

Отчёт по лабораторной работе 9

Понятие подпрограммы. Отладчик GDB.

Коне Абубакар

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	10
2.5	Программа lab9-2.asm	11
2.6	Запуск программы lab9-2.asm в отладчике	12
2.7	Дизассимилированный код	13
2.8	Дизассимилированный 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

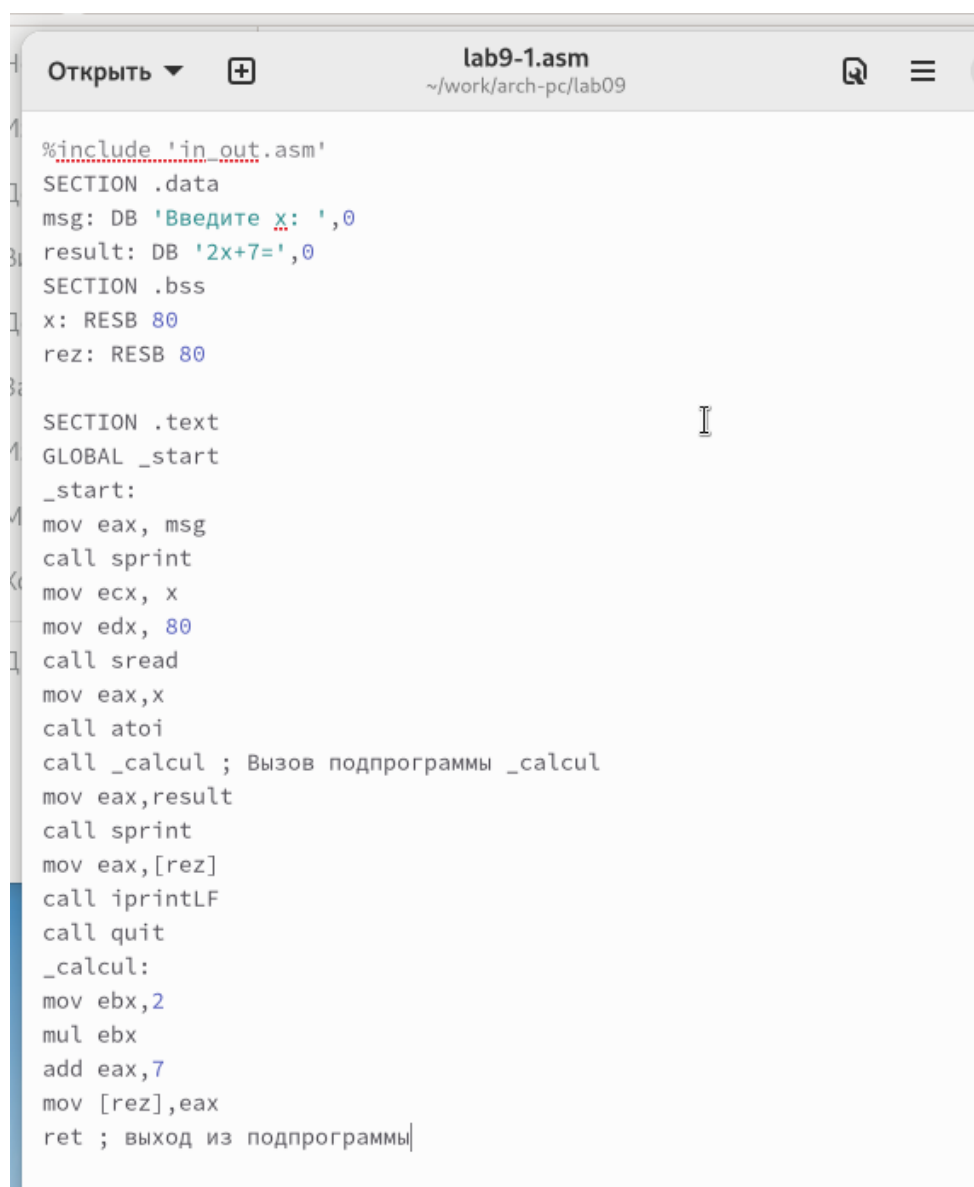
Список таблиц

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.



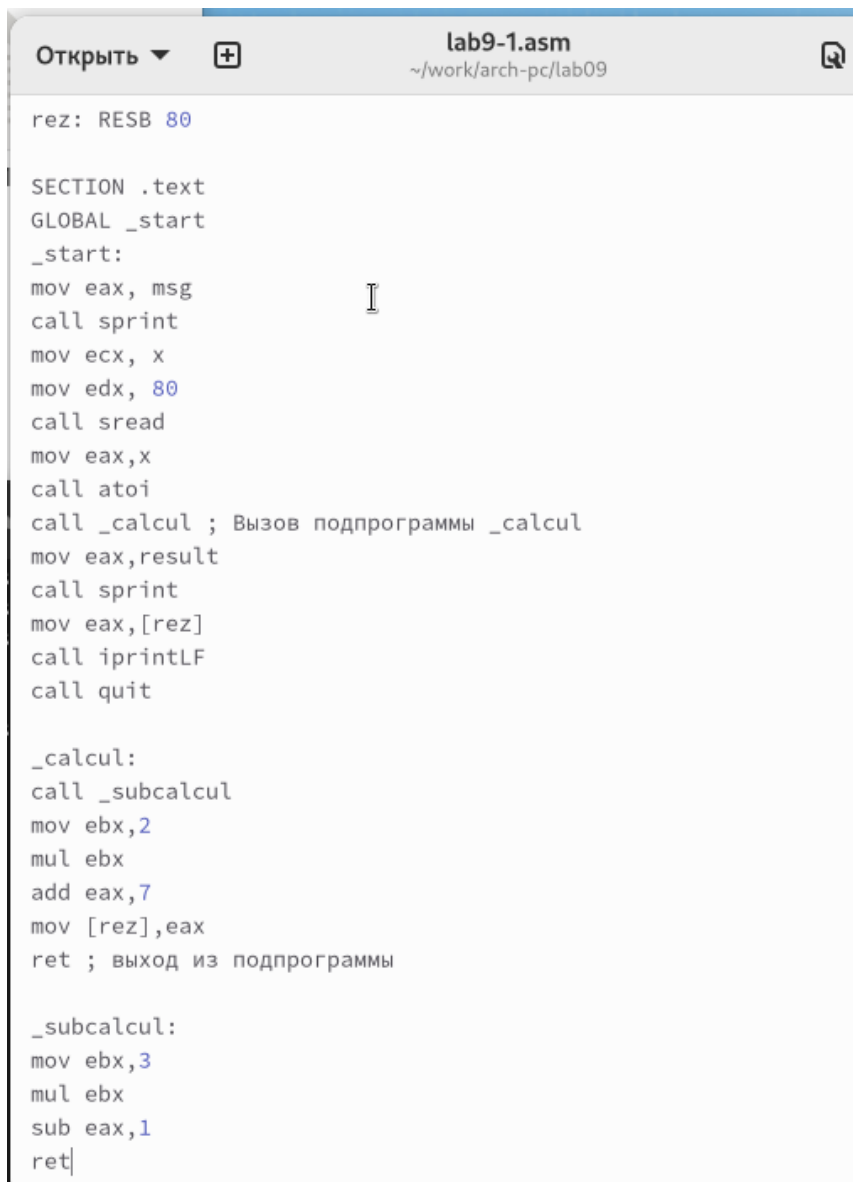
```
1  %include 'in_out.asm'
2  SECTION .data
3  msg: DB 'Введите x: ',0
4  result: DB '2x+7=',0
5  SECTION .bss
6  x: RESB 80
7  rez: RESB 80
8
9  SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax, x
18 call atoi
19 call _calcul ; 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

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

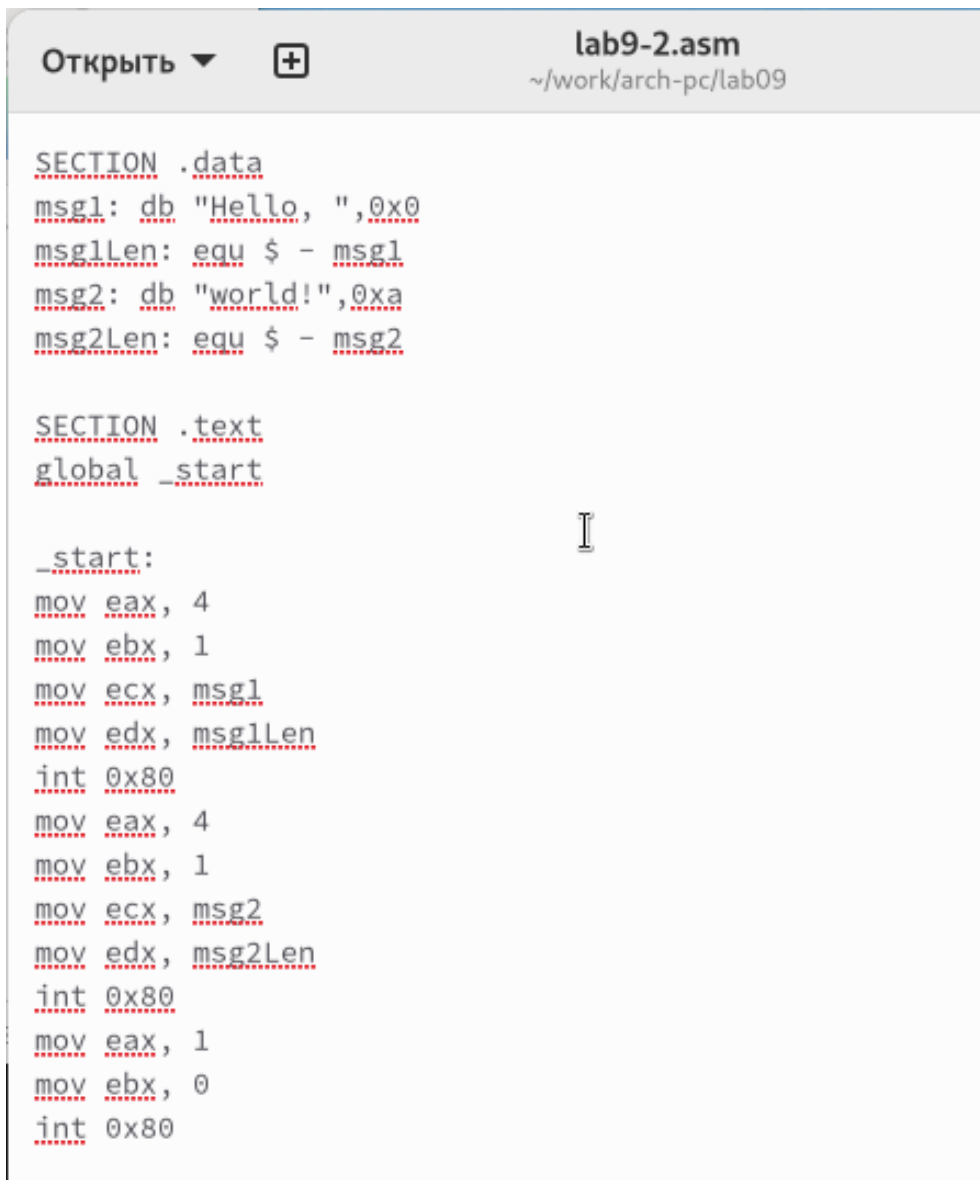
_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret
```

Рис. 2.3: Программа lab9-1.asm

```
[aboubakarkone@fedora lab09]$  
[aboubakarkone@fedora lab09]$ nasm -f elf lab9-1.asm  
[aboubakarkone@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o  
[aboubakarkone@fedora lab09]$ ./lab9-1  
Введите x: 4  
2(3x-1)+7=29  
[aboubakarkone@fedora 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!).



```
Открыть ▾ + lab9-2.asm
~/work/arch-pc/lab09

SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2

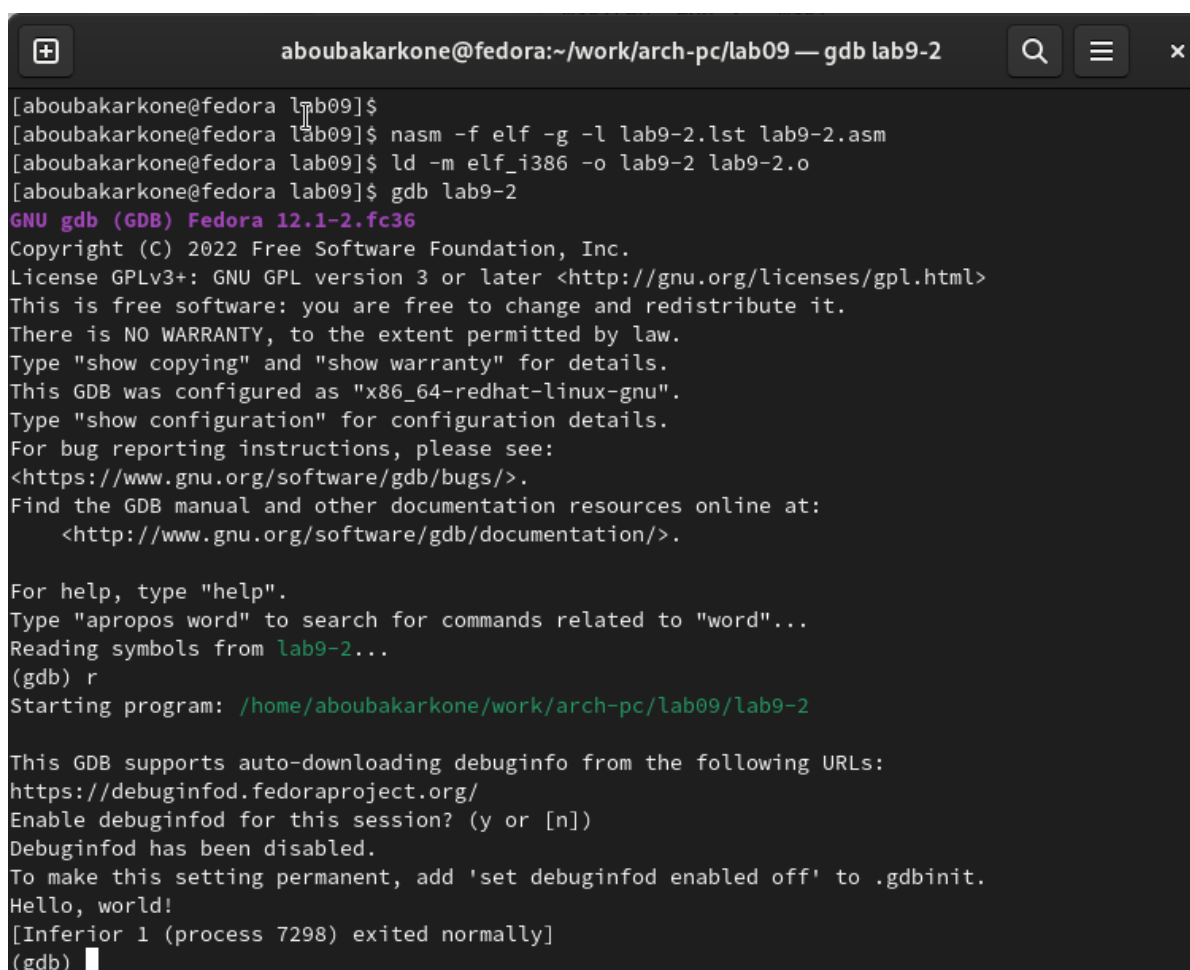
SECTION .text
global _start

_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 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).



```
aboubakarkone@fedora:~/work/arch-pc/lab09 — gdb lab9-2
[aboubakarkone@fedora lab09]$
[aboubakarkone@fedora lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
[aboubakarkone@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
[aboubakarkone@fedora lab09]$ gdb lab9-2
GNU gdb (GDB) Fedora 12.1-2.fc36
Copyright (C) 2022 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/aboubakarkone/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 7298) 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 дисассимилированный код программы.

```

(gdb) r
Starting program: /home/aboubakarkone/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: Дизассимилированный код

```

(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: Дизассимилированный 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.

The screenshot shows a GDB terminal window with the title bar "aboubakarkone@fedora:~/work/arch-pc/lab09 — gdb lab9-2". The main display area shows "[Register Values Unavailable]". Below this, a list of assembly instructions is displayed, with the first instruction highlighted in blue:

```
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
```

Below the assembly list, the status bar shows "native process 7304 In: _start L11 PC: 0x8049000". The command history shows the following commands:

```
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 22.
(gdb) i b
```

The output of the `i b` command is shown below:

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

The prompt "(gdb) " is visible at the bottom left.

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.

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

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1f0 0xffffd1f0
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 7304 In: _start L12 PC: 0x8049005
--Type <RET> for more, q to quit, c to continue without paging--eflags 0x202
[ IF ]
cs      0x23      35
ss      0x2b      43
ds      0x2b      43
es      0x2b      43
fs      0x0      0
gs      0x0      0
(gdb) si
(gdb) si
```

Рис. 2.10: Изменение регистров

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

eax	0x8	8
ecx	0x804a000	134520832
edx	0x8	8
ebx	0x1	1
esp	0xffffd1f0	0xffffd1f0
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0

. 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 current state of the process: "native process 7304 In: _start" with "L16" and "PC: 0x8049016". Below this, it lists segment registers:

ds	0x2b	43
es	0x2b	43
fs	0x0	0
gs	0x0	0

. At the very bottom, there are several "(gdb) si" commands, indicating step-through debugging.

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`.


```
native process 7304 In: _start
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>:      "Lor!d!\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`.

```
native process 7304 In: _start
```

```
$1 = 8
```

```
$2 = 8
```

```
(gdb) p/t $eax
```

```
$3 = 1000
```

```
(gdb) p/s $ecx
```

```
$4 = 134520832
```

```
(gdb) p/x $ecx
```

```
$5 = 0x804a000
```

```
(gdb) p/s $edx
```

```
$6 = 8
```

```
(gdb) p/t $edx
```

```
$7 = 1000
```

```
(gdb) p/x $edx
```

```
$8 = 0x8
```

```
(gdb) 
```

Рис. 2.13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx

```
native process 7304 In: _start
(gdb) p/x $ecx
$5 = 0x804a000
(gdb) p/s $edx
$6 = 8
(gdb) p/t $edx
$7 = 1000
(gdb) p/x $edx
$8 = 0x8
(gdb) set $ebx='2'
(gdb) p/s $ebx
$9 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$10 = 2
(gdb) █
```

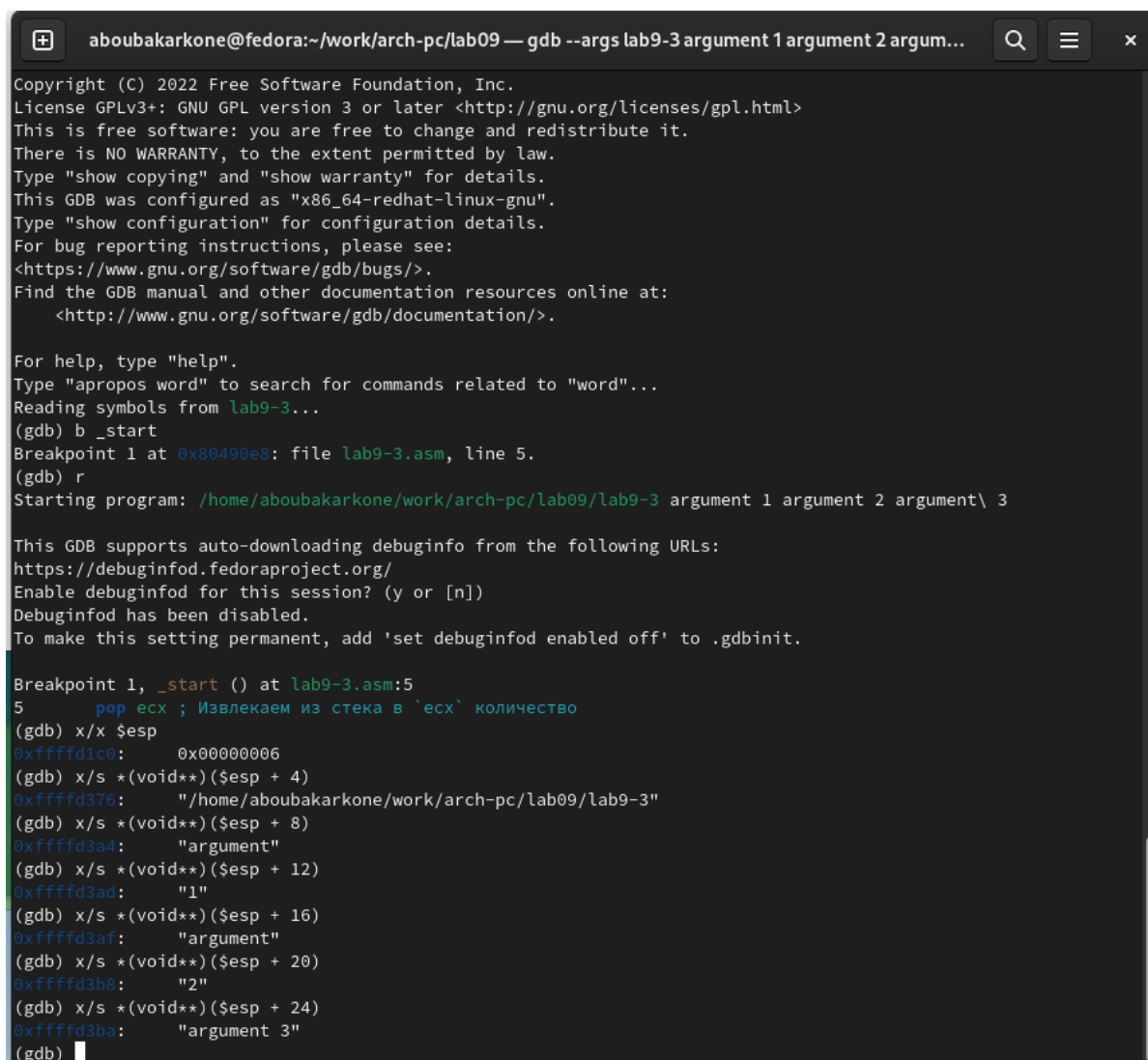
Рис. 2.14: Вывод значения регистра

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.



```
aboubakarkone@fedora:~/work/arch-pc/lab09 — gdb --args lab9-3 argument 1 argument 2 argum...
Copyright (C) 2022 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) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/aboubakarkone/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.

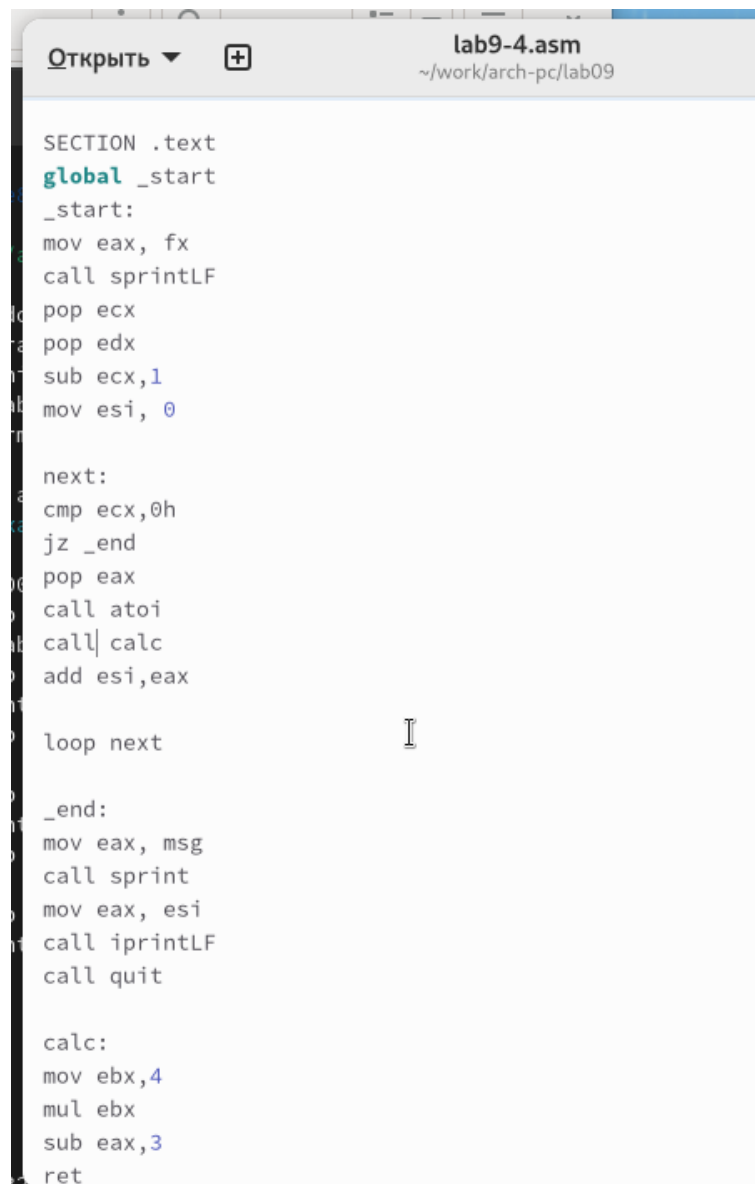
Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd1c0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd376: "/home/aboubakarkone/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd3a4: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd3ad: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd3af: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd3b8: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd3ba: "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.



```
SECTION .text
global _start
_start:
mov eax, fx
call sprintf
pop ecx
pop edx
sub ecx, 1
mov esi, 0

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

loop next

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

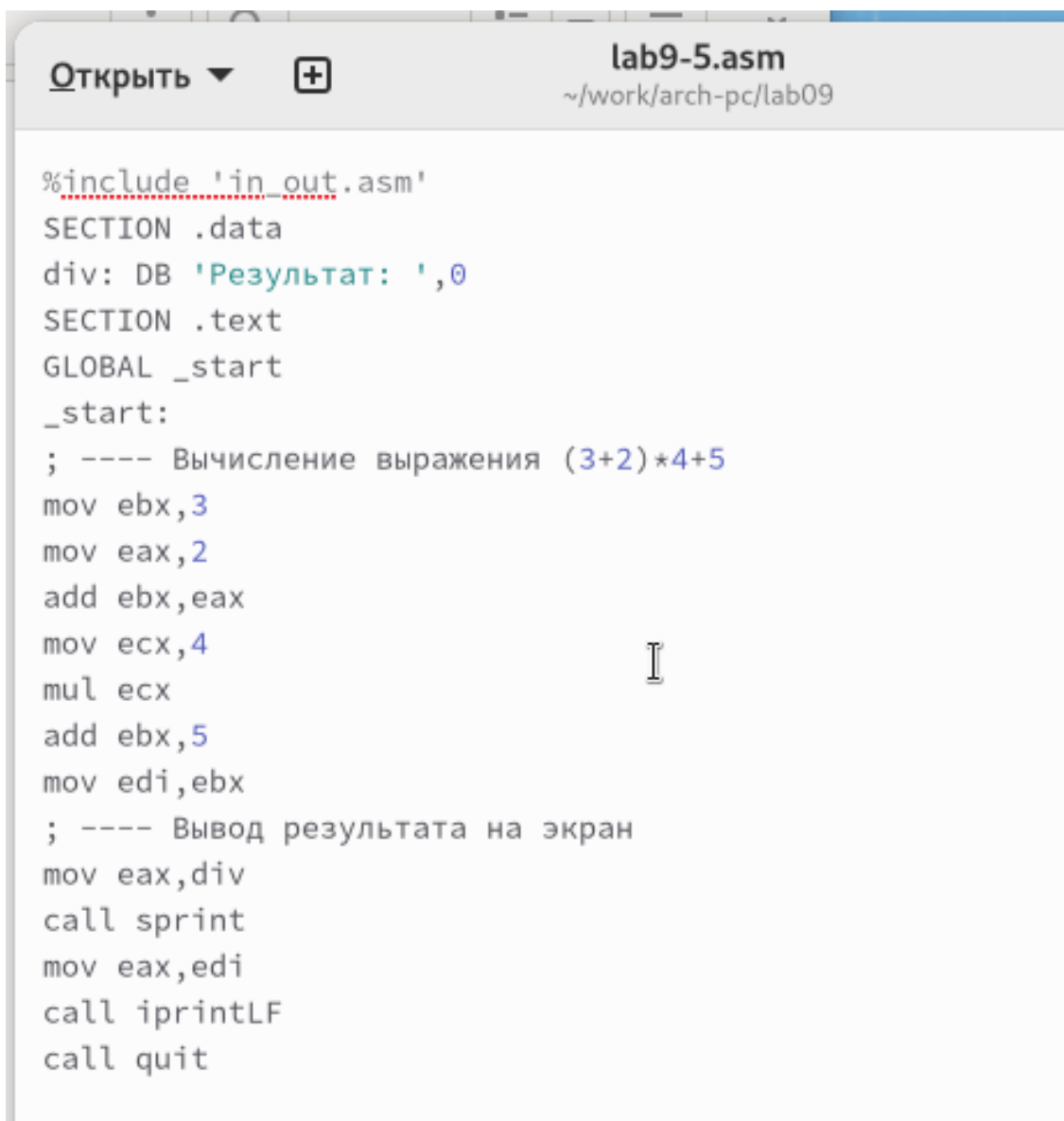
calc:
mov ebx, 4
mul ebx
sub eax, 3
ret
```

Рис. 2.16: Программа lab9-4.asm

```
[aboubakarkone@fedora lab09]$  
[aboubakarkone@fedora lab09]$ nasm -f elf lab9-4.asm  
[aboubakarkone@fedora lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o  
[aboubakarkone@fedora lab09]$ ./lab9-4 1  
f(x)= 4x - 3  
Результат: 1  
[aboubakarkone@fedora lab09]$ ./lab9-4 1 2 3 4  
f(x)= 4x - 3  
Результат: 28  
[aboubakarkone@fedora 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.



```
lab9-5.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

The screenshot shows a GDB window titled "aboubakarkone@fedora:~/work/arch-pc/lab09 — gdb lab9-5". The "Register group: general" section displays the following values:

Register	Value	Register	Value		
eax	0x804a000	134520832	ecx	0x4	4
edx	0x0	0	ebx	0xa	10
esp	0xffffd1f0	0xffffd1f0	ebp	0x0	0x0
esi	0x0	0	edi	0xa	10
eip	0x8049105	0x8049105 <_start+>	eflags	0x206	[PF IF]
cs	0x23	35	ss	0x2b	43
ds	0x2b	43	es	0x2b	43
fs	0x0	0	gs	0x0	0

The assembly window shows the following code:

```
B+ 0x80490e8 <_start>    mov     ebx,0x3
0x80490ed <_start+5>    mov     eax,0x2
0x80490f2 <_start+10>   add     ebx,eax
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 <iprintf>
0x8049111 <_start+41>   call    0x80490db <quit>
0x8049116             add     BYTE PTR [eax],al
```

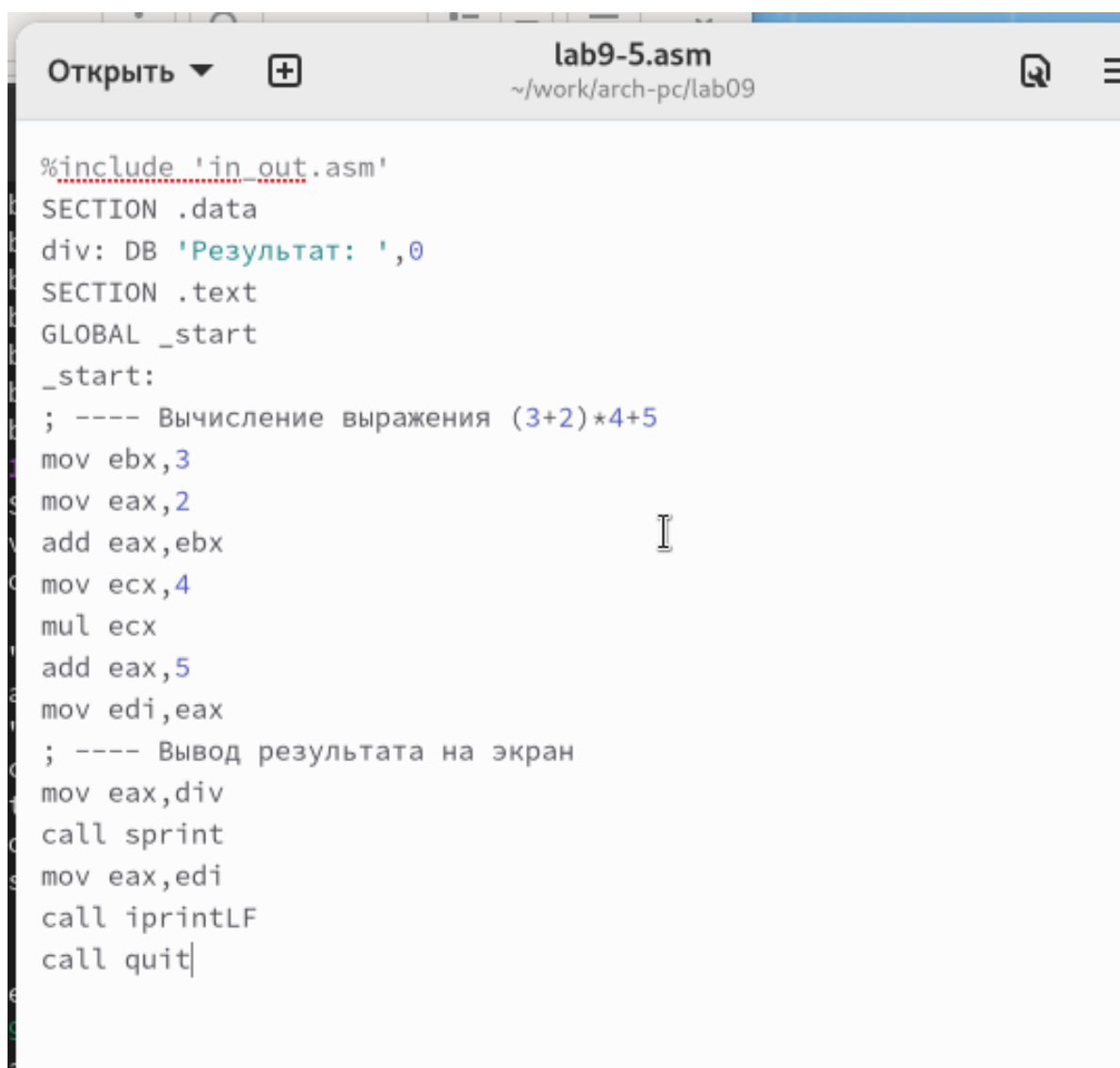
The status bar indicates "native process 7559 In: _start" and "L17 PC: 0x8049105". A message from debuginfod is also visible: "https://debuginfod.fedoraproject.org/ Enable debuginfod for this session? (y or [n]) Debuginfod has been disabled. To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit."

The command window shows the following commands and output:

```
Breakpoint 1, _start () at lab9-5.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: Отладка

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



```
Открыть ▾ + lab9-5.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: Код исправлен

```

aboutakarkone@fedora:~/work/arch-pc/lab09 — gdb lab9-5
eax      0x19      25      ecx      0x4       4
edx      0x0       0       ebx      0x3       3
esp      0xffffd1f0 0xffffd1f0 ebp      0x0       0x0
esi      0x0       0       edi      0x19      25
eip      0x8049100 0x8049100 <_start+ eflags  0x202      [ IF ]
cs       0x23      35      ss       0x2b       43
ds       0x2b      43      es       0x2b       43
fs       0x0       0       gs       0x0       0

B+ 0x80490e8 <_start>    mov     ebx,0x3
0x8049100 <_start+24>    mov     eax,0x804a000
0x8049105 <_start+29>    call    0x804900f <sprint>
0x804910a <_start+34>    mov     ecx,0di
0x804910c <_start+36>    call    0x8049086 <iprintLF>
0x8049111 <_start+41>    call    0x80490db <quit>

>                                04a000
                                rint>

                                86 <iprintLF>

                                l

native process 7612 In: _start L16 PC: 0x8049100
Enable No process In: sion? (y or [n]) Debuginfod has been disabled. L?? PC: ??

Breakpoint 1, _start () at lab9-5.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) cont
Continuing.
Результат: 25
[Inferior 1 (process 7612) exited normally]
(gdb)

```

Рис. 2.21: Проверка работы

3 Выводы

Освоили работу с подпрограммами и отладчиком.