

Отчёт по лабораторной работе 9

дисциплина: Архитектура компьютера

Абдулрахман Джатал

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	27

Список иллюстраций

2.1	Программа lab9-1.asm	7
2.2	Запуск программы lab9-1.asm	8
2.3	Программа lab9-1.asm	9
2.4	Запуск программы lab9-1.asm	9
2.5	Программа lab9-2.asm	10
2.6	Запуск программы lab9-2.asm в отладчике	11
2.7	Дизассемблированный код	12
2.8	Дизассемблированный код в режиме интел	13
2.9	Точка остановки	14
2.10	Изменение регистров	15
2.11	Изменение регистров	16
2.12	Изменение значения переменной	17
2.13	Вывод значения регистра	18
2.14	Вывод значения регистра	19
2.15	Вывод значения регистра	20
2.16	Программа lab9-4.asm	21
2.17	Запуск программы lab9-4.asm	22
2.18	Код с ошибкой	23
2.19	Отладка	24
2.20	Код исправлен	25
2.21	Проверка работы	26

Список таблиц

1 Цель работы

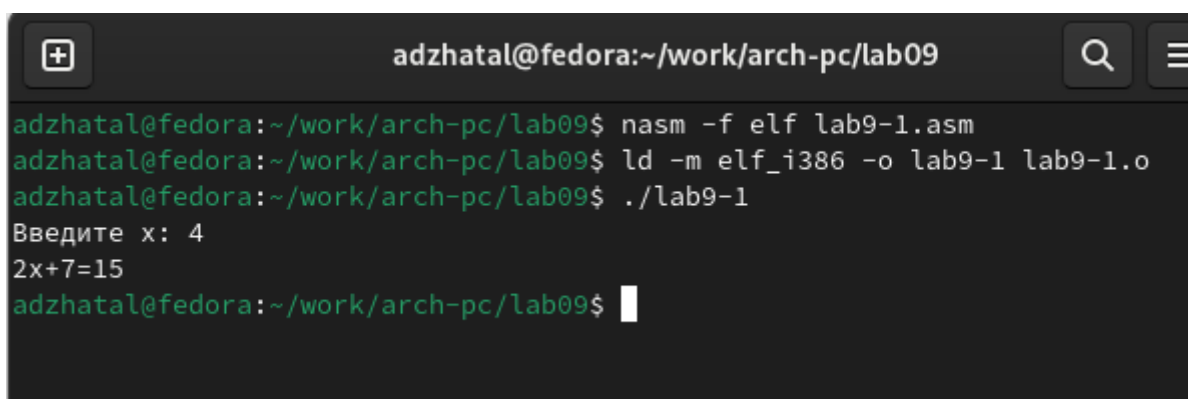
Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

1. Создал каталог для выполнения лабораторной работы № 9, перешел в него и создал файл lab9-1.asm.
2. В качестве примера рассмотрим программу вычисления арифметического выражения $f(x) = 2x + 7$ с помощью подпрограммы calcul. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме.

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 result: DB '2x+7=',0
5 SECTION .bss
6 x: RESB 80
7 rez: RESB 80
8
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax,x
18 call atoi
19 call _calcul ; Вызов подпрограммы _calcul
20 mov eax,result
21 call sprint
22 mov eax,[rez]
23 call iprintLF
24 call quit
25 _calcul:
26 mov ebx,2
27 mul ebx
28 add eax,7
29 mov [rez],eax
30 ret ; выход из подпрограммы
```

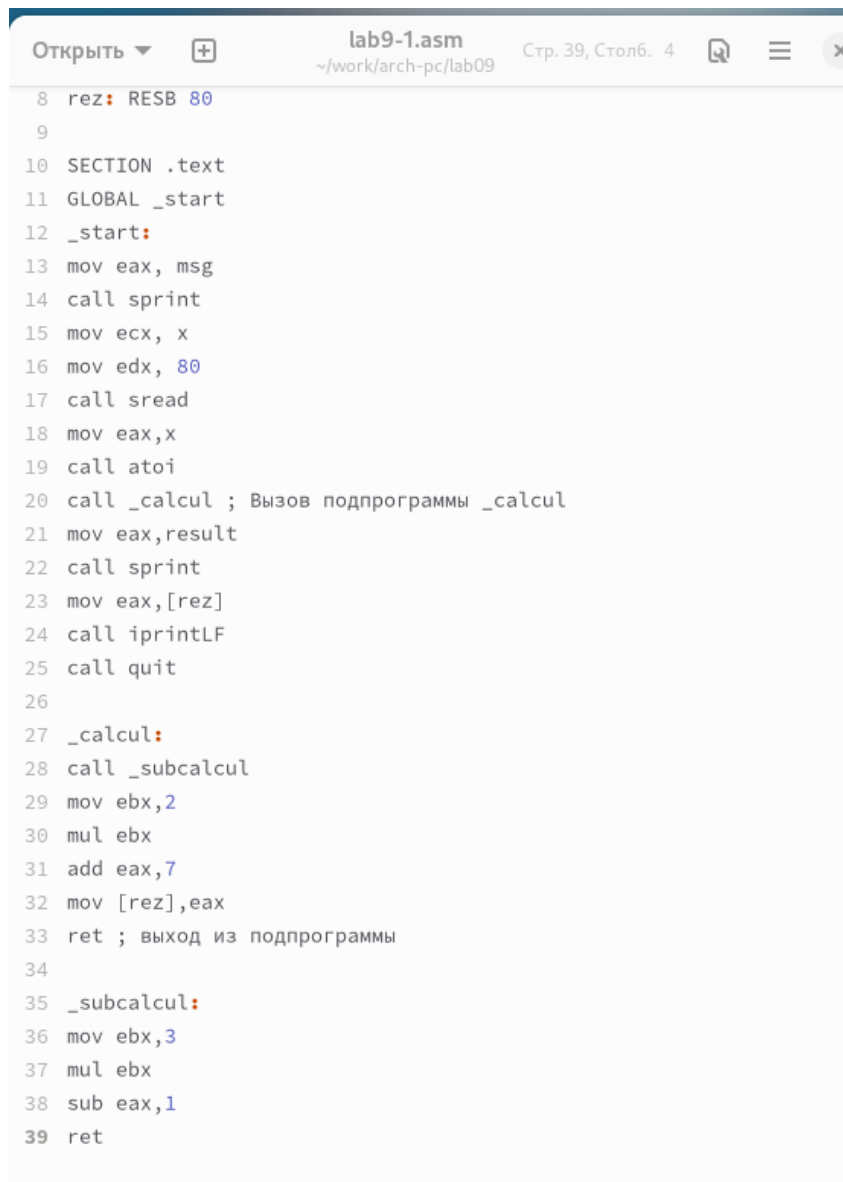
Рис. 2.1: Программа lab9-1.asm

A terminal window with a dark background. The title bar shows 'adzhatal@fedora:~/work/arch-pc/lab09'. The terminal content shows the following commands and output:

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

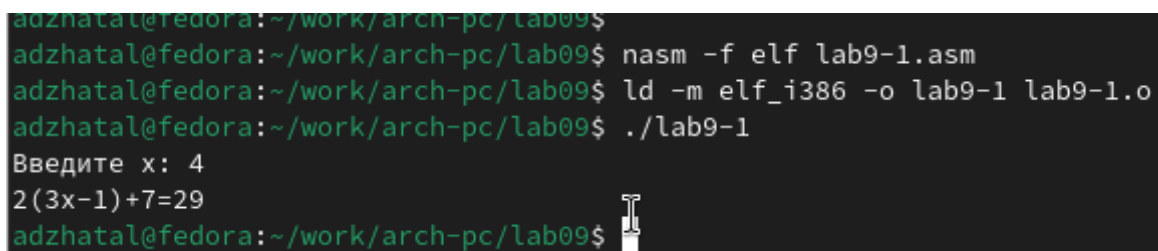
Рис. 2.2: Запуск программы lab9-1.asm

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



```
Открыть ▾ + lab9-1.asm Стр. 39, Столб. 4
~/work/arch-pc/lab09
8 rez: RESB 80
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprint
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax, x
19 call atoi
20 call _calcul ; Вызов подпрограммы _calcul
21 mov eax, result
22 call sprint
23 mov eax, [rez]
24 call iprintLF
25 call quit
26
27 _calcul:
28 call _subcalcul
29 mov ebx, 2
30 mul ebx
31 add eax, 7
32 mov [rez], eax
33 ret ; выход из подпрограммы
34
35 _subcalcul:
36 mov ebx, 3
37 mul ebx
38 sub eax, 1
39 ret
```

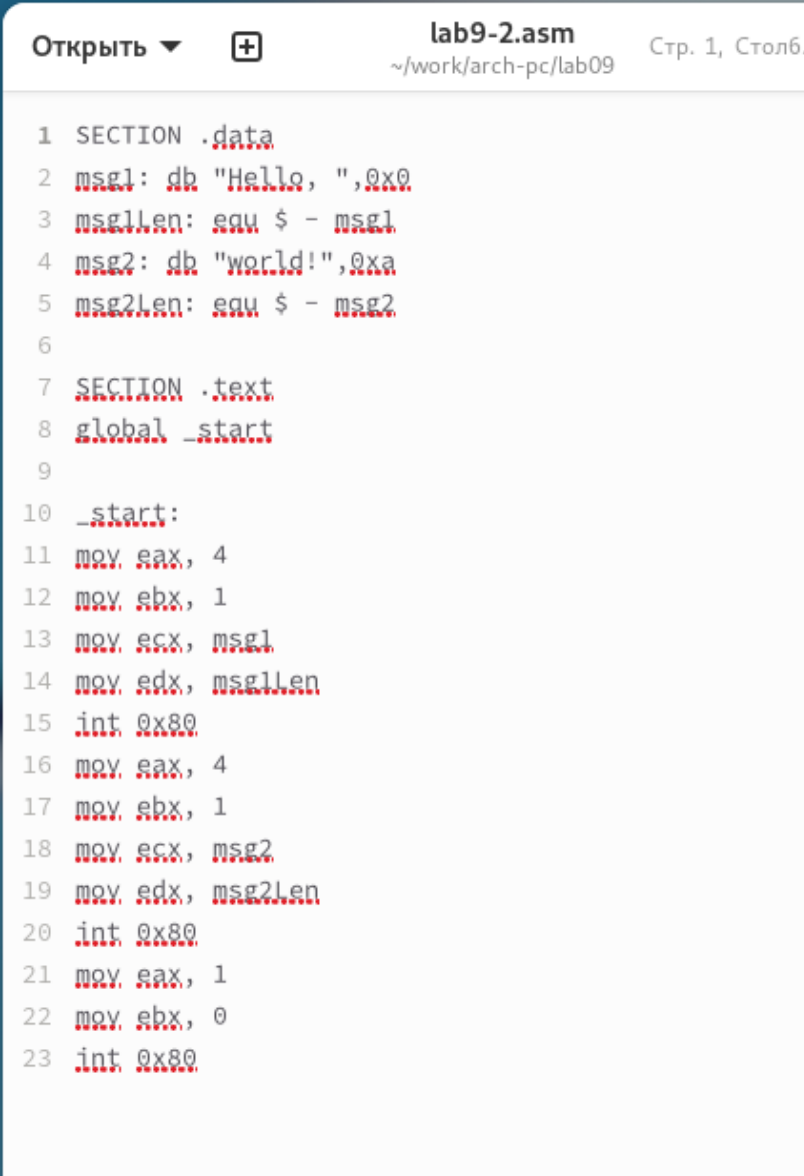
Рис. 2.3: Программа lab9-1.asm



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

Рис. 2.4: Запуск программы lab9-1.asm

4. Создал файл lab9-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!).



```
1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msg1len: equ $ - msg1
4 msg2: db "world!",0xa
5 msg2len: equ $ - msg2
6
7 SECTION .text
8 global _start
9
10 _start:
11 mov eax, 4
12 mov ebx, 1
13 mov ecx, msg1
14 mov edx, msg1len
15 int 0x80
16 mov eax, 4
17 mov ebx, 1
18 mov ecx, msg2
19 mov edx, msg2len
20 int 0x80
21 mov eax, 1
22 mov ebx, 0
23 int 0x80
```

Рис. 2.5: Программа lab9-2.asm

Получил исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом '-g'.

Загрузил исполняемый файл в отладчик gdb. Проверил работу программы,

запустив ее в оболочке GDB с помощью команды `run` (сокращённо `r`).

```
adzhatal@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
adzhatal@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
adzhatal@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/adzhatal/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 3606) exited normally]
(gdb) █
```

Рис. 2.6: Запуск программы `lab9-2.asm` в отладчике

Для более подробного анализа программы установите брейкпоинт на метку `start`, с которой начинается выполнение любой ассемблерной программы, и запустите её. Посмотрите дизассемблированный код программы.

```
adzhatal@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 3606) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/adzhatal/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

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

=> 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 интел

На предыдущих шагах была установлена точка остановки по имени метки (`_start`). Проверил это с помощью команды `info breakpoints` (кратко `i b`). Установил еще одну точку остановки по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определил адрес предпоследней инструкции (`mov ebx,0x0`) и установил точку.

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

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd100 0xffffd100
ebp      0x0      0x0
esi      0x0      0
edi      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
0x8049020 <_start+32>     mov     ecx,0x804a008
0x8049025 <_start+37>     mov     edx,0x7

native process 3610 (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: Точка остановки

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Выполнил 5 инструкций с помощью команды `stepi` (или `si`) и проследил за изменением значений регистров.

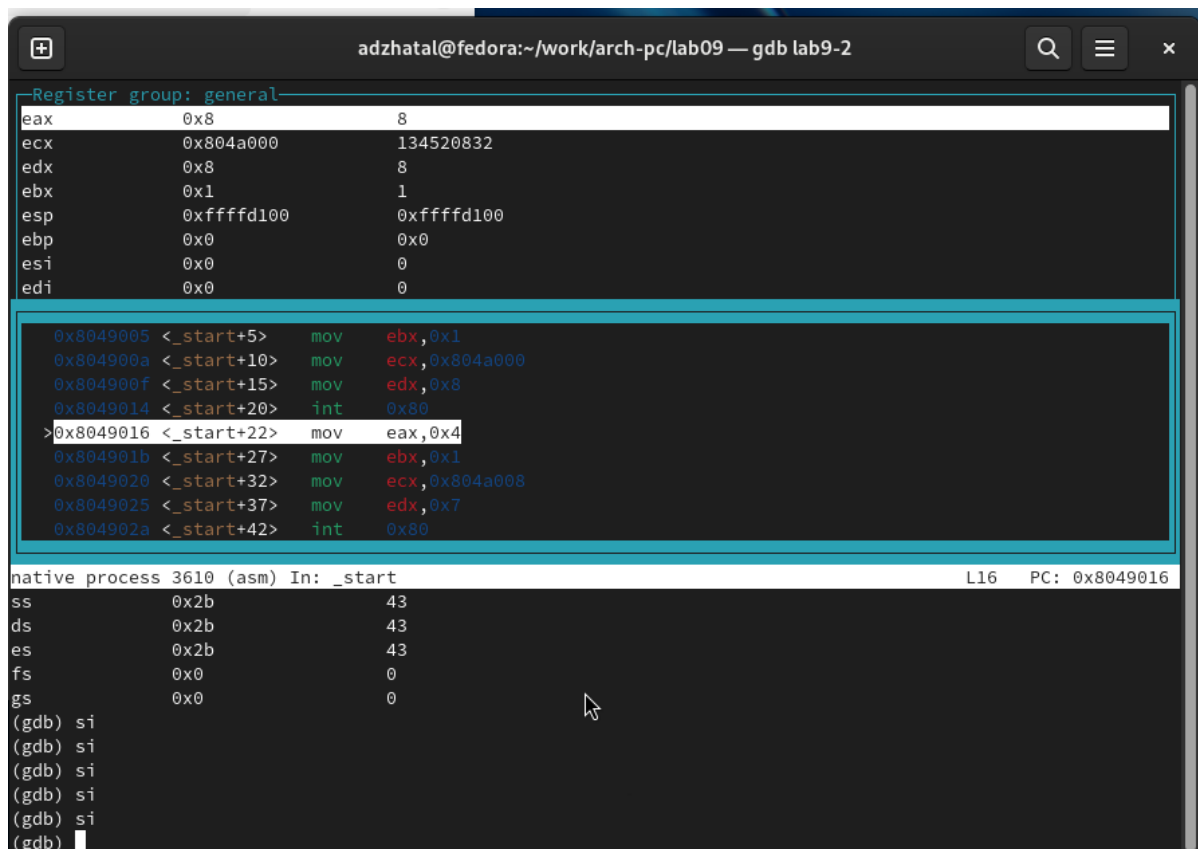
```
adzhatal@fedora:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd100 0xffffd100
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

>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

native process 3610 (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--si
cs       0x23      35
ss       0x2b      43
ds       0x2b      43
es       0x2b      43
fs       0x0      0
gs       0x0      0
(gdb) si
(gdb)
```

Рис. 2.10: Изменение регистров



The screenshot shows a GDB window titled "adzhatal@fedora:~/work/arch-pc/lab09 — gdb lab9-2". The window is divided into several sections. At the top, there is a "Register group: general" section displaying the values of general-purpose registers:

Register	Value	Comment
eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd100	0xffffd100
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0

. Below this, a list of assembly instructions is shown, with the instruction at address 0x8049016 selected:

```
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
```

. At the bottom, the "native process 3610 (asm) In: _start" section shows segment registers:

Segment	Value	Comment
ss	0x2b	43
ds	0x2b	43
es	0x2b	43
fs	0x0	0
gs	0x0	0

. The bottom status bar shows "(gdb) si" repeated five times, indicating the instruction pointer is being stepped through.

Рис. 2.11: Изменение регистров

Посмотрел значение переменной `msg1` по имени. Посмотрел значение переменной `msg2` по адресу.

Изменить значение для регистра или ячейки памяти можно с помощью команды `set`, задав ей в качестве аргумента имя регистра или адрес. Изменил первый символ переменной `msg1`.


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

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd100 0xffffd100
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0

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

native process 3610 (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

Вывел в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx.

```
adzhatal@fedora:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd100 0xffffd100
ebp      0x0      0
esi      0x0      0
edi      0x0      0

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

native process 3610 (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

С помощью команды set изменил значение регистра ebx

The screenshot shows a GDB terminal window with the title bar 'adzhatal@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 registers: `eax` (0x8), `ecx` (0x804a000), `edx` (0x8), `ebx` (0x2), `esp` (0xffffd100), `ebp` (0x0), `esi` (0x0), and `edi` (0x0). The middle section shows a list of disassembled instructions with their addresses and mnemonics: `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`, and `0x804902a <_start+42> int 0x80`. The bottom section shows the GDB prompt and a series of commands and their outputs: `(gdb) p/t $edx` results in `$6 = 1000`; `(gdb) p/x $edx` results in `$7 = 0x8`; `(gdb) set $ebx='2'`; `(gdb) p/s $ebx` results in `$8 = 50`; `(gdb) set $ebx=2`; `(gdb) p/s $ebx` results in `$9 = 2`; and `(gdb)` is followed by a cursor.

Рис. 2.14: Вывод значения регистра

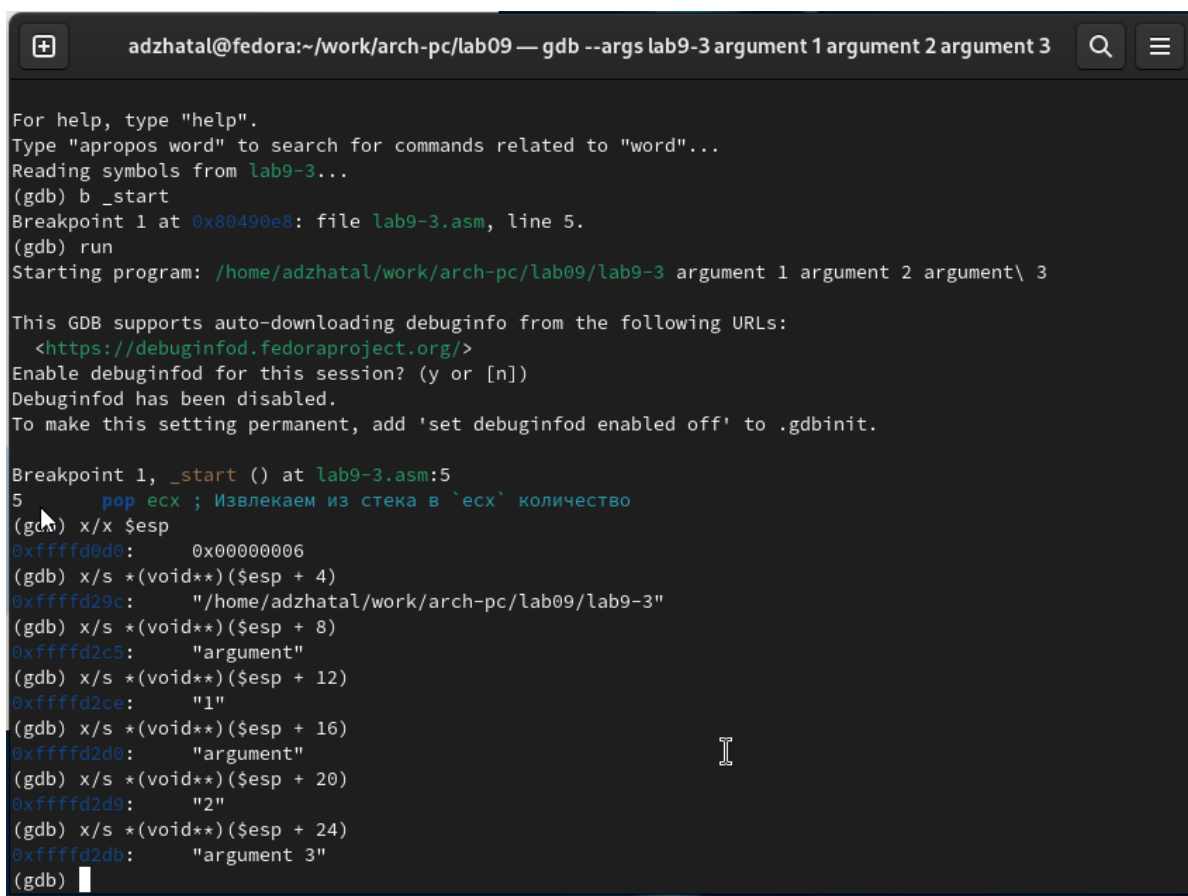
5. Скопировал файл `lab8-2.asm`, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки. Создал исполняемый файл. Для загрузки в `gdb` программы с аргументами необходимо использовать ключ `-args`. Загрузил исполняемый файл в отладчик, указав аргументы.

Для начала установил точку останова перед первой инструкцией в программе и запустил ее.

Адрес вершины стека храниться в регистре `esp` и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5 – это имя программы `lab9-3` и непосредственно аргументы: аргумент1, аргумент, 2 и ‘аргумент 3’.

Посмотрел остальные позиции стека – по адресу `[esp+4]` располагается адрес в

памяти где находится имя программы, по адресу [esp+8] храниться адрес первого аргумента, по адресу [esp+12] – второго и т.д.



```
adzhatal@fedora:~/work/arch-pc/lab09 — gdb --args lab9-3 argument 1 argument 2 argument 3

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) run
Starting program: /home/adzhatal/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

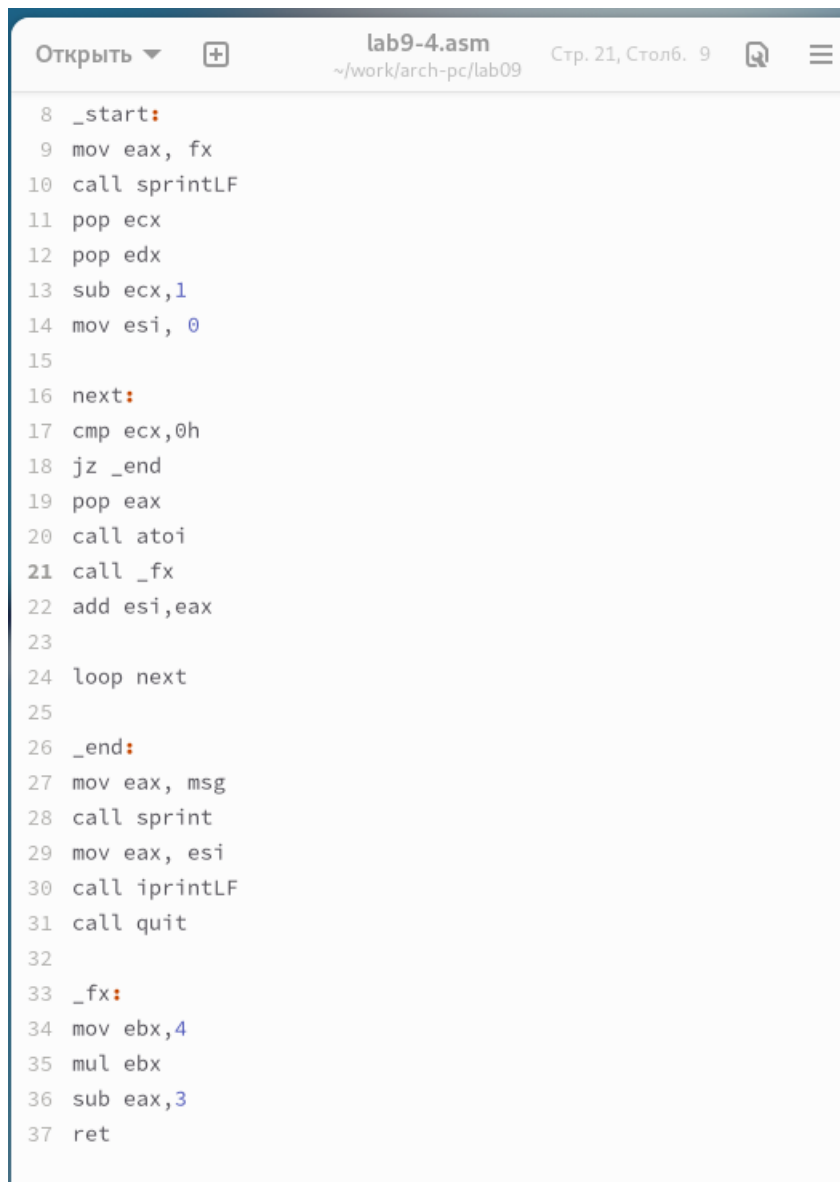
This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd0d0:  0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd29c:  "/home/adzhatal/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd2c5:  "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd2ce:  "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd2d0:  "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd2d9:  "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd2db:  "argument 3"
(gdb)
```

Рис. 2.15: Вывод значения регистра

Объясню, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12]) - шаг равен размеру переменной - 4 байтам.

- Преобразовал программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму.





```
8  _start:
9  mov eax, fx
10 call sprintf
11 pop ecx
12 pop edx
13 sub ecx,1
14 mov esi, 0
15
16 next:
17 cmp ecx,0h
18 jz _end
19 pop eax
20 call atoi
21 call _fx
22 add esi,eax
23
24 loop next
25
26 _end:
27 mov eax, msg
28 call sprintf
29 mov eax, esi
30 call iprintLF
31 call quit
32
33 _fx:
34 mov ebx,4
35 mul ebx
36 sub eax,3
37 ret
```

Рис. 2.16: Программа lab9-4.asm

```
adzhatal@fedora:~/work/arch-pc/lab09$  
adzhatal@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-4.asm  
adzhatal@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-4.o -o lab9-4  
adzhatal@fedora:~/work/arch-pc/lab09$  
adzhatal@fedora:~/work/arch-pc/lab09$ ./lab9-4 1  
f(x)= 4x - 3  
Результат: 1  
adzhatal@fedora:~/work/arch-pc/lab09$ ./lab9-4 2  
f(x)= 4x - 3  
Результат: 5  
adzhatal@fedora:~/work/arch-pc/lab09$ ./lab9-4 2 3 4 6  
f(x)= 4x - 3  
Результат: 48  
adzhatal@fedora:~/work/arch-pc/lab09$
```

Рис. 2.17: Запуск программы lab9-4.asm

7. В листинге приведена программа вычисления выражения $(3 + 2) * 4 + 5$. При запуске данная программа дает неверный результат. Проверил это. С помощью отладчика GDB, анализируя изменения значений регистров, определяю ошибку и исправлю ее.

Открыть ▾  lab9-5.asm Стр. 15, Столб. 33 

```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 SECTION .text
5 GLOBAL _start
6 _start:
7 ; ---- Вычисление выражения (3+2)*4+5
8 mov ebx,3
9 mov eax,2
10 add ebx,eax
11 mov ecx,4
12 mul ecx
13 add ebx,5
14 mov edi,ebx
15 ; ---- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF
20 call quit
```

Рис. 2.18: Код с ошибкой

The screenshot shows a GDB terminal window with the title bar "adzhatal@fedora:~/work/arch-pc/lab09 — gdb lab9-5". The window is divided into three main sections. The top section, titled "Register group: general", lists the values of registers: eax (0x8), ecx (0x4), edx (0x0), ebx (0xa), esp (0xffffd100), ebp (0x0), esi (0x0), and edi (0xa). The middle section displays assembly code with addresses and instructions: 0x80490f4: mov ecx, 0x4; 0x80490f9: mul ecx; 0x80490fb: add ebx, 0x5; 0x80490fe: mov edi, ebx; 0x8049100: mov eax, 0x804a000; 0x8049105: call 0x804900f <sprint>; 0x804910a: mov eax, edi; 0x804910c: call 0x8049086 <iprintLF>; 0x8049111: call 0x80490db <quit>. The bottom section shows the GDB prompt and commands: "native process 3948 (asm) In: _start", "To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.", "Breakpoint 1, _start () at lab9-5.asm:8", and several "(gdb) si" commands. The status bar at the bottom indicates "L16 PC: 0x8049100".

```
Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd100 0xffffd100
ebp      0x0      0x0
esi      0x0      0
edi      0xa      10

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>

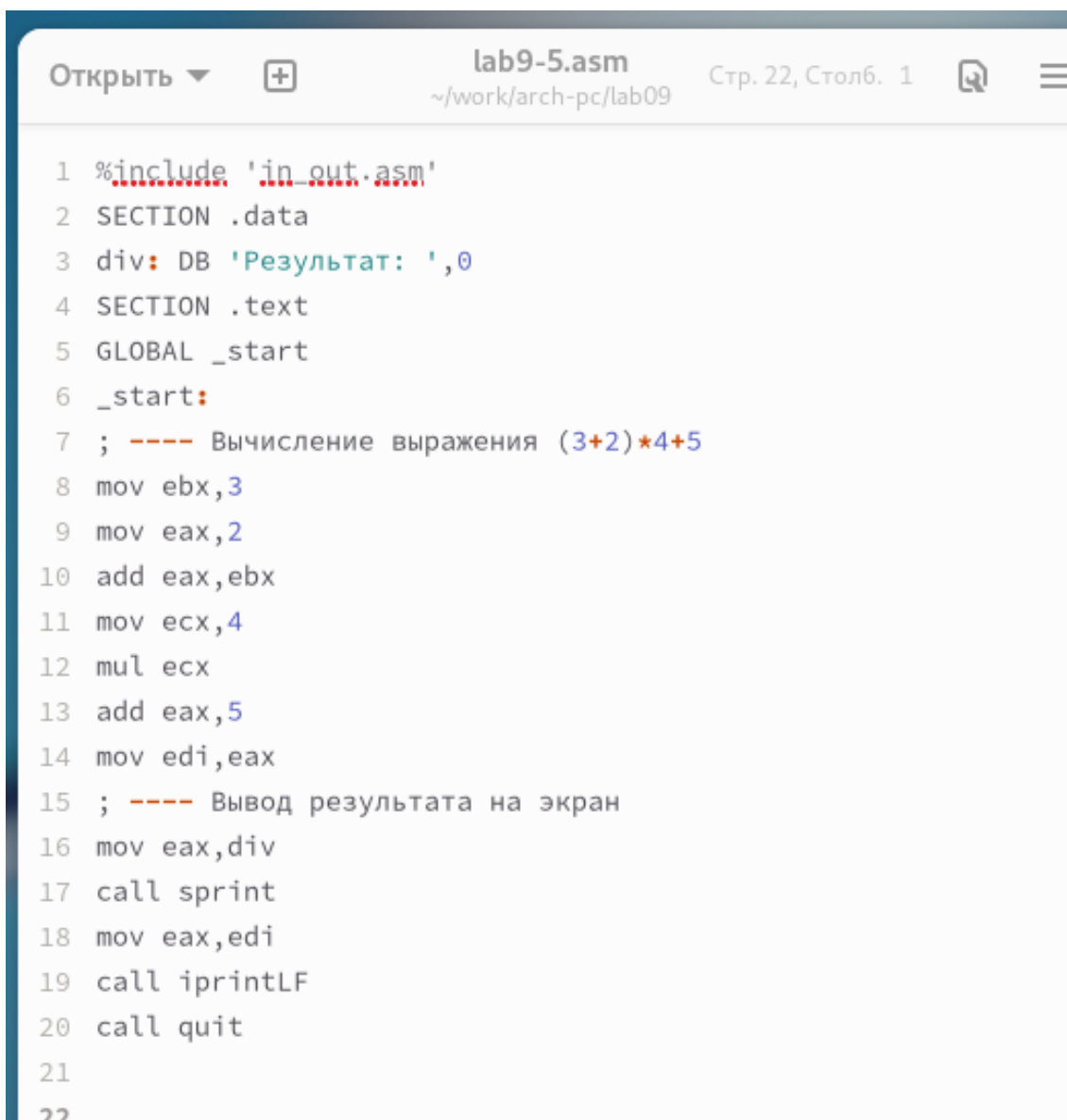
native process 3948 (asm) In: _start L16 PC: 0x8049100
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-5.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.19: Отладка

Отмечу, что перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax

Исправленный код программы



```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 SECTION .text
5 GLOBAL _start
6 _start:
7 ; ---- Вычисление выражения (3+2)*4+5
8 mov ebx,3
9 mov eax,2
10 add eax,ebx
11 mov ecx,4
12 mul ecx
13 add eax,5
14 mov edi,eax
15 ; ---- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF
20 call quit
21
22
```

Рис. 2.20: Код исправлен

```
adzhatal@fedora:~/work/arch-pc/lab09 — gdb lab9-5
eax 5 25
ffffd100 xffffd100
[ Register Values Unavailable ]

0x80490fe <_start+22> mov edi,eax
0x8049100 <_start+24> add eax,eb804a000
0x8049105 <_start+29> call 0x804900f <sprint>
0x804910a <_start+34> mul eax,edi
0x804910c <_start+36> call 0x8049086 <iprintLF>
>0x8049111 <_start+41> call 0x80490db <quit>
                                04a000
                                rint>

native process 4005 (asm) In: _start L14 PC: 0x80490fe
No process (asm) In: L?? PC: ??
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 4005) exited normally]
(gdb)
```

Рис. 2.21: Проверка работы

3 Выводы

Освоили работу с подпрограммами и отладчиком.