

Отчёт по лабораторной работе 9

дисциплина: Архитектура компьютеров

Грачев Я. М. НПИбд-01-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация подпрограмм в NASM	6
2.2	Отладка программы с помощью GDB	9
2.3	Задание для самостоятельной работы	20
3	Выводы	27

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 intel	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

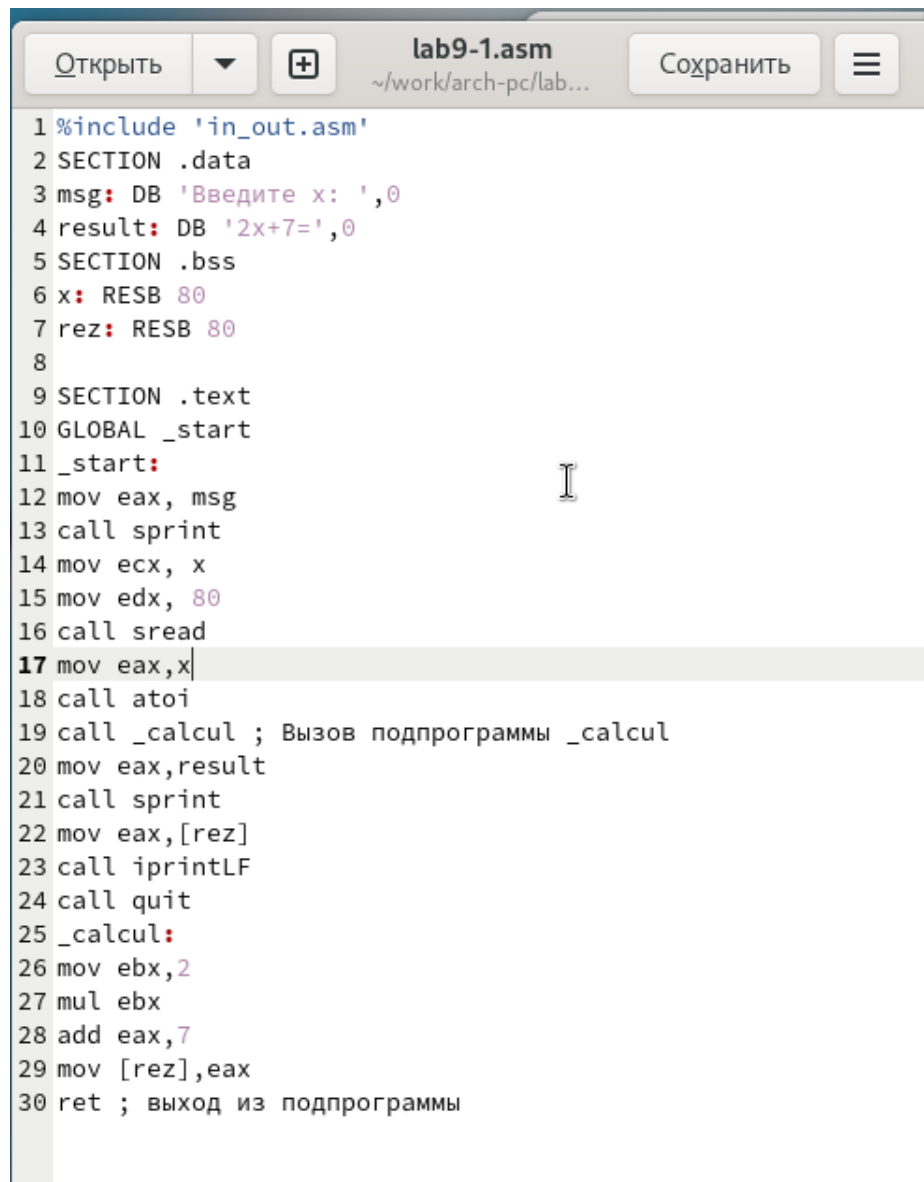
Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

2.1 Реализация подпрограмм в NASM

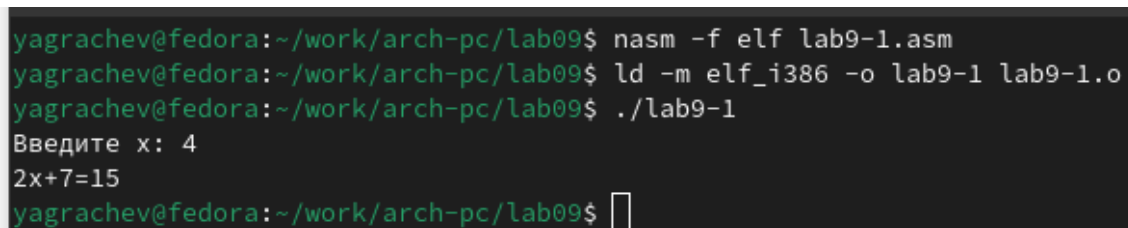
Для выполнения лабораторной работы №9 я создал новую папку и перешел в нее. Далее был создан файл с именем `lab9-1.asm`.

В качестве примера я рассмотрел 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)



```
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
```

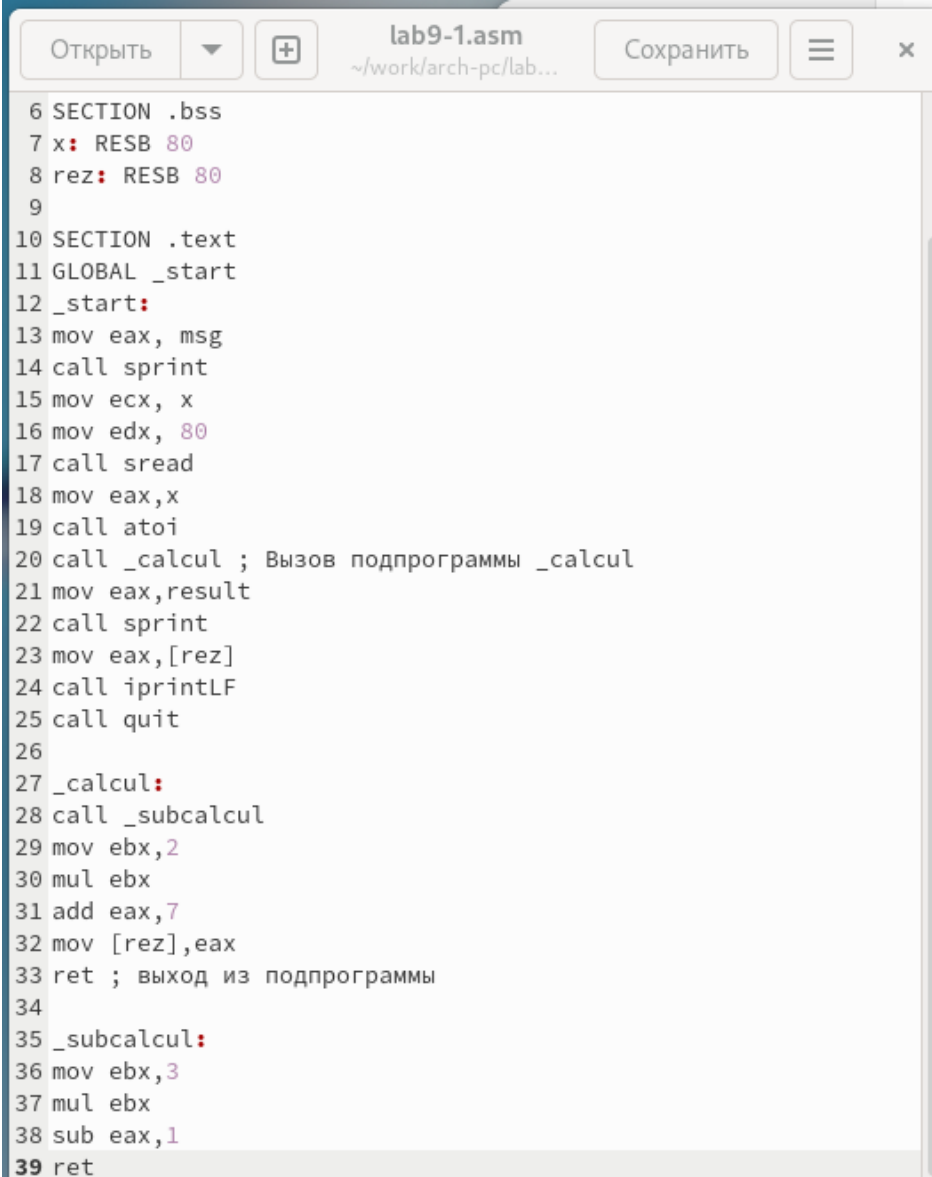
Рис. 2.1: Программа lab9-1.asm



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

Рис. 2.2: Запуск программы lab9-1.asm

После этого я внес 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)



```
6 SECTION .bss
7 x: RESB 80
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
```

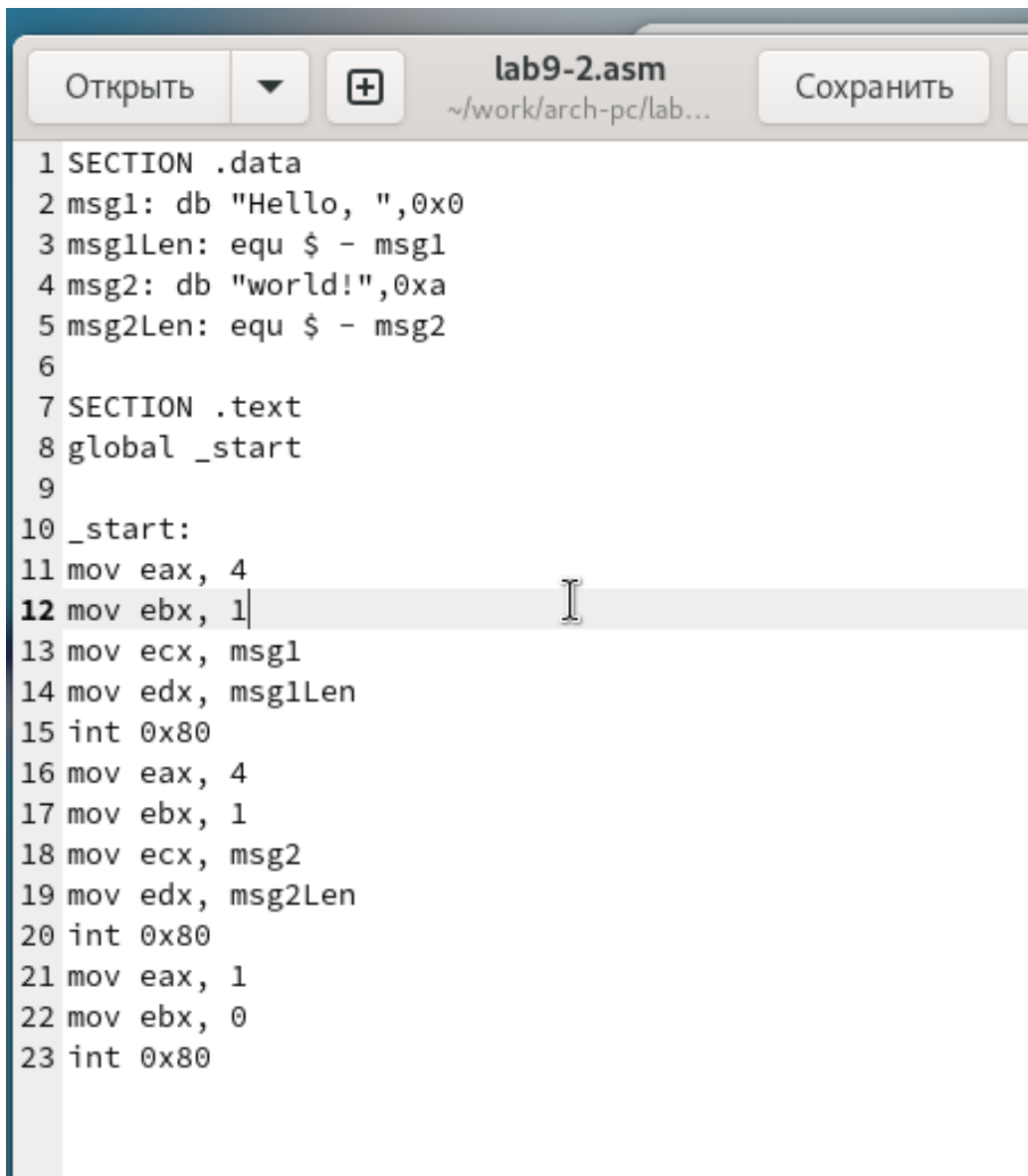
Рис. 2.3: Программа lab9-1.asm


```
yagrachev@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
yagrachev@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
yagrachev@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 4
2x+7=15
yagrachev@fedora:~/work/arch-pc/lab09$
yagrachev@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
yagrachev@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
yagrachev@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 4
2(3x-1)+7=29
yagrachev@fedora:~/work/arch-pc/lab09$
```

Рис. 2.4: Запуск программы lab9-1.asm

2.2 Отладка программы с помощью GDB

Я создал 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)



```
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

Затем я скомпилировал файл и получил исполняемый файл. Чтобы добавить отладочную информацию для работы с отладчиком GDB, я использовал ключ “-g”. После этого загрузил полученный исполняемый файл в отладчик GDB и проверил его работу, запустив программу с помощью команды “run” или “r”. (рис. 2.6)

```

yagrachev@fedora: ~/work/arch-pc/lab09$
yagrachev@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
yagrachev@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
yagrachev@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/yagrachev/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 7477) exited normally]
(gdb)

```

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

Для более детального анализа программы я установил точку остановки на метке “start”, с которой начинается выполнение любой ассемблерной программы, и запустил её. Затем я просмотрел дизассемблированный код программы. (рис. 2.7) (рис. 2.8)

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

(gdb) r
Starting program: /home/yagrachev/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 7477) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/yagrachev/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

```
yagrachev@fedora:~/work/arch-pc/lab09 — gdb lab9-2
(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 intel

Чтобы проверить точку остановки по имени метки “_start”, я использовал команду “info breakpoints” или “i b”. После этого установил ещё одну точку остановки по адресу инструкции, определив адрес предпоследней инструкции “mov ebx, 0x0”. (рис. 2.9)

```
yagrachev@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
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
0x804902a <_start+42>   int    0x80

native process 7501 (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

В отладчике GDB я имел возможность просматривать содержимое ячеек памяти и регистров, а также изменять значения регистров и переменных. Я выполнил 5 инструкций с помощью команды `stepi` (сокращенно `si`) и отслеживал изменение значений регистров. (рис. 2.10) (рис. 2.11)

```
yagrachev@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
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
0x804902a <_start+42>      int     0x80

native process 7501 (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: Изменение регистров

```
yagrachev@fedora:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xfffffd100 0xfffffd100
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
0x804902a <_start+42>     int     0x80

native process 7501 (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) si
```

Рис. 2.11: Изменение регистров

Я также просмотрел значение переменной `msg1` по имени и получил нужные данные. Для изменения значения регистра или ячейки памяти использовал команду `set`, указав имя регистра или адрес в качестве аргумента. Я изменил первый символ переменной `msg1`. (рис. 2.12) (рис. 2.13)


```
yagrachev@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
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
    0x804902a <_start+42>   int     0x80

native process 7501 (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>:      "Lor!d!\n\034"
(gdb) 
```

Рис. 2.12: Изменение значения переменной

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

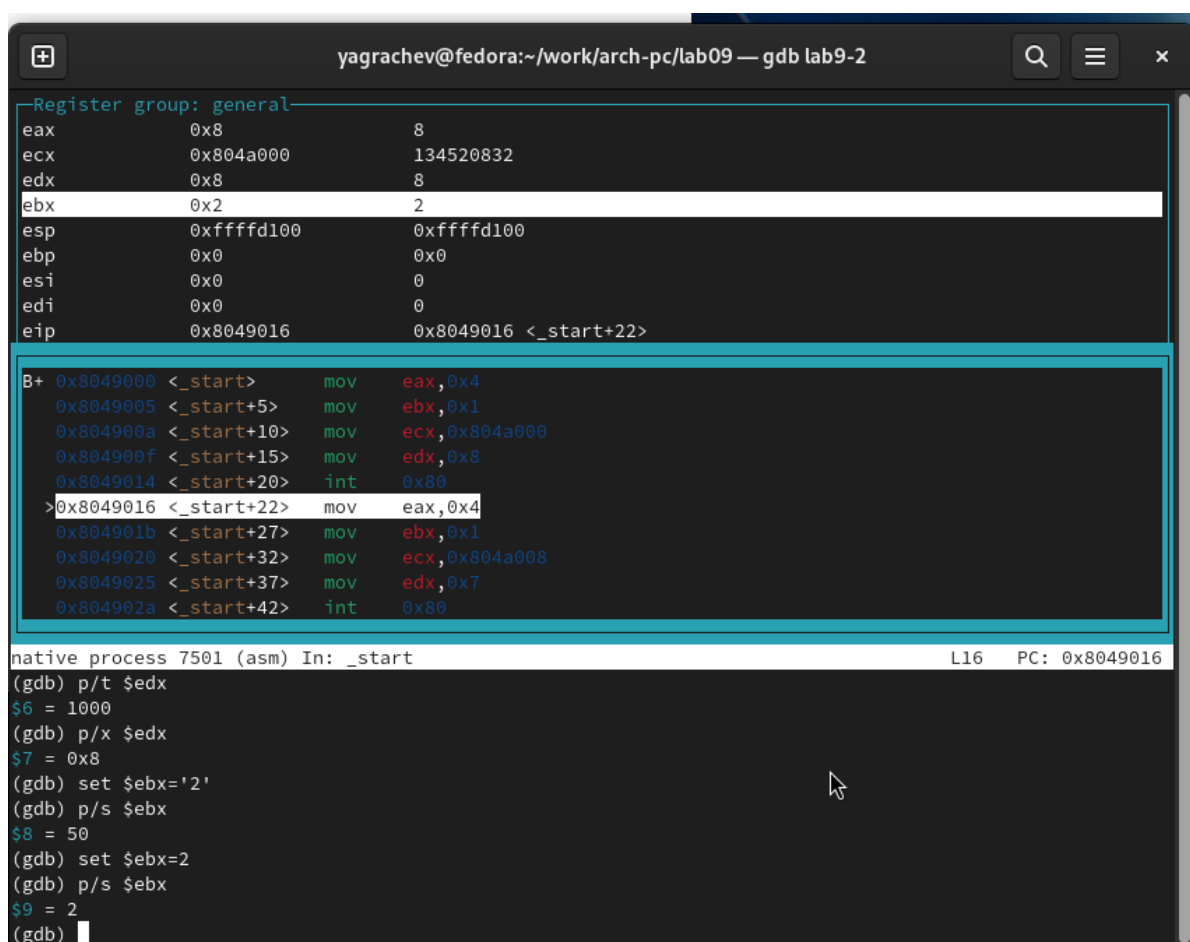
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd100 0xffffd100
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>

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
0x804902a <_start+42>     int     0x80

native process 7501 (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

Кроме того, с помощью команды `set`, я изменил значение регистра `ebx` на нужное значение. (рис. 2.14)



The screenshot shows a GDB terminal window with the title bar "yagrachev@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: `eax` (0x8), `ecx` (0x804a000), `edx` (0x8), `ebx` (0x2), `esp` (0xffffd100), `ebp` (0x0), `esi` (0x0), `edi` (0x0), and `eip` (0x8049016). The middle section shows assembly code with addresses and instructions: `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`, 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

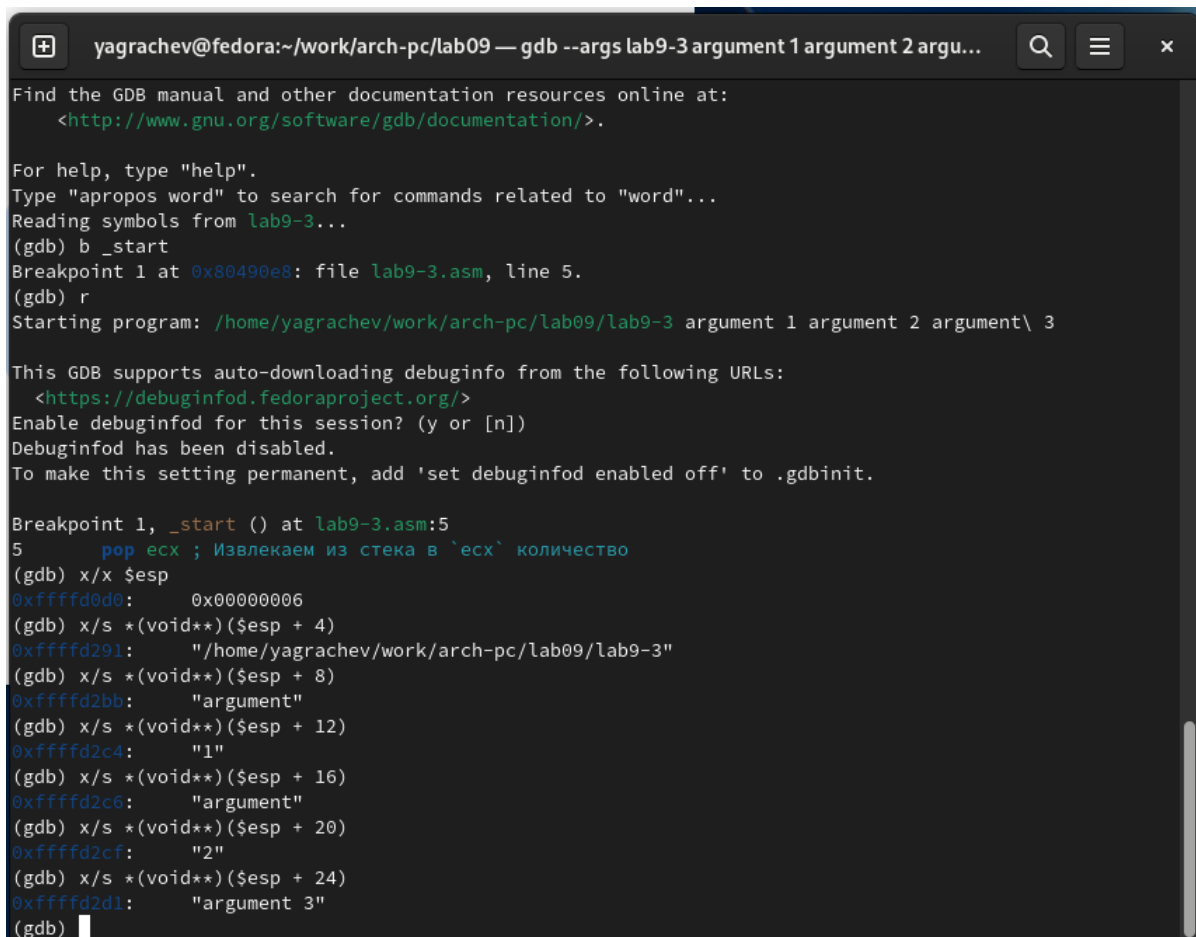
Я скопировал файл `lab8-2.asm`, который был создан в ходе выполнения лабораторной работы №8. Этот файл содержит программу для вывода аргументов командной строки. После этого создал исполняемый файл из скопированного файла.

Для загрузки программы с аргументами в отладчик GDB, использовал ключ `--args` и загрузил исполняемый файл в отладчик с указанными аргументами. Я установил точку останова перед первой инструкцией программы и запустил её.

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 количество аргументов равно 5, включая имя программы `lab9-3` и сами аргументы: `аргумент1`,

аргумент2 и аргумент 3.

Я также просмотрел 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)



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

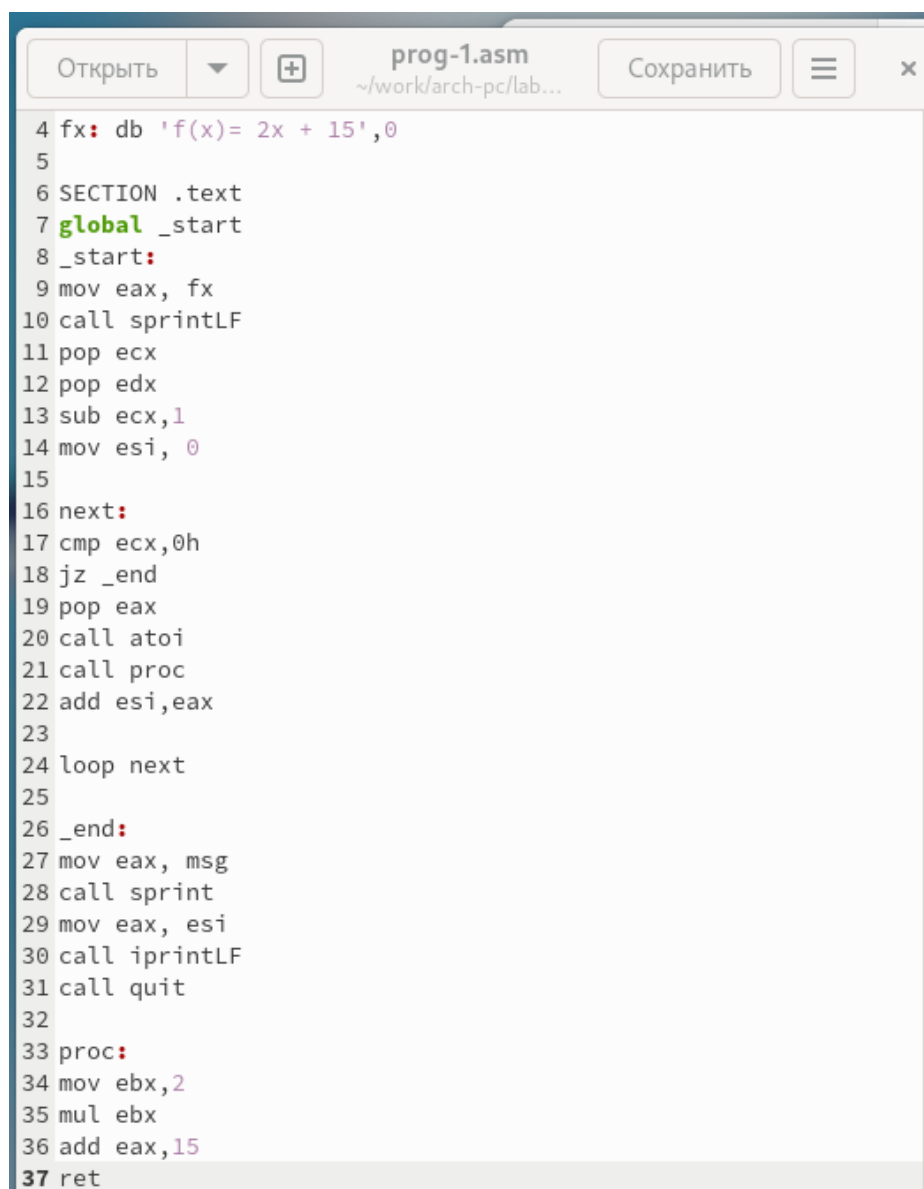
Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd0d0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd291: "/home/yagrachev/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd2bb: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd2c4: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd2c6: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd2cf: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd2d1: "argument 3"
(gdb)
```

Рис. 2.15: Вывод значения регистра

2.3 Задание для самостоятельной работы

Я решил 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)



```
4 fx: db 'f(x)= 2x + 15',0
5
6 SECTION .text
7 global _start
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 proc
22 add esi,eax
23
24 loop next
25
26 _end:
27 mov eax, msg
28 call sprintf
29 mov eax, esi
30 call iprintf
31 call quit
32
33 proc:
34 mov ebx,2
35 mul ebx
36 add eax,15
37 ret
```

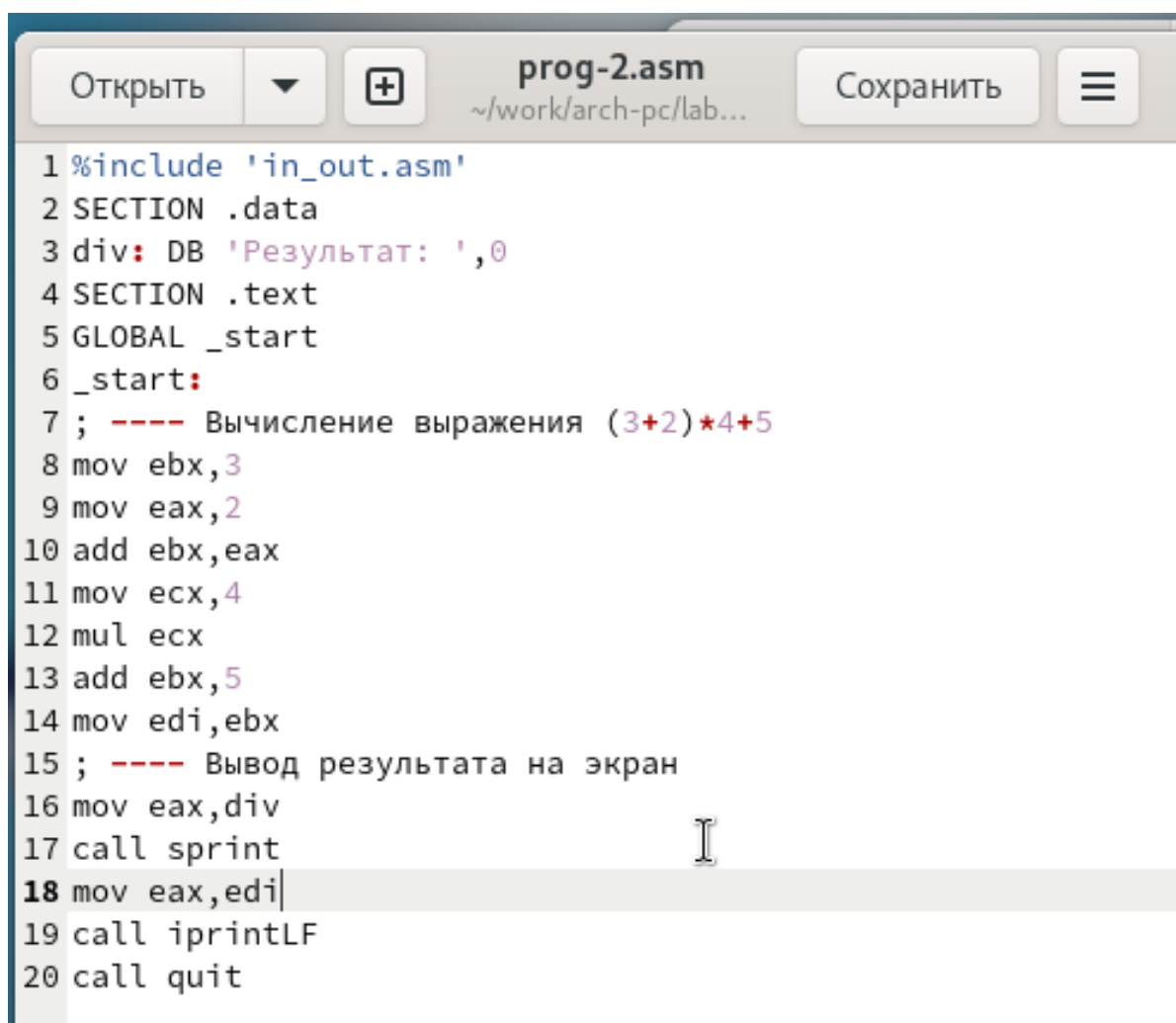
Рис. 2.16: Программа prog-1.asm

```
yagrachev@fedora:~/work/arch-pc/lab09$  
yagrachev@fedora:~/work/arch-pc/lab09$ nasm -f elf prog-1.asm  
yagrachev@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 prog-1.o -o prog-1  
yagrachev@fedora:~/work/arch-pc/lab09$ ./prog-1 3  
f(x)= 2x + 15  
Результат: 21  
yagrachev@fedora:~/work/arch-pc/lab09$ ./prog-1 6 4 1 3 1  
f(x)= 2x + 15  
Результат: 105  
yagrachev@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 обнаружил, что она даёт неверный результат. Для выявления причин я провел анализ изменений значений регистров с помощью отладчика GDB.

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



```
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

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

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
eip      0x8049100 0x8049100 <_start+24>

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

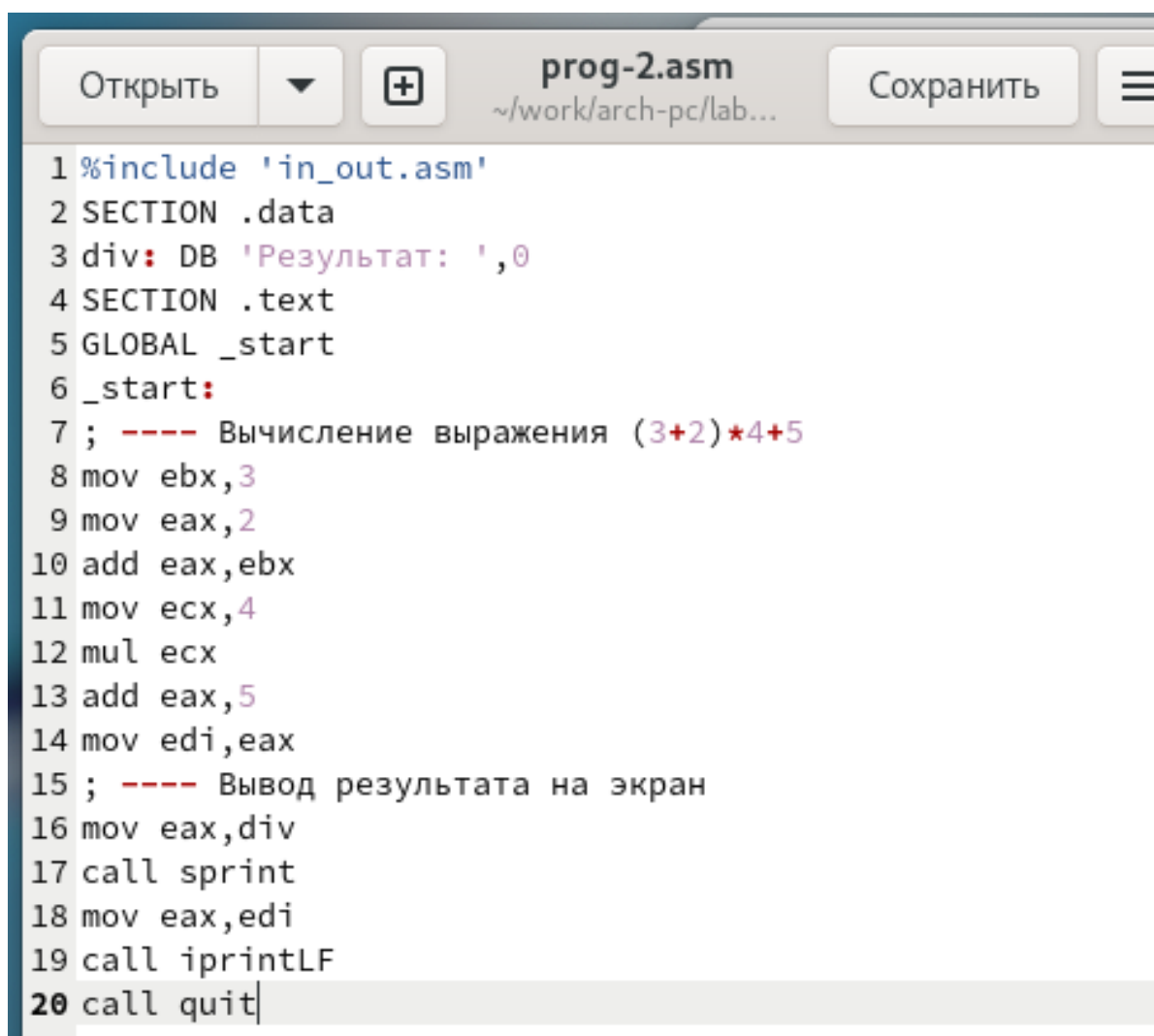
native process 7629 (asm) In: _start L16 PC: 0x8049100
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

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) 
```

Рис. 2.19: Отладка

Отметим, 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)



```
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
```

Рис. 2.20: Код исправлен

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

Register group: general
eax      0x19      25
ecx      0x4       4
edx      0x0       0
ebx      0x3       3
esp      0xffffd100 0xffffd100
ebp      0x0       0x0
esi      0x0       0
edi      0x0       0
eip      0x80490fe 0x80490fe <_start+22>

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 7674 (asm) In: _start L14 PC: 0x80490fe
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
```

Рис. 2.21: Проверка работы

3 Выводы

В ходе работы я освоил работу с подпрограммами и отладчиком.