

# **Отчёт по лабораторной работе 9**

**Понятие подпрограммы. Отладчик GDB.**

Скобеева Алиса Алексеевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Реализация подпрограмм в NASM . . . . .	6
2.2	Отладка программ с помощью GDB . . . . .	9
2.3	Задание для самостоятельной работы . . . . .	20
<b>3</b>	<b>Выводы</b>	<b>27</b>

## Список иллюстраций

2.1	Программа для вычисления $f(x)$ . . . . .	7
2.2	Результаты выполнения программы для $f(x)$ . . . . .	7
2.3	Программа для вычисления $f(g(x))$ . . . . .	8
2.4	Результаты выполнения программы для $f(g(x))$ . . . . .	9
2.5	Программа вывода сообщения “Hello, world!” . . . . .	10
2.6	Запуск программы в GDB . . . . .	11
2.7	Дизассемблированный код . . . . .	12
2.8	Дизассемблированный код в режиме Intel . . . . .	13
2.9	Установка точки останова . . . . .	14
2.10	Просмотр изменений регистров . . . . .	15
2.11	Дальнейшие изменения регистров . . . . .	16
2.12	Изменение значения переменной . . . . .	17
2.13	Вывод значения регистра <code>edx</code> . . . . .	18
2.14	Изменение регистра <code>ebx</code> . . . . .	19
2.15	Исследование стека и аргументов . . . . .	20
2.16	Программа с подпрограммой для вычисления $f(x)$ . . . . .	21
2.17	Результаты выполнения программы . . . . .	22
2.18	Код с ошибкой . . . . .	23
2.19	Процесс отладки . . . . .	24
2.20	Исправленный код . . . . .	25
2.21	Проверка исправленного кода . . . . .	26

## **Список таблиц**

# 1 Цель работы

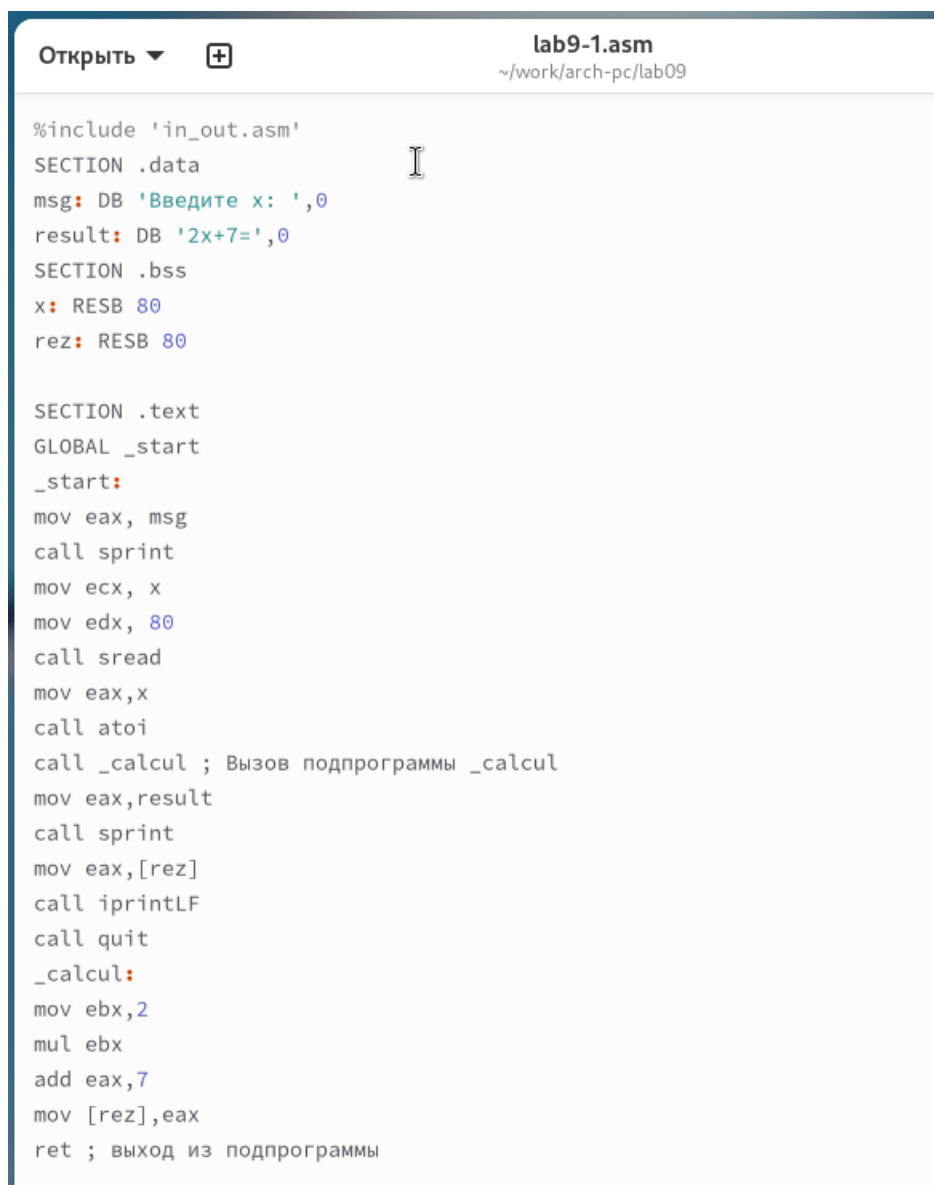
Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.


## 2 Выполнение лабораторной работы

### 2.1 Реализация подпрограмм в NASM

Мы создали каталог для выполнения лабораторной работы №9, перешли в него и добавили файл lab9-1.asm.

В качестве примера рассмотрена программа для вычисления арифметического выражения  $f(x) = 2x + 7$  с использованием подпрограммы calcul. Ввод значения  $x$  осуществляется с клавиатуры, а вычисление производится в подпрограмме.

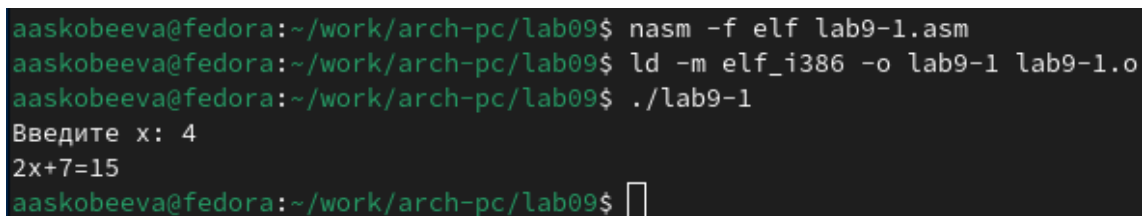


```
Открыть ▾  lab9-1.asm
~/work/arch-pc/lab09

%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
rez: 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, [rez]
call iprintLF
call quit
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы
```

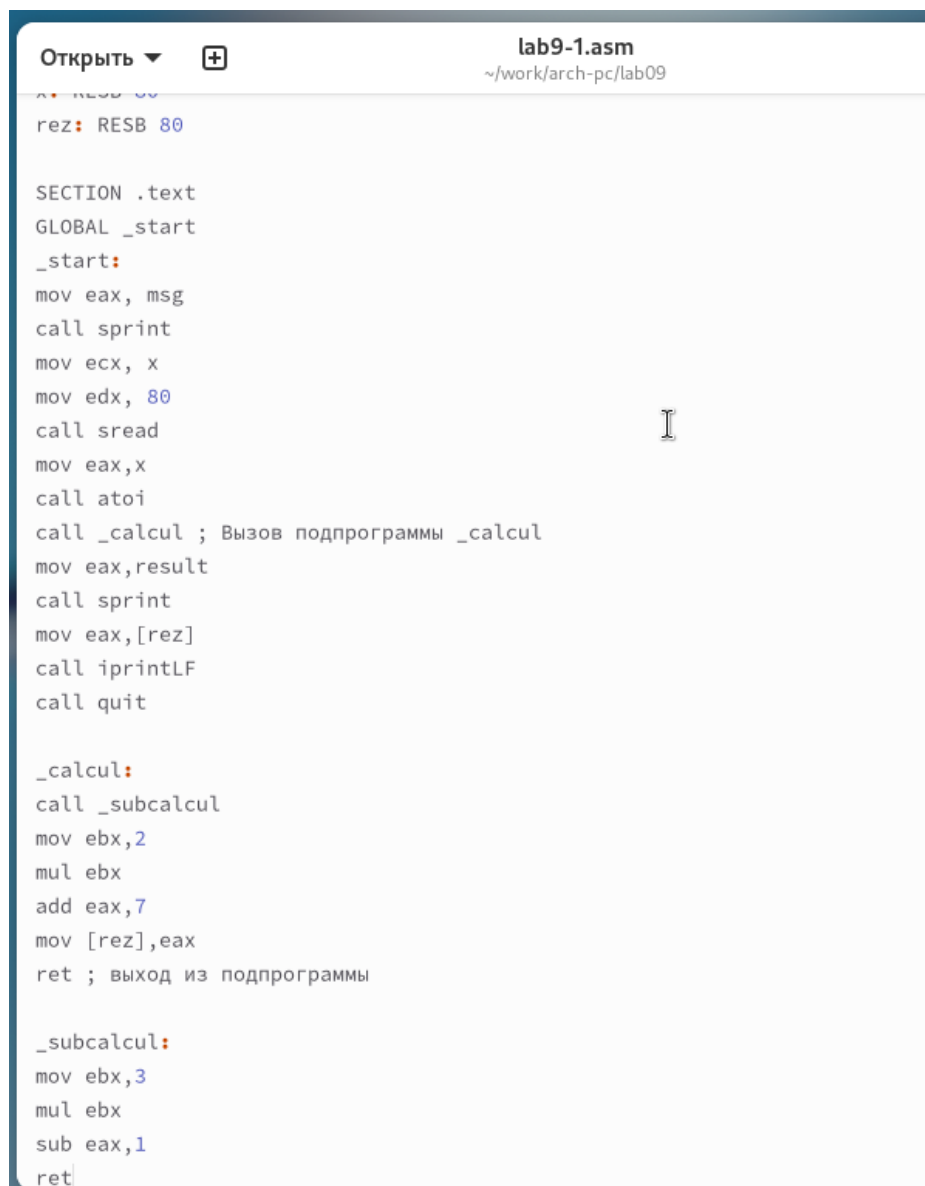
Рис. 2.1: Программа для вычисления  $f(x)$



```
aaskobeeva@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
aaskobeeva@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
aaskobeeva@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 4
2x+7=15
aaskobeeva@fedora:~/work/arch-pc/lab09$
```

Рис. 2.2: Результаты выполнения программы для  $f(x)$

Изменили текст программы, добавив подпрограмму `subcalcul` в `calcul` для вычисления выражения  $f(g(x))$ , где  $f(x) = 2x + 7$ ,  $g(x) = 3x - 1$ , и  $x$  вводится с клавиатуры.



```
Открыть ▾ + lab9-1.asm
~/work/arch-pc/lab09

rez: 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, [rez]
call iprintLF
call quit

_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы

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

Рис. 2.3: Программа для вычисления  $f(g(x))$



```
aaskobeeva@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
aaskobeeva@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
aaskobeeva@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 4
2(3x-1)+7=29
aaskobeeva@fedora:~/work/arch-pc/lab09$
```

Рис. 2.4: Результаты выполнения программы для  $f(g(x))$

## 2.2 Отладка программ с помощью GDB

Мы создали файл lab9-2.asm, содержащий текст программы из листинга 9.2 (программа для вывода сообщения “Hello, world!”). Добавили отладочную информацию для работы с GDB, указав флаг -g при трансляции. Программа была протестирована в GDB.

Открыть ▾ 

lab9-2.asm  
~/work/arch-pc/lab09

```
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. 2.5: Программа вывода сообщения “Hello, world!”

```

aaskobeeva@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
aaskobeeva@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
aaskobeeva@fedora:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.1-1.fc39
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) r
Starting program: /home/aaskobeeva/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])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 4724) exited normally]
(gdb)

```

Рис. 2.6: Запуск программы в GDB

Установили брейкпоинт на метке `_start`, запустили программу и просмотрели дизассемблированный код в различных режимах.

```
aaskobeeva@fedora:~/work/arch-pc/lab09 — gdb lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 4724) exited normally]
(gdb)
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/aaskobeeva/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:11
11      mov eax, 4
(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.
(gdb)
```

Рис. 2.7: Дизассемблированный код

```
aaskobeeva@fedora:~/work/arch-pc/lab09 — gdb lab9-2
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.
(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. 2.8: Дизассемблированный код в режиме Intel

Определили адрес предпоследней инструкции и установили точку останова. Проверили изменения значений регистров с помощью команды `stepi`.

```
aaskobeeva@fedora:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0f0 0xffffd0f0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>

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

native process 4729 (asm) In: _start L11 PC: 0x8049000
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 22.
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint      keep y  0x08049000 lab9-2.asm:11
       breakpoint already hit 1 time
2      breakpoint      keep y  0x08049031 lab9-2.asm:22
(gdb) 
```

Рис. 2.9: Установка точки останова

```
aaskobeeva@fedora:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0f0 0xffffd0f0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>

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

native process 4729 (asm) In: _start L12 PC: 0x8049005
eip      0x8049000 0x8049000 <_start>
eflags   0x202     [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
cs       0x23      35
ss       0x2b      43
ds       0x2b      43
es       0x2b      43
fs       0x0       0
gs       0x0       0
(gdb) si
(gdb) 
```

Рис. 2.10: Просмотр изменений регистров

The screenshot shows a GDB terminal window with the title bar "aaskobeeva@fedora:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", displays the current values of general-purpose registers: `eax` (0x8, 8), `ecx` (0x804a000, 134520832), `edx` (0x8, 8), `ebx` (0x1, 1), `esp` (0xffffd0f0, 0xffffd0f0), `ebp` (0x0, 0x0), `esi` (0x0, 0), `edi` (0x0, 0), and `eip` (0x8049016, 0x8049016 <\_start+22>). The middle section shows assembly code starting from address 0x8049000, with instructions like `mov eax, 0x4`, `mov ebx, 0x1`, `mov ecx, 0x804a000`, `mov edx, 0x8`, `int 0x80`, and `mov eax, 0x4` at address 0x8049016. The bottom section shows the state of segment registers: `ss` (0x2b, 43), `ds` (0x2b, 43), `es` (0x2b, 43), `fs` (0x0, 0), and `gs` (0x0, 0). The prompt "(gdb) si" is repeated five times, indicating step-through execution.

```
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd0f0 0xffffd0f0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>

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

native process 4729 (asm) In: _start          L16  PC: 0x8049016
ss      0x2b      43
ds      0x2b      43
es      0x2b      43
fs      0x0      0
gs      0x0      0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) 
```

Рис. 2.11: Дальнейшие изменения регистров

Посмотрели значения переменных `msg1` и `msg2` по имени и адресу соответственно. Изменили значение первого символа переменной `msg1`.



The screenshot shows a GDB terminal window with the title bar "aaskobeeva@fedora:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", displays the current values of general-purpose registers: 

Register	Value	Comment
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd0f0	0xffffd0f0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>

. The middle section displays assembly code with addresses and instructions: 

```
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
```

. The bottom section shows the GDB prompt and a series of commands and their outputs: 

```
native process 4729 (asm) In: _start          L16  PC: 0x8049016
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) set {char}0x804a008='L'
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "Lorld!\n\034"
(gdb) 
```

Рис. 2.12: Изменение значения переменной

Вывели значение регистра `edx` в различных форматах (шестнадцатеричном, двоичном, символьном) и изменили его значение.

The screenshot shows a GDB terminal window with the title bar "aaskobeeva@fedora:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", displays the values of general-purpose registers: 

Register	Hex Value	Decimal Value
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd0f0	0xffffd0f0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x8049016	0x8049016 <_start+22>

. The middle section displays assembly code with addresses and instructions: 

```
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
```

. The bottom section shows the GDB prompt and a series of commands and their outputs: 

```
native process 4729 (asm) In: _start          L16    PC: 0x8049016
(gdb) p/s $ecx
$3 = 134520832
(gdb) p/x $ecx
$4 = 0x804a000
(gdb) p/s $edx
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb)
```

Рис. 2.13: Вывод значения регистра edx

```
aaskobeeva@fedora:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd0f0 0xffffd0f0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>

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

native process 4729 (asm) In: _start L16 PC: 0x8049016
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb) set $ebx='2'
(gdb) p/s $ebx
$8 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$9 = 2
$10 = 2
(gdb)
```

Рис. 2.14: Изменение регистра ebx

Скопировали файл lab8-2.asm из предыдущей работы, создали исполняемый файл и запустили его с аргументами в GDB, установив точку останова перед началом программы. Исследовали адреса, связанные с аргументами командной строки.

```
aaskobeeva@fedora:~/work/arch-pc/lab09 — gdb --args lab9-3 argument 1 ...
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/aaskobeeva/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument
\ 3

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

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd0c0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd287: "/home/aaskobeeva/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd2b2: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd2bb: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd2bd: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd2c6: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd2c8: "argument 3"
(gdb) 
```

Рис. 2.15: Исследование стека и аргументов

## 2.3 Задание для самостоятельной работы

Преобразовали программу из лабораторной работы №8 (задание №1) для вычисления значения функции  $f(x)$  как подпрограмму.



```
Открыть ▾ + task-1.asm
~/work/arch-pc/lab09

mov eax, fx
call sprintLF
pop ecx
pop edx
sub ecx, 1
mov esi, 0

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

loop next

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


subproc:
sub eax, 1
mov ebx, 10
mul ebx
ret
```


Рис. 2.16: Программа с подпрограммой для вычисления  $f(x)$

```
aaskobeeva@fedora:~/work/arch-pc/lab09$  
aaskobeeva@fedora:~/work/arch-pc/lab09$ nasm -f elf task-1.asm  
aaskobeeva@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 task-1.o -o task-1  
aaskobeeva@fedora:~/work/arch-pc/lab09$ ./task-1 3  
f(x)= 10(x - 1)  
Результат: 20  
aaskobeeva@fedora:~/work/arch-pc/lab09$ ./task-1 4 5 1 6  
f(x)= 10(x - 1)  
Результат: 120  
aaskobeeva@fedora:~/work/arch-pc/lab09$
```

Рис. 2.17: Результаты выполнения программы

Проанализировали и исправили ошибку в программе для вычисления  $(3 + 2) * 4 + 5$  с помощью отладчика GDB. Ошибка заключалась в неправильном порядке аргументов инструкции `add` и некорректном завершении программы (использование `ebx` вместо `eax`).

Открыть ▾ 

task-2.asm  
~/work/arch-pc/lab09 

```
%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. 2.18: Код с ошибкой

```
aaskobeeva@fedora:~/work/arch-pc/lab09 — gdb task-2

Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd0f0 0xffffd0f0
ebp      0x0      0x0
esi      0x0      0
edi      0xa      10
eip      0x8049100 0x8049100 <_start+24>

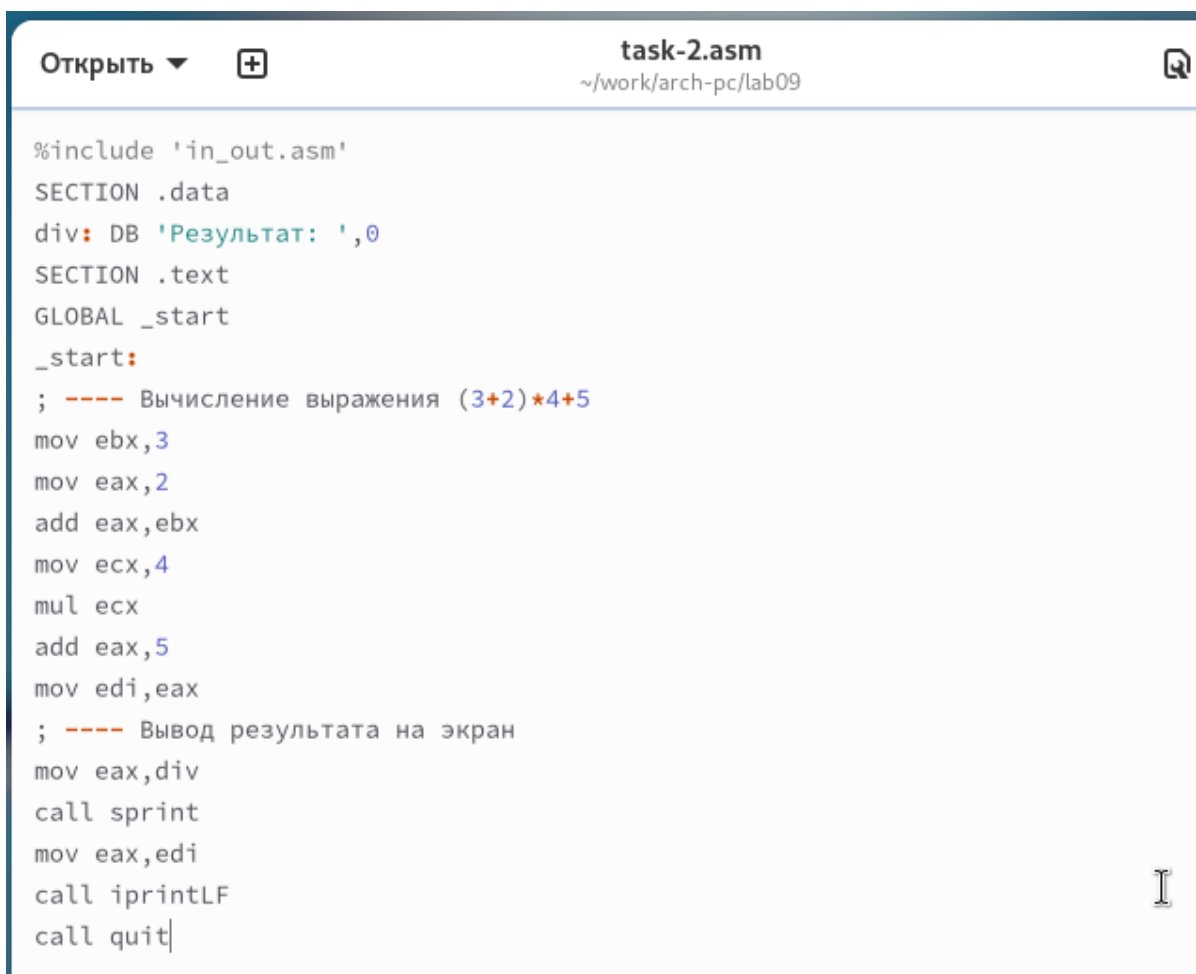
0x80490f4 <_start+12> mov    ecx,0x4
0x80490f9 <_start+17> mul    ecx
0x80490fb <_start+19> add    ebx,0x5
0x80490fe <_start+22> mov    edi,ebx
>0x8049100 <_start+24> mov    eax,0x804a000
0x8049105 <_start+29> call   0x804900f <sprint>
0x804910a <_start+34> mov    eax,edi
0x804910c <_start+36> call   0x8049086 <iprintLF>
0x8049111 <_start+41> call   0x80490db <quit>

native process 5014 (asm) In: _start L16 PC: 0x8049100
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at task-2.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.19: Процесс отладки







```
Открыть ▾  task-2.asm  
~/work/arch-pc/lab09   
  
%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 eax,ebx  
mov ecx,4  
mul ecx  
add eax,5  
mov edi,eax  
; ---- Вывод результата на экран  
mov eax,div  
call sprint  
mov eax,edi  
call iprintLF  
call quit
```

Рис. 2.20: Исправленный код

The screenshot shows a GDB terminal window titled "aaskobeeva@fedora:~/work/arch-pc/lab09 — gdb task-2". The window is divided into several sections. At the top, a register window shows "eax" with a value of "25". Below this, a memory window shows two memory locations: "fffd0f0" and "xffffd0f0", both with the text "[ Register Values Unavailable ]". Below the memory window, there are two input fields with the values "9" and "5". The main window displays assembly code for the function "\_start":  
0x8049100 <\_start+24> mul eax,0x804a000  
0x804910a <\_start+34> mov eax,edi  
0x804910c <\_start+36> call 0x8049086 <iprintLF>  
0x8049111 <\_start+41> call 0x80490db <quit>  
Below the assembly code, the GDB prompt shows the execution of the program: "native process 5044 (asm) In: \_start" with "L16" and "PC: 0x8049100". The output of the program is "Результат: 25". The GDB prompt also shows "Breakpoint 1 (process 5044) hit" and "Continuing.". The final output is "[Inferior 1 (process 5044) exited normally]".

```
eax 25

fffd0f0      xffffd0f0
[ Register Values Unavailable ]

9 5

0x8049100 <_start+24> mul    eax,0x804a000
0x804910a <_start+34> mov    eax,edi
0x804910c <_start+36> call   0x8049086 <iprintLF>
0x8049111 <_start+41> call   0x80490db <quit>

native process 5044 (asm) In: _start L16 PC: 0x8049100
Breakpoint 1 (process 5044) hit L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 5044) exited normally]
(gdb)
```

Рис. 2.21: Проверка исправленного кода

## 3 Выводы

Мы освоили работу с подпрограммами и отладчиком GDB, а также изучили методы анализа и исправления ошибок в ассемблерных программах.