtseva@fedora: /work/study/2024-2025/Архитектура компьютера/arch-pc/lab$ gdb -args lab9-3 аргумент1 аргумент2 'аргумент 3'
GNU gdb (Fedora Linux) 14.2-1.fc40
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-3...
(gdb)

```

Рис. 2.22: Запуск файла в отладчике

20) Поставила метку на \_start и запустила файл.

```

(gdb) b _start
Breakpoint 1 at 0x00400000: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9/lab9-3 аргумент1 аргумент2 а
ргумент\ 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)

```

Рис. 2.23: Запуск файла lab9-3 через метку

21) Я проверила адрес вершины стека и убедилась что там хранится 5 элементов.

```

(gdb) x/x $esp
0xfffffcf70:      0x00000004
(gdb)

```

Рис. 2.24: Адрес вершины стека

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

```

(gdb) x/x $esp
0xfffffcf70:      0x00000004
(gdb) x/s +(void**)( $esp + 4)
0xfffffd13a:      "/home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9/lab9-3"
(gdb) x/s +(void**)( $esp + 8)
0xfffffd14c:      "аргумент1"
(gdb) x/s +(void**)( $esp + 12)
0xfffffd1b7:      "аргумент2"
(gdb) x/s +(void**)( $esp + 16)
0xfffffd1c9:      "аргумент 3"
(gdb) x/s +(void**)( $esp + 20)
> <error: Cannot access memory at address 0x0>

```

Рис. 2.25: Все позиции стека

### **3 Самостоятельная работа**

- 1) Я преобразовала программу из лабораторной работы №8 и реализовала вычисления как подпрограмму.

```

%include 'in_out.asm'
SECTION .data
prim DB 'f(x)=(x+1)*7',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

pop ecx

pop edx

sub ecx,1

mov esi,0

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

mov ebx,7
pop eax
call atoi
add eax,1
mul ebx
|

add esi,eax

loop next

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

```

Рис. 3.1: Текст программы

```
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/lab8$ nasm -f elf lab8-4.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/lab8$ ld -m elf_i386 -o lab8-4 lab8-4.o
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура_компьютера/arch-pc/lab8$ ./lab8-4 1 2 3 4
f(x)=(x+1)*7
Результат: 98
```

Рис. 3.2: Запуск программы

- 2) Я переписала программу и попробовала запустить ее чтобы увидеть ошибку.  
Ошибки не было, программа выдает верный ответ.

```

%include 'in_out.asm'
SECTION .data
prim DB 'f(x)=(x+1)*7',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

pop ecx

pop edx

sub ecx,1

mov esi,0

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

pop eax
call atoi
call fir
add esi,eax

loop next

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

fir:
mov ebx,7
add eax,1
mul ebx
ret

```

Рис. 3.3: Текст программы

```

evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ./lab9-4 1 2 3 4
f(x)=(x+1)*7
Результат: 98

```

Рис. 3.4: Запуск программы

На всякий случай, я запустила программу в отладчике.

```

evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ld -m elf_i386 -o lab9-4 lab9-4.o
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ gdb lab9-4
GNU gdb (Fedora Linux) 14.2-1.fc40
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-4...
(gdb) b _start
Breakpoint 1 at 0x00490e8: file lab9-4.asm, line 9.
(gdb) r
Starting program: /home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9/lab9-4

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 lab9-4.asm:9
9      pop ecx
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x080490e8 <+0>:  pop    ecx
    0x080490e9 <+1>:  pop    edx
    0x080490ea <+2>:  sub    ecx,0x1
    0x080490ed <+5>:  mov    esi,0x0
    0x080490f2 <+10>: mov    eax,0x004a0000
    0x080490f7 <+15>: call   0x0804902d <sprintf@plt>
End of assembler dump.

```

Рис. 3.5: Запуск программы в отладчике

Я открыла регистры и проанализировал их, не увидела никаких ошибок.

# Вывод

Я приобрела навыки написания программ использованием подпрограмм. Познакомилась с методами отладки при помощи GDB и его основными возможностями.