

# **Отчёт по лабораторной работе 9**

**Архитектура компьютера**

Цыкунова Екатерина Михайловна НКАбд-05-24

# 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 . . . . . | 21        |
| <b>3</b> | <b>Выводы</b>                                | <b>27</b> |

## Список иллюстраций

|      |   |    |
|------|---|----|
| 2.1  | Программа lab9-1.asm . . . . .                    | 7  |
| 2.2  | Запуск программы lab9-1.asm . . . . .             | 7  |
| 2.3  | Программа lab9-1.asm . . . . .                    | 8  |
| 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 prog-1.asm . . . . .                    | 21 |
| 2.17 | Запуск программы prog-1.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

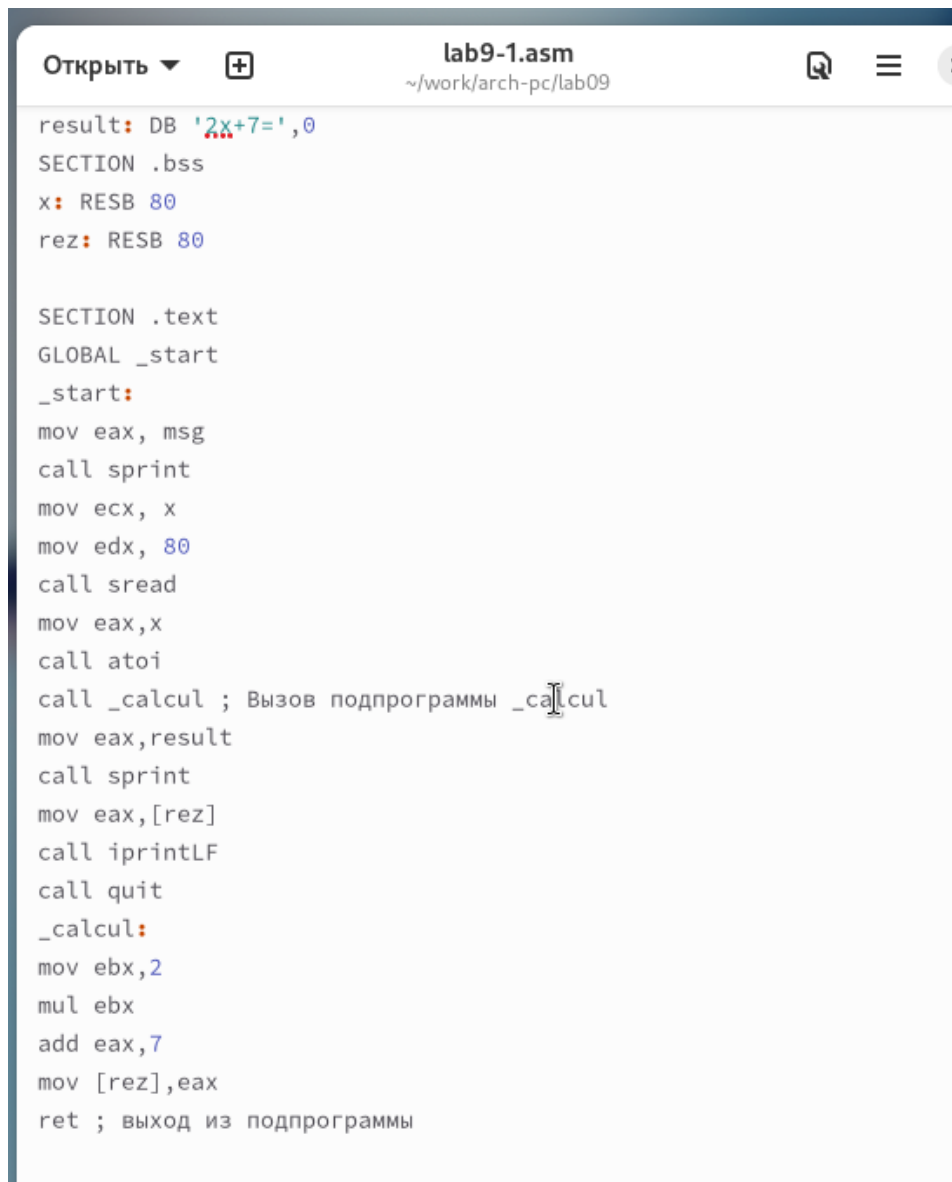
Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

## 2 Выполнение лабораторной работы

### 2.1 Реализация подпрограмм в NASM

Сначала я создала новую папку, чтобы выполнять лабораторную работу номер 9, и перешла в нее. Затем я создала файл с именем lab9-1.asm.

В качестве примера, я рассмотрела программу, которая вычисляет арифметическое выражение  $f(x) = 2x + 7$  с использованием подпрограммы calcul. В этом примере значение переменной  $x$  вводится с клавиатуры, а само выражение вычисляется внутри подпрограммы.(рис. 2.1) (рис. 2.2)

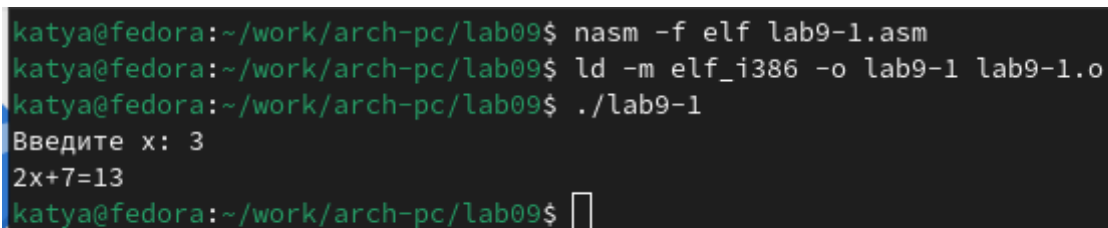


```
Открыть ▾ + lab9-1.asm
~/work/arch-pc/lab09

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
```

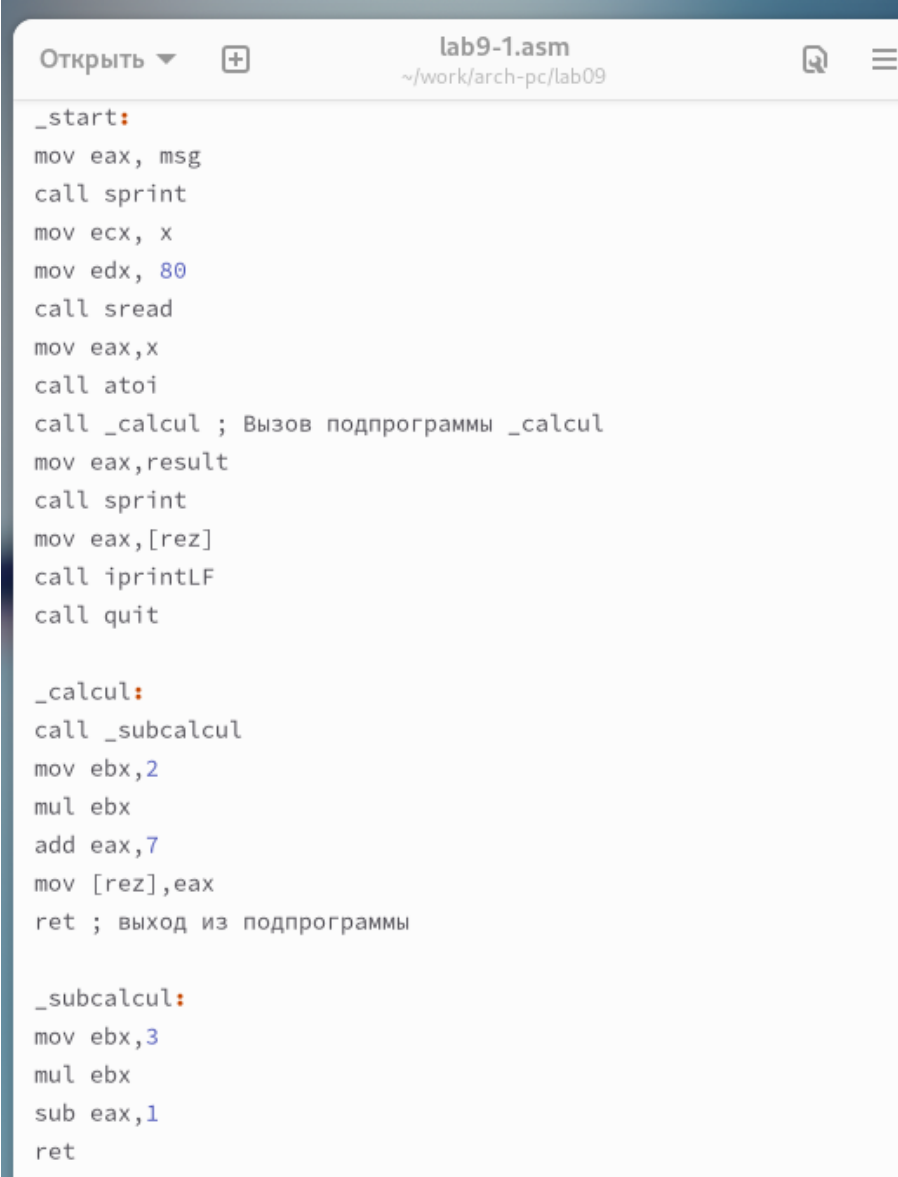
Рис. 2.1: Программа lab9-1.asm



```
katya@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
katya@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
katya@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 3
2x+7=13
katya@fedora:~/work/arch-pc/lab09$
```

Рис. 2.2: Запуск программы lab9-1.asm

После этого я внесла изменения в текст программы, добавив подпрограмму `subcalcul` внутрь подпрограммы `calcul`. Это позволило вычислить составное выражение  $f(g(x))$ , где значение  $x$  также вводится с клавиатуры. Функции определены следующим образом:  $f(x) = 2x + 7$ ,  $g(x) = 3x - 1$ . (рис. 2.3) (рис. 2.4)



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

_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



```
katya@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
katya@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
katya@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 3
2x+7=13
katya@fedora:~/work/arch-pc/lab09$
katya@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
katya@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
katya@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 3
2(3x-1)+7=23
katya@fedora:~/work/arch-pc/lab09$
```

Рис. 2.4: Запуск программы lab9-1.asm

## 2.2 Отладка программ с помощью GDB

Я создала файл с названием lab9-2.asm, в котором содержится программа из Листинга 9.2. Эта программа отвечает за вывод сообщения “Hello world!” на экран.(рис. 2.5)



```
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

После этого я скомпилировала файл и получила исполняемый файл. Чтобы добавить отладочную информацию для работы с отладчиком GDB, я использовала ключ “-g”. Затем я загрузила полученный исполняемый файл в отладчик GDB и проверила его работу, запустив программу с помощью команды “run” или “r”. (рис. 2.6)

```

katya@fedora:~/work/arch-pc/lab09$
katya@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
katya@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
katya@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/katya/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 3796) exited normally]
(gdb)

```

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

Для более детального анализа программы я установила точку остановки на метке “start”, с которой начинается выполнение любой ассемблерной программы, и запустила ее. Затем я просмотрела дизассемблированный код программы.(рис. 2.7) (рис. 2.8)

```
katya@fedora:~/work/arch-pc/lab09 — gdb lab9-2
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) r
Starting program: /home/katya/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 3796) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/katya/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

```
katya@fedora:~/work/arch-pc/lab09 — gdb 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) 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

Чтобы проверить точку остановки по имени метки “\_start”, я использовала команду “info breakpoints” или “i b”. Затем я установила еще одну точку остановки по адресу инструкции, определив адрес предпоследней инструкции “mov ebx, 0x0”. (рис. 2.9)

The screenshot shows the GDB debugger interface. The top window, titled 'Register group: general', displays the state of the general-purpose registers. Below it, the assembly window shows the instructions being executed, starting from address 0x8049000. The assembly instructions are as follows:

| Address   | Disassembly                    |
|-----------|--------------------------------|
| 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           |

The bottom window shows the GDB command prompt with the following commands and output:

```
(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

В отладчике GDB у меня была возможность просматривать содержимое ячеек памяти и регистров, а также изменять значения регистров и переменных. Я выполнила 5 инструкций с помощью команды 'stepi' (сокращенно 'si') и отслеживала изменение значений регистров. (рис. 2.10) (рис. 2.11)

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

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]

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

Native process 3809 (asm) In: _start L12 PC: 0x8049005
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si
(gdb) 
```

Рис. 2.10: Изменение регистров

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

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]

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

Native process 3809 (asm) In: _start L16 PC: 0x8049016
--Type <RET> for more, q to quit, c to continue without paging--
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) si
(gdb) 
```

Рис. 2.11: Изменение регистров

Я также просмотрела значение переменной `msg1` по имени и получила нужные данные. Чтобы изменить значение регистра или ячейки памяти, я использовала команду `'set'`, указав имя регистра или адрес в качестве аргумента. Я изменила первый символ переменной `msg1`. (рис. 2.12) (рис. 2.13)



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

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]

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

native process 3809 (asm) In: _start      L16  PC: 0x8049016
(gdb) si
(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

The screenshot shows a GDB terminal window with the title bar 'katya@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), `ecx` (0x804a000), `edx` (0x8), `ebx` (0x1), `esp` (0xffffd120), `ebp` (0x0), `esi` (0x0), `edi` (0x0), `eip` (0x8049016), and `eflags` (0x202). The middle section displays assembly code with addresses and instructions, such as `mov eax,0x4` at address `0x8049000`. The bottom section shows the output of GDB commands: `(gdb) p/s $ecx` resulting in `$3 = 134520832`, `(gdb) p/x $ecx` resulting in `$4 = 0x804a000`, `(gdb) p/s $edx` resulting in `$5 = 8`, and `(gdb) p/t $edx` resulting in `$6 = 1000`. The status bar at the bottom indicates 'native process 3809 (asm) In: \_start' and 'PC: 0x8049016'.

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

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]

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

native process 3809 (asm) In: _start                               L16   PC: 0x8049016
$2 = 1000
(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

Также, с помощью команды 'set', я изменила значение регистра `ebx` на нужное значение.(рис. 2.14)

The screenshot shows a GDB terminal window with the title bar "katya@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 several registers: `eax` (0x8, 8), `ecx` (0x804a000, 134520832), `edx` (0x8, 8), `ebx` (0x2, 2), `esp` (0xffffd120, 0xffffd120), `ebp` (0x0, 0x0), `esi` (0x0, 0), `edi` (0x0, 0), `eip` (0x8049016, 0x8049016 <\_start+22>), and `eflags` (0x202, [ IF ]). The middle section shows a list of assembly instructions with their addresses and disassembled forms, with the instruction at address 0x8049016 (`mov eax, 0x4`) highlighted. The bottom section shows the command prompt and a series of GDB commands and their outputs: `$5 = 8`, `(gdb) p/t $edx`, `$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`, and `(gdb)` followed by a cursor.

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

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]

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

Native process 3809 (asm) In: _start L16 PC: 0x8049016
$5 = 8
(gdb) p/t $edx
$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
(gdb)
```

Рис. 2.14: Вывод значения регистра

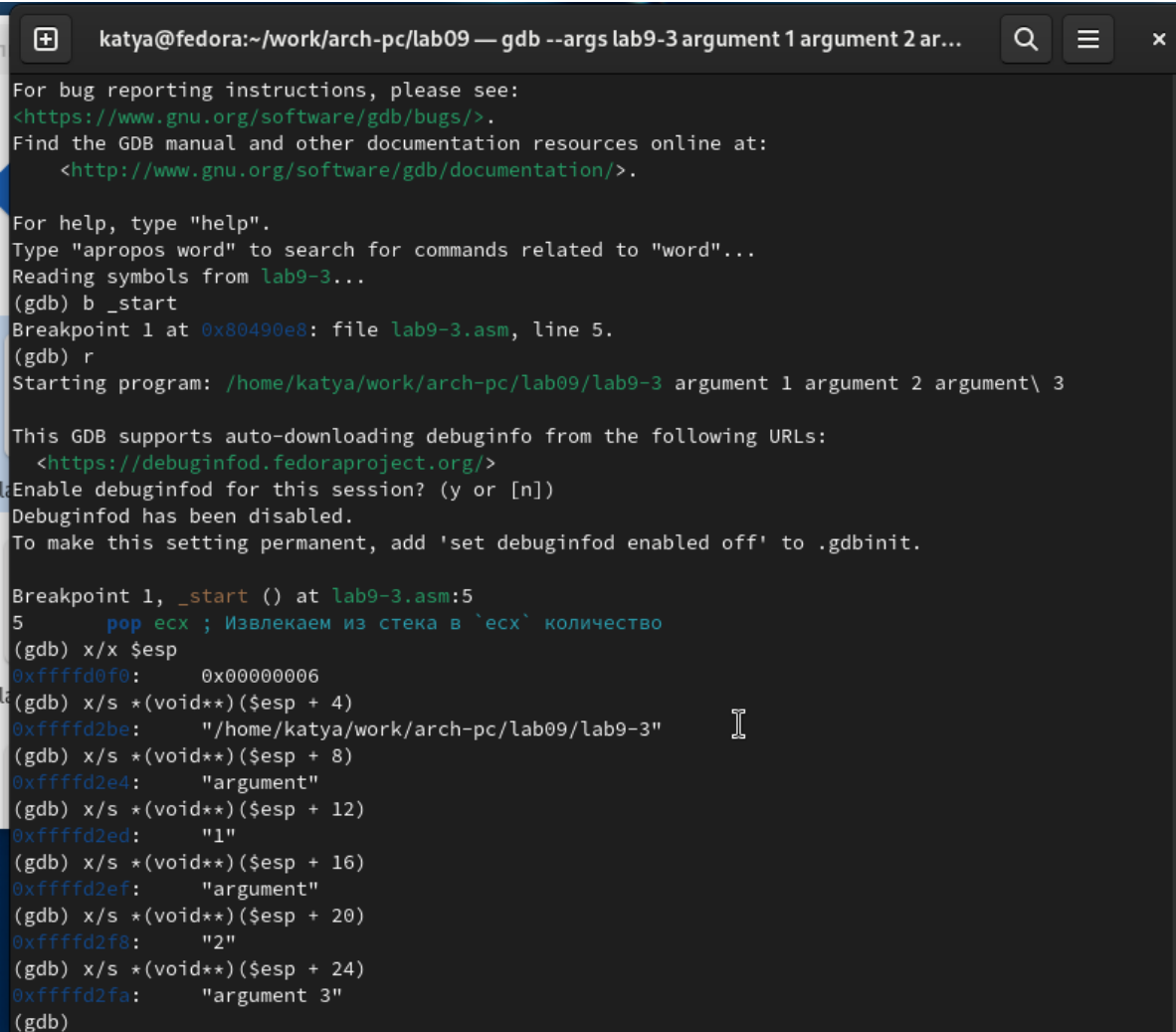
Я скопировала файл lab8-2.asm, который был создан во время выполнения лабораторной работы №8. Этот файл содержит программу для вывода аргументов командной строки. Затем я создала исполняемый файл из скопированного файла.

Для загрузки программы с аргументами в отладчик GDB, я использовала ключ `-args` и загрузила исполняемый файл в отладчик с указанными аргументами. Я установила точку останова перед первой инструкцией программы и запустила ее.

Адрес вершины стека, где хранится количество аргументов командной строки (включая имя программы), находится в регистре `esp`. По этому адресу находится

число, указывающее количество аргументов. В данном случае я увидела, что количество аргументов равно 5, включая имя программы lab9-3 и сами аргументы: аргумент1, аргумент2 и 'аргумент 3'.

Я также просмотрела остальные позиции стека. По адресу [esp+4] находится адрес в памяти, где располагается имя программы. По адресу [esp+8] хранится адрес первого аргумента, по адресу [esp+12] - второго и так далее. Шаг изменения адреса равен 4, так как каждый следующий адрес на стеке находится на расстоянии 4 байт от предыдущего ([esp+4], [esp+8], [esp+12]). (рис. 2.15)



```
katya@fedora:~/work/arch-pc/lab09 — gdb --args lab9-3 argument 1 argument 2 ar...
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/katya/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

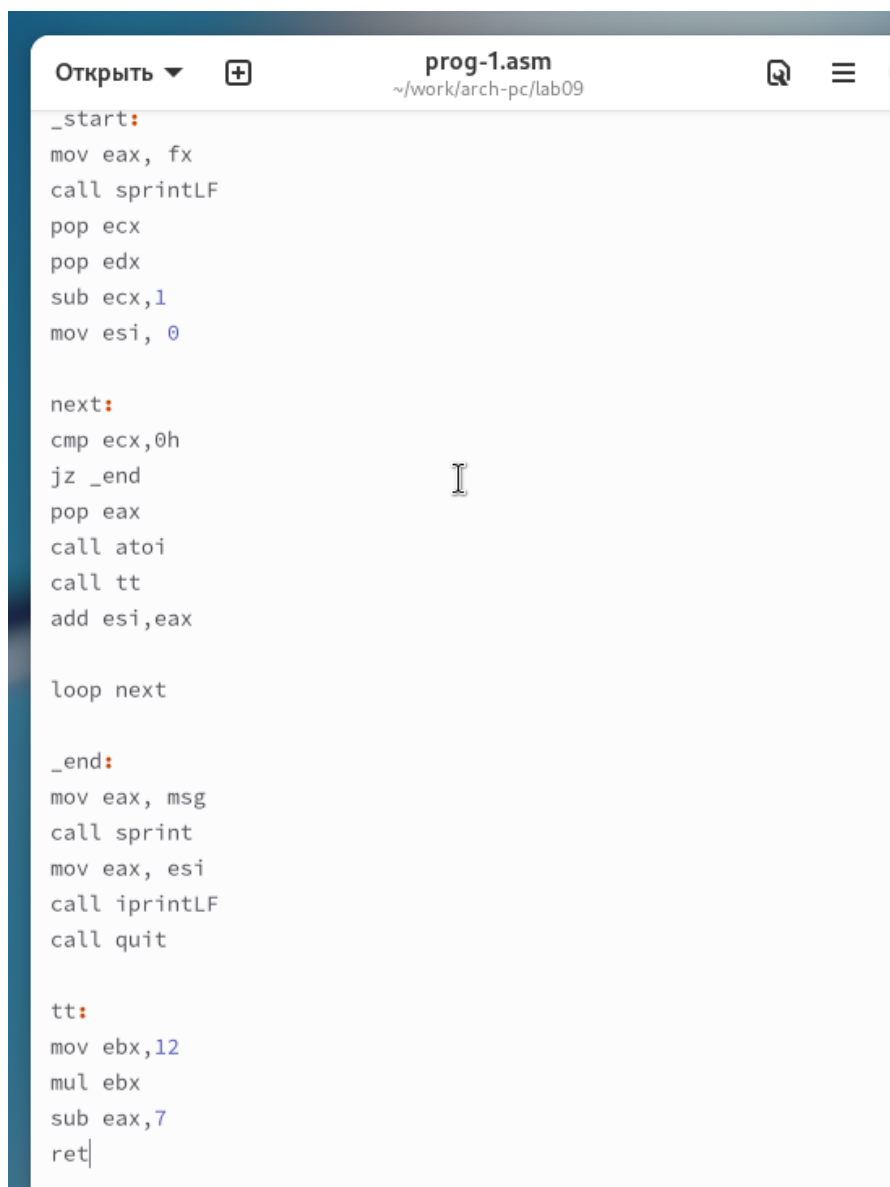
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd0f0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd2be: "/home/katya/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd2e4: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd2ed: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd2ef: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd2f8: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd2fa: "argument 3"
(gdb)
```

Рис. 2.15: Вывод значения регистра

## 2.3 Задание для самостоятельной работы

Я решила преобразовать программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), добавив вычисление значения функции  $f(x)$  в виде подпрограммы. (рис. 2.16) (рис. 2.17)



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

_start:
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 tt
add esi,eax

loop next

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

tt:
mov ebx,12
mul ebx
sub eax,7
ret
```

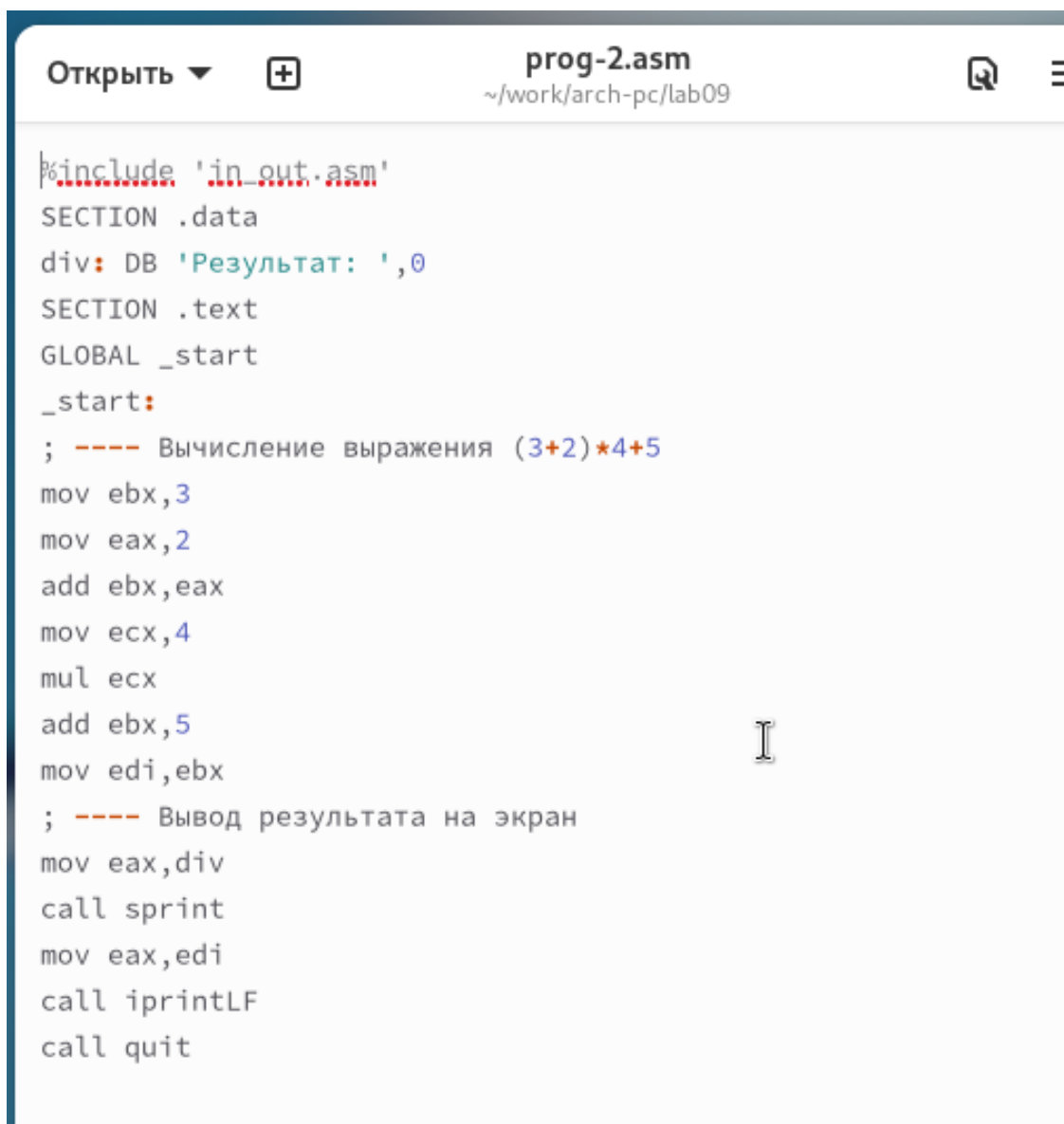
Рис. 2.16: Программа prog-1.asm

```
katya@fedora:~/work/arch-pc/lab09$ nasm -f elf prog-1.asm
katya@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 prog-1.o -o prog-1
katya@fedora:~/work/arch-pc/lab09$ ./prog-1
f(x)= 12x - 7
Результат: 0
katya@fedora:~/work/arch-pc/lab09$ ./prog-1 1
f(x)= 12x - 7
Результат: 5
katya@fedora:~/work/arch-pc/lab09$ ./prog-1 3 4 2 1
f(x)= 12x - 7
Результат: 92
katya@fedora:~/work/arch-pc/lab09$
```

Рис. 2.17: Запуск программы prog-1.asm

В листинге представлена программа для вычисления выражения  $(3 + 2) * 4 + 5$ . Однако, при запуске программы, я обнаружила, что она дает неверный результат. Чтобы разобраться в причинах, я провела анализ изменений значений регистров с помощью отладчика GDB.

В результате анализа, я обнаружила, что порядок аргументов у инструкции `add` был перепутан. Кроме того, я заметила, что по окончании работы программы, значение `ebx` было отправлено в `edi` вместо `eax`. (рис. 2.18)



```
Открыть ▾ + prog-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

```
katya@fedora:~/work/arch-pc/lab09 — gdb prog-2

Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0xa      10
eip      0x8049100 0x8049100 <_start+24>
eflags   0x206    [ PF IF ]

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

native process 3930 (asm) In: _start L16 PC: 0x8049100
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

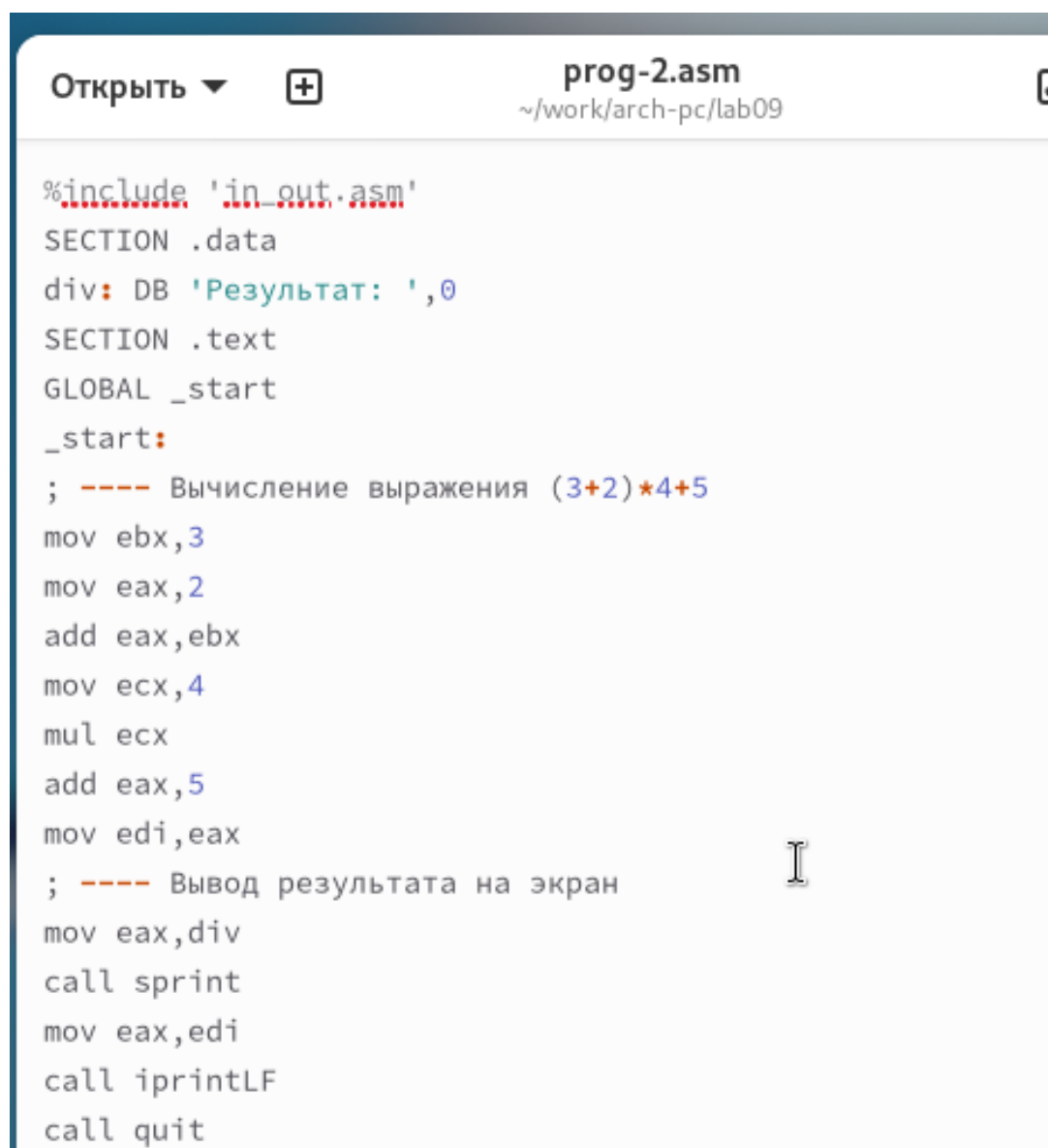
Breakpoint 1, _start () at prog-2.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.19: Отладка

Отмечу, что перепутан порядок аргументов у инструкции `add` и что по окончании работы в `edi` отправляется `ebx` вместо `eax` (рис. 2.19)

Исправленный код программы (рис. 2.20) (рис. 2.21)







```
Открыть ▾  prog-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: Код исправлен

```
katya@fedora:~/work/arch-pc/lab09 — gdb prog-2

Register group: general
eax      0x19      25
ecx      0x4       4
edx      0x0       0
ebx      0x3       3
esp      0xffffd120 0xffffd120
ebp      0x0       0x0
esi      0x0       0
edi      0x19      25
eip      0x8049100  0x8049100 <_start+24>
eflags   0x202     [ IF ]

B+ 0x80490e8 <_start>      mov     ebx,0x3
0x80490ed <_start+5>      mov     eax,0x2
0x80490f2 <_start+10>     add     eax,ebx
0x80490f4 <_start+12>     mov     ecx,0x4
0x80490f9 <_start+17>     mul     ecx
0x80490fb <_start+19>     add     eax,0x5
0x80490fe <_start+22>     mov     edi,eax
>0x8049100 <_start+24>    mov     eax,0x804a000
0x8049105 <_start+29>     call    0x804900f <sprint>
0x804910a <_start+34>     mov     eax,edi

native process 4072 (asm) In: _start L16 PC: 0x8049100
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

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

Рис. 2.21: Проверка работы

## **3 Выводы**

Освоили работу с подпрограммами и отладчиком.